Information Operations Across Infospheres

Annual Report

Prepared by

The University of Texas at Dallas

Submitted to:
Air Force Office of Scientific Research

November 30, 2008

Under
Contract: FA9550-06-1-0045

Period of Performance:
December 1, 2007 – November 30, 2008

Subcontractor:
The University of Texas
at San Antonio

EXECUTIVE SUMMARY OF PROJECT

There is a critical need for organizations to share data within and across infospheres and form coalitions so that analysts could examine the data, mine the data, and make effective decisions. Each organization could share information within its infosphere. An infosphere may consist of the data, applications and services that are needed for its operation. Organizations may share data with one another across what is called a global infosphere that spans multiple infospheres. It is critical that the war fighters get timely information. Furthermore, secure data and information sharing is an important requirement. The challenge is for data processing techniques to meet timing constraints and at the same time ensure that security is maintained.

This proposal addresses information operations across infospheres. We first describe secure timely data sharing across infospheres and then focus on Role-based access control and Usage control in such an environment. Our goal is to send timely information to the war fighter while maintaining security. We will also address the application of game theory as well as decision centric data mining techniques to extract information from both trustworthy and untrustworthy partners of the coalition.

In particular, the **objectives** of this project are as follows:

- Develop a Framework for Secure and Timely Data Sharing across Infospheres.
- Investigate Access Control and Usage Control policies for Secure Data Sharing.
- Develop innovative techniques for extracting information from trustworthy and untrustworthy partners.

Technical Merit: While there has been work on data sharing across coalitions, an in-depth investigation of security issues as well as a study of the tradeoffs between security and timely processing has yet to be carried out. To our knowledge, this project is the first to investigate sophisticated security techniques such as Usage Control as well as decision centric data mining techniques for timely and secure data sharing across coalitions.

Broader Impact: The research to be carried out on this project is directly applicable to Network Centric Operations (NCO) that implement Network Centric Warfare (NCW). NCW promotes information sharing, shared situational awareness and knowledge of commander's intent. In addition it also enables war fighting advantage by providing synchronization, speed of command and increased combat power. We focus mainly on information sharing aspects of NCW. In particular, the results of this project can be transferred to the timely and secure data sharing services of the Network Centric Services activity being carried out by the Department of Defense.

Research Team: The research will be carried out both at the University of Texas at Dallas and at George Mason University. The principal investigators are among the leading researchers in Data and Applications Security. They have conducted innovative research in Secure Database Design, the Inference Problem, Role-based Access Control and Usage Control techniques as well as and carried out technology transfer activities. They are Fellows of IEEE, ACM, AAAS and the British Computer Society and have received prestigious awards for their research in Data and Applications Security.

ABSTRACT OF ANNUAL REPORT

The research reported in this annual report was carried out mainly at the University of Texas at Dallas (UTD) between December 1, 2007 and November 30, 2008. It describes the issues and challenges for information operations across infospheres and focuses on assured information sharing. We have examined three models: In the first model the partners of the coalition are considered to be trustworthy. In the second model, the partners are semi-trustworthy. In the third model the partners are untrustworthy.

The report essentially consists of three parts. We first provide an introduction to the project as well as the developments during Year 3. Impact of this work is also discussed. This introduction was also presented at the AFOSR review in June 2008. In the case of trustworthy models we conducted experiments on data sharing vs. data policy enforcement and developed a prototype system during Year 1 and Year 2. During Year 3 we have developed an approach for group-based information sharing (Part 1). For the semi-trustworthy model we enhanced the research carried out during Year 1 and Year 2. We examined the use of game theory for extracting information from the partners and demonstrated with bioterrorism applications in Year 3 (Part II). For the untrustworthy model, we examined the use of data mining for defensive operations during Year 1 and Year 2. We continued with this research and developed new tools (Part III.A). In addition, we also designed techniques to handle offensive (i.e. active defense) operations (Part III.B).

George Mason University (GMU) received a subcontract from the University of Texas at Dallas to examine the use of Role-based Access Control (RBAC) and Usage Control models for Coalition data sharing. The PI moved to UTSA and received the next phase of the funding to work on group-based information sharing (Part 1).

While this report discusses the developments during Year 3 of the project, it is also the final report for the project.

ACKNOWLEDGEMENT

Much of the research discussed in this annual report was supported buy the Air Force office of Scientific Office under Contract FA9550-06-1-0045. We thank Dr. Robert Herklotz of AFOSR for funding his encouragement and motivation. This research as also partially supported by the Erik Jonsson School of Engineering and Computer Science at the University of Texas at Dallas under the Texas Enterprise Funds. We thank Prof. Mark Spong (Dean) for this support.

Table of Contents

INTRODUCTION

1. Problem and Objectives

There is a critical need for organizations to share data within and across infospheres and form coalitions so that analysts could examine the data, mine the data, and make effective decisions. Each organization could share information within its infosphere. An infosphere may consist of the data, applications and services that are needed for its operation. Organizations may share data with one another across what is called a global infosphere that spans multiple infospheres. It is critical that the war fighters get timely information. Furthermore, secure data and information sharing is an important requirement. The challenge is for data processing techniques to meet timing constraints and at the same time ensure that security is maintained.

This proposal addresses information operations across infospheres. We first describe secure timely data sharing across infospheres and then focus on Role-based access control and Usage control in such an environment. Our goal is to send timely information to the war fighter while maintaining security. We will also address the application of game theory as well as decision centric data mining techniques to extract information from both trustworthy and untrustworthy partners of the coalition.

In particular, the **objectives** of this project are as follows:

- Develop a Framework for Secure and Timely Data Sharing across Infospheres.
- Investigate Access Control and Usage Control policies for Secure Data Sharing.
- Develop innovative techniques for extracting information from trustworthy and untrustworthy partners.

2. Our Approach

We developed three classes of solutions to handle the different types of partners. For trustworthy partners we developed solutions for policy based information sharing. For semi-trustworthy partners we applied game theoretic techniques to extract as much information as possible from the partners. For untrustworthy partners we protected our systems by providing solutions to malicious code detection. In addition, we also designed an approach to carry out active defensive (i.e. offensive) operations.

3. Our Contributions

This is the final report for the project. The contents of this report describe our research during Year 3. In this section we discuss our solutions for handling all three types of partners and mention the contributions during each year.

To handle trustworthy partners we first determined the amount of information that is lost when policies are enforced (report 1). We introduced the notion of release factor and showed that the amount of information lost decreases with release factor. We also conducted simulation experiments for policy based information sharing (report 1). Next we developed a proof of concept prototype that demonstrated policy based information sharing. We utilized medical databases and applications (report 2). In addition we also developed an approach to assign trusts levels to the partners in a peer to peer information sharing environment (report 2). Finally our subcontractor (initially at GMU and then at UTSA) has developed the idea of group-based information sharing where documents as well as members join and leave the coalition. The usage control model was extended to a group-based environment (report 3 – that is, this report).

To handle semi-trustworthy partners, we explored game theoretic techniques. Our goal is to extract as much information as possible from out partners and not divulge any of our information. During the first two years we conducted simulation studies using various types of games (report 1 and report 2). During Year 3 we applied the techniques to a bioterrorism attack (report 3). The goal of our work was to apply proven human-oriented situation analysis with existing simulation techniques to explore new ways of enhancing security through anticipation of human thought processes and activity. In particular, we used social networking to model relationships and game theory to model motivations of those participating. The end result of these studies yielded a combination of methods to anticipate, plan for, and reduce the impact of a biological attack. The SIR model was modified and applied to an individual-level social network through the use of theatres and approximated influences due to relationships, creating a unique, high performance mathematical model to observe the spread of disease. This model was then simulated in a number of scenarios spanning the use of possible attack situations, inoculations, and several novel intervention methods. The results of the simulation were then analyzed as a Stackelberg game in order to search for a lower bound to the expected costs and loss of human life under the assumption that the attacker goes last.

To handle untrustworthy partners we explored mainly defensive operations. Here we applied data mining techniques for malicious code detection (report 1 and report 2). During Year 3 we continued to apply data mining techniques for botnet detection (report 3). In addition we also designed techniques for offensive operations where the viruses that we develop will change patterns when new patches are introduced (report 3). We propose to enhance the design in our follow-on proposal that will focus on offensive operations.

3. Significant Outcomes

The total budget for this project is 300K from AFOSR and 150K in matching funds from the State of Texas. (i) One significant outcome of our research is the one pager we submitted to AFOSR on assured information sharing. This one pager was released as a MURI BAA in 2007 and subsequently AFOSR has made two multimillion dollar awards. (ii) We have also made presentations of our results to various air force bases through AFCEA including Edwards AFB and Kirkland AFB in 2006. (iii) We have also presented our research to other agencies and now have contracts and grants with NGA for geospatial semantic web and with IARPA to solve challenging problems in semantic web. (iv) We have published several papers in high quality journals and conferences and

have given keynote addresses including at the IEEE Intelligence and Security Informatics Conference in 2008. (iv) Finally this research has developed a new area in data and applications security and that is on inventive based information sharing. We will further develop these ideas under our MURI project.

4. Impact on Theses and Education

This project had a major impact on MS/PhD degrees and courses. The research has provided support for the PhD Thesis research of Ryan Layfield who has successfully defended his thesis (graduation December 2008) on applying game theory for information sharing (report 3). Ryan started his PhD work in the Spring of 2005 just before this project began. In addition it has also supported the PhD Thesis of Mehedy Masud on Data Mining for Cyber Security Applications (to graduate in 2009). Nathalie Tsybulnik was partially supported by this project for her PhD to develop techniques for assigning trust levels in a peer to peer environment. A PhD student at GMU/UTSA was also supported by this project on group-based information sharing. Several MS students have contributed to the programming projects. In particular Yashaswini Harshakumar developed the prototype for policy based information sharing Dilsad Cavus worked on examining the amount of information that is lost by enforcing policies. Srinivas developed simulation experiments on assured information sharing.

Mamoun Awad, a post-doctoral researcher was particularly supported by the project to supervise the experiments carried out during the first year of the project. The professors who have advised the students are: Bhavani Thuraisingham, Latifur Khan, Murat Kantarcioglu and Kevin Hamlen at UTD and Ravi Sandhu at UTSA.

In addition to incorporating units on information sharing to courses taught at AFCEA, we are also introducing a new graduate level course on data mining for cyber security applications. We have incorporated several units based on this research to our course in data and applications security.

5. Organization of this Report.

This report describes mainly the research that was carried out during year 3 of the project. The first paper is on group-based information sharing. The second paper is on applying game theory for information sharing with bioterrorism as an application. The third paper is a collection of papers that describe our research on applying data mining techniques for defensive operations. The fourth paper is our approach for handling offensive operations.

Part I

Handling Trustworthny Partners: Policy-based Information Sharing

Group-Centric Secure Information Sharing

Prof. Ravi Sandhu
Executive Director
Institute for Cyber Security
University of Texas at San Antonio, TX
ravi.sandhu@utsa.edu

Final report on AFSOR project FA 9550-06-01-0045
Subcontract through University of Texas at Dallas

November 20, 2008

1 Introduction and Background

Sharing information *while* protecting it is one of the earliest problems to be recognized in computer security, and yet remains a challenging problem to solve. Classic access control models are either inherently weak or do not even address this problem domain. The Discretionary Access Control model or DAC as discussed in [7, 9, 4] is fundamentally limited in that they control access only to original objects but not to copies. If objects could be read, one can read and create a copy and own this object. Mandatory Access Control models or MAC (like Bell-Lapadula) as discussed in [6] address information flow but are too rigid for fine-grained access control and falling back to DAC for fine-grained access control as the Orange Book suggests [4] is pointless.

We introduce and formalize the concept of Group-Centric Secure Information Sharing (g-SIS). Designers of security systems have traditionally made a distinction between security policy and mechanism. The general goal has been to build flexible and robust mechanisms that can conveniently support a wide range of policies. We follow the Policy, Enforcement, Implementation or PEI framework [12] (see figure 1) in distinguishing three separate but related models for the g-SIS problem: Policy Models which focus on subjects, objects and authorization policies for access, Enforcement Models which additionally accommodate trusted servers and authorities and thereby facilitate specification of protocols amongst these entities and Implementation Models which realize the enforcement model with specific software, system and cryptographic modules and algorithms. Note that a single Policy Model may have one or more Enforcement Models each of which may have one or more Implementation Models that realize the same set of objectives.

Such a layered approach allows us to focus on questions that are relevant to the respective layer and at a right level of abstraction. For example, at the Policy layer, we are primarily concerned with a subject's privileges on an object and not on Access Control Lists (ACLs). It is more appropriate to push this question of whether to use ACLs or Capability Lists to realize the policy to the Enforcement Layer. This enables us to study important aspects of the policy model without getting side-tracked by enforcement and implementation level issues. Another important aspect of the PEI framework is that the designer needs to always make a trade-off when transitioning from one layer to another. For instance, is a distributed setting, we suggest that an Enforcement Model is always an approximation of the Policy Model. Consider a distributed system with a policy that a subject can never access an object after 5PM. We call this an "ideal" policy because it assumes that there is no time delay in propagation of authorization information (the current time in this example). In the Enforcement Layer, as a practical matter, the designer needs to make a trade-off and approximate this "ideal" policy. One such approximation could be that in the given enforcement model, the subject can never access an object after 5PM with an accuracy of $\pm$ 5ms (thereby accounting for network delay).

The traditional approach to information sharing, characterized as Dissemination-Centric Sharing in this report, focuses on attaching attributes and policies to an object as it is disseminated from producers to consumers in a system. These policies are sometimes described as being "sticky". As an object is disseminated further down a supply chain the policies may get modified, such modification itself being controlled by existing policies. This mode of information

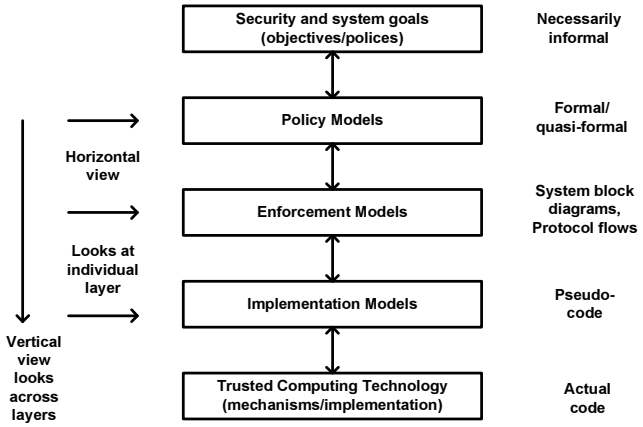

Figure 1: The PEI Models Framework.

sharing goes back to early discussions on originator-control systems [8, 10, 5, 11] in the 1980's and Digital Rights Management in the 1990's and 2000's. XrML [1], ODRL [3] and XACML [2] are recent examples of policy languages developed for this purpose. Dissemination-Centric Sharing describes in advance the characteristics or properties of subjects who may access the object by attaching "sticky policies" to be enforced when a subject attempts to access the object.

The vision of Group-Centric Sharing differs in that it advocates bringing the subjects and objects together to facilitate sharing. The metaphor is that of a secure meeting room where participants and information come together to enable the participants to "share" information for some common purpose. This common purpose can range from collaboration on a specific goal-oriented task (such as designing a new product) to participation in a shared activity (such as a semester long class) to subscription to a magazine (where the publisher contributes information that the participants read and possibly respond to content in associated blogs and forums). Visualize a conversation room where users may join, leave and re-join but only hear the conversation occurring during their participation period. For instance, in a Program Committee meeting Alice may be excused from the room when her paper is being discussed and may re-join the room after that portion of the discussion has concluded. In doing so, the conversation that occurred during her absence is not accessible to her. In another setting, all conversations are recorded on a whiteboard in the room and as Alice re-joins she is able to see what happened during her absence. Such a room may also be appropriate in a different context such as a design group wherein Alice participates as a consultant on demand. Further, Alice may view all the contents on the whiteboard while participating but may not be allowed to access the contents after she leaves the room. In another scenario, when Alice leaves, she may be allowed to retain access to the discussions in which she participated. We envision that Dissemination-Centric and Group-Centric Sharing will co-exist in a mutually supportive manner. For example, objects could be Added with "sticky" policies in a Group-Centric model. In this case, the objects may have controls imposed by both the group-centric model and the "sticky policies".

2 Overview

In this report, we confine our attention to Policy and Enforcement Models for g-SIS. Specifically, at the Policy Layer, we develop the foundations for a theory of Group-Centric Information Sharing, characterize a specific sub-family of models in this arena and identify several directions in which this theory can be extended. We propose an abstract set of group operations: Join and Leave for subjects, Add and Remove for objects. Subjects may Join, Leave and re-Join the group. Similarly, objects may be Added, Removed and re-Added to the group. An authorization policy is specified based on the relative membership period of the subject and object in question.

Clearly there has to be some control and authorization of each of these group operations. Rules concerning who can Join a group and who can authorize the Join are critical to security of the information sharing achieved via the group, and likewise for the other operations. Such issues are typically addressed as administrative tasks expressed in an administrative model. We leave the development of such an administrative model for future work. In this report, we focus solely on the operational aspects that bear on group membership. Furthermore, we confine our attention to correct

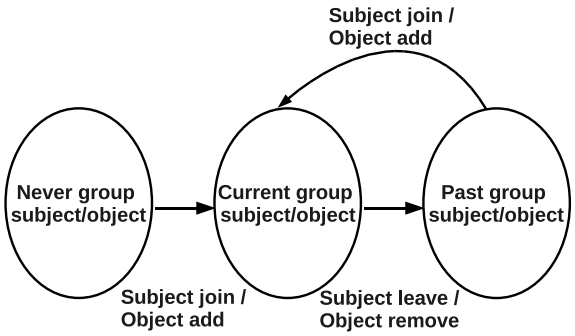

Figure 2: Subject and object membership states.

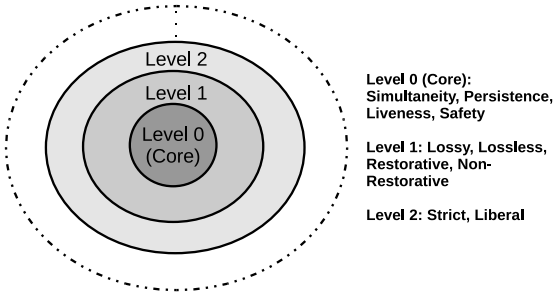

Figure 3: Layered g-SIS Properties.

authorization behavior with respect to "Read" access. The techniques can be extended easily to other forms of accesses as needed.

At the Enforcement Layer, we develop a flexible architecture that can realize any type of g-SIS policy. The components of this architecture such as the authorization information, decision and enforcement points are physically distributed across various computer systems. As noted earlier, in such a distributed setting, it is inevitable that authorization decisions will be based on authorization information that are stale or time-delayed. In a theoretical sense, some staleness is inherent in the intrinsic limit of network latencies. In a practical sense, authorization information is typically cached at the authorization decision point and refreshed periodically with the authorization information point. For the purpose of this report, we assume that attributes are the only carriers of authorization information and use these two terms interchangeably.

Given that the use of stale attributes is inevitable, the question is how do we safely use stale attributes for access control decisions and enforcement? Our central contribution at the enforcement layer is to formalize this notion of "safe use of a stale property" in the specific context of g-SIS. We also demonstrate specifications of systems that do and do not satisfy this requirement. We believe that the requirements for "safe use of a stale property" identified in this layer represent fundamental security properties the need for which arises in virtually any secure distributed systems in which the management and representation of authorization state is not centralized. We now present a high-level overview of g-SIS policy models in section 3 and the enforcement models and stale-safe properties in section 4.

3 Policy Model for g-SIS

Subjects and objects in a group go through various states as shown in figure 2. Different access policies are possible depending on the relative state of subjects and objects. For example, a joining subject could be allowed access only to new objects or also to objects that currently exist in the group. Similarly, a past subject may lose access to all objects or retain access to objects authorized during his membership period. When a subject rejoins the group, he may either gain access to objects authorized during his past membership or simply join the group as a new subject. Similarly, many different object policies are possible. Each group may thus pick a specific set of group-level access policies for subjects and objects.

3.1 Layered g-SIS properties

A g-SIS policy model is a Finite State Machine (FSM) that responds to sequences of group events (Join, Leave, Add and Remove) and determines at each state what read requests would be authorized should they occur. Each of these group operations (Join, Leave, Add and Remove) could be of various types—characterized as Lossy/Lossless, Restorative/Non-Restorative, Strict/Liberal, etc. We take a layered approach in developing the g-SIS policy models. First we have Core or Level 0 properties that any g-SIS model should satisfy regardless of the additional properties that it may support at Levels 1 or 2. A model is admitted as g-SIS only if it satisfies *all* of the core properties. Next, we have Level 1 properties that are based on specific variations of group operations. At level 1, we propose that group operation variations be fixed for all subjects in a given g-SIS model. That is, a g-SIS model should pick a specific subset of Level

1 properties that it wants to support depending on the requirements posed by the application. However, this subset may vary from one model to another. Finally, Level 2 properties are based on specific variations of Level 1 group operations. Unlike Level 1 properties, a g-SIS model is free to dynamically choose any variation of Level 2 group operations. We informally discuss these three layers below.

3.2 Core or Level 0 Properties

A g-SIS policy model should satisfy *all* of the following core properties.

- **Overlapping Membership (Simultaneity) Property:** This property states that a subject may access an object only if both the subject and object were simultaneously members of the group at some point in time.
- **Persistence Properties:** These properties state that authorization value (True, False) can change only when a group event such as Join, Leave, Add or Remove occurs (for the subject and object in question).
- **Liveness Property:** When a subject joins a group any object that is added to the group after Join time must be authorized unless the subject or object exits the group. This is a Liveness Property because it specifies the conditions under which authorization *must* succeed.
- **Safety Properties:** The safety properties specify the conditions under which a subject *cannot* access an object. They specify conditions in which authorization *must* be denied. For example, if an object is added to the group for the first time after the subject leaves, the subject will not be authorized unless he/she re-joins the group. Similarly, the set of all objects that a subject can access during non-membership period is bounded at Leave time. This set cannot grow until the subject rejoins the group.

3.3 Level 1 Properties

A g-SIS policy model should fix a specific set of group operations that it wants to support from the following list. For example, for every subject in the group, the g-SIS model may support Lossless Join, Gainless Leave, Non-Restorative Join and Leave.

- **Lossless Vs Lossy Join:** In a Lossless Join operation, when joining a group the subject never loses his/her authorizations granted prior to Join (non-membership period). In Lossy Join, the subject may be required to lose some prior authorizations at Join time. This is useful in specifying separation of duty kinds of policies.
- **Non-Restorative Vs Restorative Join:** These operations apply in the context of subject rejoin. In Non-Restorative Join, at Join time, authorizations granted during past membership period(s) may not be restored. In Restorative Join, at Join time, past authorizations are restored. Restorative Join is useful when an incentive is to be given to Join a group such as in subscription models.
- **Gainless Vs Gainful Leave:** In Gainless Leave, after leaving the group, the subject cannot access any object that was not authorized during the most recent membership period. In Gainful Leave, the subject gain the authorizations granted during past membership period(s).
- **Non-Restorative Vs Restorative Leave:** In Non-Restorative Leave, authorizations prior to joining the group may not be restored. In Restorative Leave, authorizations prior to joining the group are restored.

3.4 Level 2 Properties

A g-SIS model is allowed to use any of the following operations for subjects and objects. In general, Strict operations are more restrictive than their Liberal counter-parts.

- **Strict Vs Liberal** Join**:** On Strict Join (SJ), the subject may only access objects added after Join time. In Liberal Join (LJ), the joining subject, in addition, may access objects added prior to Join time.
- **Strict Vs Liberal** Leave**:** On Strict Leave (SL), the subject loses access to objects authorized during membership period. In Liberal Leave (LL), the leaving subject may retain authorizations granted during membership period.

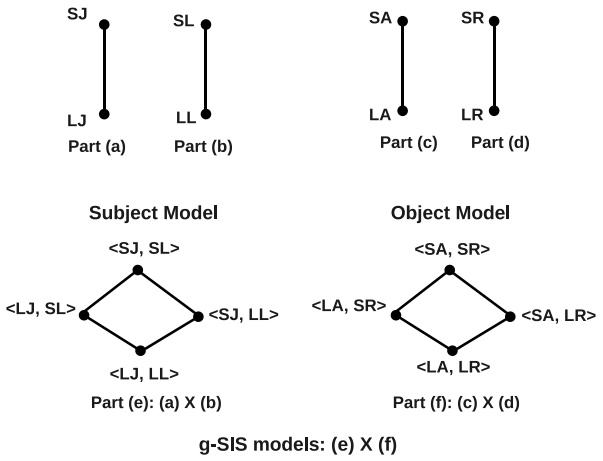

Figure 4: A family of g-SIS models: *The Cartesian product of Subject and Object Model results in a lattice of 16 g-SIS models with fixed operation types (products are ordered pointwise).*

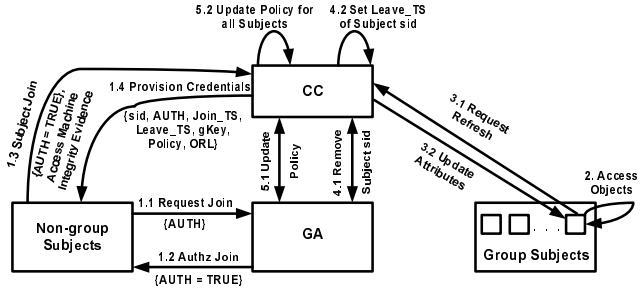

Figure 5: g-SIS Architecture.

- **Strict Vs Liberal** Add**:** On Strict Add (SA), the added object may only be accessed by existing group subjects at Add time. In Liberal Add (LA), the object may be accessed both by exiting and future subjects.
- **Strict Vs Liberal** Remove**:** On Strict Remove (SR), the removed object cannot be accessed by any group subject. On Liberal Remove (LR), subjects who had access at Remove time may continue to access.

Given that Level 1 properties are fixed, a g-SIS model allows four group operations: (Join, Leave, Add, Remove). If the type of Level 2 operations are also fixed for all subjects and objects (i.e. a specific type of operation is applied for all group subjects and objects), there are 16 possible models ranging from the most restrictive model allowing only Strict operations: (SJ, SL, SA, SR) to the most permissive model allowing only Liberal operations: (LJ, LL, LA, LR). This is illustrated in figure 4. Parts (a) through (d) show that the Strict operation is more restrictive than the Liberal operation. Parts (e) and (f) show the subject and object model that is obtained by the Cartesian product of subject and object operations respectively. Finally, a lattice of 16 g-SIS models can be obtained by a Cartesian product of subject and object models (parts (e) and (f)). An authorization policy exists for each of these 16 models that specify the conditions under which a subject may access an object. On the other hand, a highly flexible g-SIS model could simply allow different types of operations on a case by case basis. For example, SJ for s1, LJ when s1 re-joins, LJ for s2, LL for s1, SL for s2, SJ when s2 re-joins, etc. (similarly for objects). In this case, we have one all encompassing authorization policy that specifies the conditions under which a subject may access an object.

4 g-SIS Enforcement Model

Figure 5 shows a generic g-SIS architecture. A Group Administrator (GA) controls group membership for subjects and objects. A Control Center (CC) acts as a server that is responsible for maintaining subject and object attributes such as

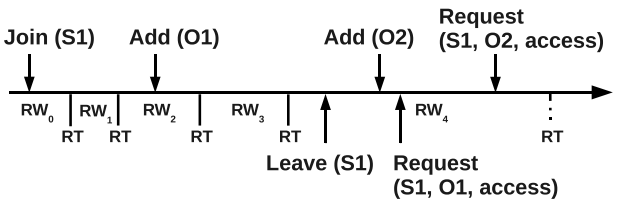

Figure 6: Events on a time line illustrating staleness leading to access violation.

$\mathrm{Join\_TS}$, $\mathrm{Leave\_TS}$, $\mathrm{Add\_TS}$ and $\mathrm{Remove\_TS}$ which represent the corresponding event time. Thus the CC acts as a Authorization Information Point in g-SIS. We assume the presence of a Trusted Reference Monitor (TRM) on the access machines which the subjects use to access objects. That is the TRM acts as the Authorization Decision Point. Thus all interactions in figure 5 are mediated by the TRM in the subject's access machine. We also assume that objects need not be always obtained from the server every time the subject needs to access. They are encrypted with a group key (gKey) and can be downloaded and stored locally either from the server or from other subjects. The resident TRM will decrypt only if the subject is allowed to access the object as per the group policy.

In steps 1.1-1.2, a non-group subject gets authorization from the GA to join the group. In steps 1.3-1.4, this subject gets the group attributes from the CC. Thereafter, the TRM (step 2) can make access decisions based on the local copy of subject and object attributes. The TRM periodically refreshes the local copy of attributes with the CC (steps 3.1-3.2). This refresh could be triggered based on a timeout or other mechanisms such as a usage count on the number of times objects may be accessed. In the mean time, the GA may update subject and object attributes at the CC (steps 4.1-4.2). For example, the subject may be removed from the group by setting his/her $\mathrm{Leave\_TS}$. This updated attribute value is copied to the TRM at refresh time (during steps 3.1-3.2). Similarly, the group policy may also be updated by the GA (steps 5.1-5.2). In summary, the g-SIS system may be characterized as below:

Subject attributes	$\{\mathrm{id}, \mathrm{Join\_TS}, \mathrm{Leave\_TS}, \mathrm{ORL}, \mathrm{gKey}\}$
Object attributes	$\{\mathrm{id}, \mathrm{Add\_TS}\}$.
Refresh Time (RT)	TRM contacts CC to refresh subject attributes and ORL.
Access Policy	$\mathrm{Authz}(S, O, R) \rightarrow O \notin \mathrm{ORL} \wedge \mathrm{Leave\_TS}(S) = \mathrm{NULL} \wedge \mathrm{Join\_TS}(S) \leq \mathrm{Add\_TS}(O)$.

Note that the $\mathrm{Add\_TS}$ attribute is part of the object itself. But since multiple copies of the same object may be scattered across various systems, we use an Object Revocation List (ORL) to Remove an object. The ORL lists the object id and the corresponding $\mathrm{Add\_TS}$ and $\mathrm{Remove\_TS}$ of the object to be removed thereby indicating the time at which the object is removed. As mentioned earlier, an authorization policy in g-SIS is based on relative membership period of the subject and object. For our discussion, we will use Authz specified above which states that a subject can access an object if the object was added after the subject joined the group and both the subject and object are still current members. Note that this is the same as the (SJ, SL, $\mathrm{LA/SA}$, SR) model in figure 4 in the policy layer.

4.1 Stale-Safety

When discussing the PEI framework we pointed out that the enforcement model is always an approximation of the policy model. Let us discuss one approximation that needs to be made in g-SIS. At the policy layer, we assumed instant revocation. However, in the Enforcement Model in figure 5, it takes a refresh to update this revocation information at the TRM. Thus until a timeout occurs at the TRM (at which point attribute values are synchronized with the CC), the subject may continue to access the object even though it may not be authorized as per the up-to-date authorization information available at the CC. In a way, we can assume that access decision at the TRM is always based on stale attributes. While it may not be practical to eliminate staleness due to inherent network latencies and more importantly due to limited refresh cycles in large distributed systems, it is possible to limit the access violations due to staleness of authorization information.

Figure 6 shows one sample trace in g-SIS that leads to access violation due to attribute staleness. Subject S1 joins the group and the attributes are refreshed with the CC periodically. RT represents the time at which refreshes happen. The time period between any two RT's is a Refresh Window, denoted RW_i. After join, RW_0 is the first window, RW_1 is the next and so on. Suppose RW_4 is the current Refresh Window. Objects O1 and O2 were added to the group by *some* group subject (or the GA) during RW_2 and RW_4 respectively and they are available to S1 via super-distribution.

In RW_4, S1 requests access to O1 and O2. An access decision will be made by the TRM in the access machine as per the attributes obtained at the latest RT.

Clearly, our access policy will allow access to both O1 and O2. However it is possible that S1 was removed by the GA right after the last RT and before Request(S1, O1, access) in RW_4. Ideally, S1 should not be allowed to access both O1 and O2.

From a confidentiality perspective in information sharing, granting S1 access to O1 is relatively less of a problem than granting access to O2. This is because the CC or the GA can assume that S1 was always authorized access to O1 and hence information has already been released to S1. In the worst case, S1 continues to access the same information (O1) until the next RT. However, S1 never had an authorization to access O2 and letting S1 access O2 means that S1 has gained knowledge of new information. This is a critical violation and should not be allowed. Such scenarios are what our stale-safe security properties address. A subject cannot access an object if it was added to the group after the last refresh time even if the authorization policy allows access.

Informally, we have two versions of stale-safe security properties that a g-SIS enforcement model should satisfy—Weak and Strong. The Weak Stale-Safe Security Property allows safe access decisions to be made without having to contact the CC when a request is received. This property requires that the subject may access any object during a refresh window if it was authorized at the time of refresh. Thus in figure 6, as per the attributes available at the most recent RT, the subject would not be granted to access O2. The Strong Stale-Safe Security Property prohibits access without refreshing attribute values with the CC. Thus when a request is received, the TRM is required to synchronize attribute values with the CC before authorization decision is made. Note that it is still possible for attribute values to change right after synchronization but the property is only intended to minimize the risk.

We can make the TRM satisfy the Weak Stale-Safe Security Property by simply maintaining a refresh time-stamp indicating the most recent refresh time and ensuring that the $\mathrm{Add\_TS}$ of the requested object is no greater than the refresh time-stamp. This way we can ensure that the object was authorized at the refresh time. The TRM can be modeled as a Finite State Machine (FSM) and Stale-Safe Security properties can be verified against the machine using well-known model checking techniques.

A rigorous formal specification of the g-SIS policy model (using Linear Temporal Logic) and the verification of the stale-safe security properties against the g-SIS enforcement model using model checking can be found in the appendices.

References

[1] eXtensible rights Markup Language. *www.xrml.org*.

[2] OASIS eXtensible Access Control Markup Language . *www.oasis-open.org/committees/xacml/*.

[3] The Open Digital Rights Language Initiative. *www.odrl.net*.

[4] DOD Trusted Computer Systems Evaluation Criteria. Dec 1985.

[5] M. Abrams, J. Heaney, O. King, L. LaPadula, M. Lazear, and I. Olson. Generalized Framework for Access Control: Towards Prototyping the ORGCON Policy. *Proceedings of the 14th National Computer Security Conference*, pages 257–266, 1991.

[6] D. Bell and L. LaPadula. Secure computer systems: Unified exposition and multics interpretation. *Technical Report, The Mitre Corp.*, March 1975.

[7] G. Graham and P. Denning. Protection principles and practice. *AFIPS Joint Computer Conference*, 1972.

[8] R. Graubart. On the Need for a Third Form of Access Control. *Proceedings of the 12th National Computer Security Conference*, pages 296–304, 1989.

[9] B. Lampson. Protection. *In fifth Princeton Symposium on Information Science and Systems*, 40:437443, 1971.

[10] C. McCollum, J. Messing, and L. Notargiacomo. Beyond the pale of MAC and DAC - defining new forms of access control. *Proceedings of the 1990 IEEE Symposium on Security and Privacy*, pages 190–200, 1990.

[11] J. Park and R. Sandhu. Originator control in usage control. *Policies for Distributed Systems and Networks, 2002. Proceedings. Third International Workshop on*, pages 60–66, 2002.

[12] R. Sandhu, K. Ranganathan, and X. Zhang. Secure information sharing enabled by trusted computing and pei models. *Proc. of ASIACCS 2006*, pages 2–12, 2006.

Appendix A

Group-Centric Secure Information Sharing Models

ABSTRACT

In this paper, we develop the foundations for a theory of Group-Centric Secure Information Sharing (g-SIS), characterize a specific sub-family of models in this arena and identify several directions in which this theory can be extended. The traditional approach to information sharing, characterized as Dissemination-Centric Sharing in this paper, focuses on attaching attributes and policies to an object (sometimes called "sticky policies") as it is disseminated from producers to consumers in a system. In contrast, Group-Centric Sharing envisions bringing the subjects and objects together in a group to facilitate sharing. The metaphor is that of a secure meeting room where participants and information come together to enable parties to "share" information for some common purpose.

We propose an abstract set of group operations: Join and Leave for subjects, Add and Remove for objects. Each of these operations could be of various flavors characterized in the paper as Lossy, Lossless, Restorative, Non-Restorative, Strict and Liberal. For example, in Lossless Join, subjects do not lose existing authorizations by joining a group. In Lossy Join, a subject loses some or all of existing authorizations. We formalize the concept of an Information-Sharing Group using Linear Temporal Logic (LTL), by specifying three layers of properties. We begin with a core set of properties called Level 0 (Simultaneity, Persistence, Liveness, Safety, etc.) that any g-SIS model must satisfy. Next at Levels 1 and 2 we identify additional properties regarding specific variations of group operations (Lossy, Lossless, Strict, Liberal, etc.). Finally, we specify the correct authorization behavior for a sub-family of g-SIS models using LTL and formally prove using model checking that the models satisfy the properties.

1. INTRODUCTION

This paper introduces and formalizes the concept of Group-Centric Secure Information Sharing (g-SIS). The traditional approach to information sharing, characterized as Dissemination-Centric Sharing in this paper, focuses on attaching attributes and policies to an object as it is disseminated from producers to consumers in a system. These policies are sometimes described as being "sticky". As an object is disseminated further down a supply chain the policies may get modified, such modification itself being controlled by existing policies. This mode of information sharing goes back to early discussions on originator-control systems [18, 23, 5, 26] in the 1980's and Digital Rights Management in the 1990's and 2000's. XrML [1], ODRL [4] and XACML [2] are recent examples of policy languages developed for this purpose. Dissemination-Centric Sharing describes in advance the characteristics or properties of subjects who may access the object by attaching "sticky policies" to be enforced when a subject attempts to access the object.

The vision of Group-Centric Sharing differs in that it advocates bringing the subjects and objects together to facilitate sharing. The metaphor is that of a secure meeting room where participants and information come together to enable the participants to "share" information for some common purpose. This common purpose can range from collaboration on a specific goal-oriented task (such as designing a new product) to participation in a shared activity (such as a semester long class) to subscription to a magazine (where the publisher contributes information that the participants read and possibly respond to content in associated blogs and forums). Visualize a conversation room where users may join, leave and re-join but only hear the conversation occurring during their participation period. For instance, in a Program Committee meeting Alice may be excused from the room when her paper is being discussed and may re-join the room after that portion of the discussion has concluded. In doing so, the conversation that occurred during her absence is not accessible to her. In another setting, all conversations are recorded on a whiteboard in the room and as Alice rejoins she is able to see what happened during her absence. Such a room may also be appropriate in a different context such as a design group wherein Alice participates as a consultant on demand. Further, Alice may view all the contents on the whiteboard while participating but may not be allowed to access the contents after she leaves the room. In another scenario, when Alice leaves, she may be allowed to retain access to the discussions in which she participated.

We envision that Dissemination-Centric and Group-Centric Sharing will co-exist in a mutually supportive manner. For example, objects could be Added with "sticky" policies in a Group-Centric model. In this case, the objects may have controls imposed by both the group-centric model and the "sticky policies". Also, the "sticky policies" on the object could determine whether or not an object can be added to the group in the first place. It may turn out that at a theoretical level whatever Dissemination-Centric can achieve Group-Centric can also achieve and vice versa. But at a pragmatic level, we believe these are significantly different approaches to information sharing.

In this paper we develop the foundations for a theory of Group-Centric Information Sharing, characterize a specific sub-family of models in this arena and identify several directions in which this theory can be extended. We propose an abstract set of group operations: Join and Leave for subjects, Add and Remove for objects. Subjects may Join, Leave and re-Join the group. Similarly, objects may be Added, Removed and re-Added to the group. Further each of these operations could be of various flavors characterized in the paper as Lossy, Lossless, Restorative, Non-Restorative, Strict and Liberal. For example, in Lossless Join, a joining subject never loses access to objects authorized prior to joining the group. Similarly, in Restorative Join, the joining subject may regain access to objects authorized during past membership period. In general, there may be any number of such variations beyond those explicitly identified in this paper.

Clearly there has to be some control and authorization of each of these operations. Rules concerning who can Join a group and who can authorize the Join are critical to security of the information sharing achieved via the group, and likewise for the other operations. Such issues are typically

addressed as administrative tasks expressed in an administrative model. We leave the development of such an administrative model for future work. In this paper we focus solely on the operational aspects that bear on group membership. Furthermore, we confine our attention to correct authorization behavior with respect to "Read" access. The techniques can be extended easily to other forms of accesses as needed.

The principal contributions of this paper are as follows. We formalize the concept of Group-Centric Information Sharing (g-SIS) using Linear Temporal Logic (LTL), by specifying three layers of properties. We identify a core set of properties at level 0 (Simultaneity, Persistence, Liveness, Safety, etc.) that should be satisfied by any g-SIS model. Next at Levels 1 and 2 we specify additional properties regarding specific variations of group operations (Lossy, Lossless, Strict, Liberal etc.). We formally specify the authorization policies for a specific sub-family of g-SIS models using LTL. Finally, we discuss the formal verification of the core properties against the g-SIS models using the well-known technique of model checking.

The remainder of this paper is organized as follows. In section 2, we discuss related work in this area. We specify core properties (level 0) using LTL in section 3. In section 4, we specify additional level 1 and level 2 properties based on specific variations of group operations. In section 5, we specify the authorization policies for a sub-family of g-SIS models and prove that they satisfy the core properties. In section 6, we discuss future work and conclude in section 7.

2. RELATED WORK

Older approaches to Secure Information Sharing (SIS) can be classified into at least three categories. First is Discretionary Access Control (DAC) [17, 21, 16] which proposes to enforce controls on sharing information at the discretion of the "owner" of the object. Although, this is similar in objective to SIS, DAC fails to solve the problem since it does not correlate the controls on copies of information with copies of the original.

The second is Mandatory Access Control (MAC) [8, 15, 16] which allows information to flow in one direction in a lattice of security labels. Copies of information made from one or more objects inherit the least upper bound of the labels from the individual objects. Thereby the copies are controlled at least as strictly as the original. Historically, one directional information flow has not been the most common requirement of SIS. In particular, MAC does not allow the owner of the object to share information but also to protect it from other users in the same or higher security levels. MAC also suffers from covert channel issues whereby information flow contrary to the labels can occur via malware.

The third is Originator Control or ORCON [18, 23, 5, 26] in which the owner of the object decides which user(s) may have access to it. The owner is the principal source of the policy to be enforced. As information flows from one container to another, the policy is also propagated. In other words, it is a "sticky policy".

Recently, information sharing challenges have been considered in the context of Dynamic Coalition Problem or DCP (see [27, 9, 19, 20, 7, 32] for example). The DCP is concerned with the challenges involved when a coalition is dynamically formed, for example, in response to a crisis. Government, civilian and other commercial organizations may need to form a coalition (who may otherwise distrust each other) and share information quickly to solve the problem at hand. In [6], the authors use a role-based delegation framework for specifying policies for resource and information sharing within and across organizations. Finally, the Secure Information Sharing Architecture or SISA [3] is an alliance formed by major technology corporations that provide Commercial Off-The Shelf (COTS) architectural solutions to share information between various organizations. Our work largely differs from all these approaches in that we solely focus on the policy models for a group-centric SIS problem thereby separating enforcement and implementation issues from policy [31]. Further, formal specification of g-SIS properties using LTL enables us to rapidly automate verification against the models using well-known model checking techniques.

To the best of our knowledge, this is the first effort towards developing a formal model for group-centric SIS. At a policy level, the closest work to group-centric sharing that can be found in the literature is in the area of Secure Multicast [28]. It will be evident later that the g-SIS models subsume policies considered by Secure Multicast.

3. CORE G-SIS PROPERTIES (LEVEL 0)

In this section, we specify a core set of properties that should be satisfied by any g-SIS policy model. We formally specify the properties using Linear Temporal Logic [22].

3.1 Overview

A g-SIS policy model is a Finite State Machine (FSM) that responds to sequences of group events (Join, Leave, Add and Remove) and determines at each state what read requests would be authorized should they occur. As will be discussed in detail later, each group operation (Join, Leave, Add and Remove) could be of various types. For example, a Join operation could be Lossless or Lossy. In Lossless Join, the joining subject never loses access to objects authorized prior to Join time. In Lossy Join, the joining subject may lose access to some or all of the objects authorized prior to Join time. A Lossless Join is the most common SIS scenario. Revisiting our earlier secure meeting room scenario, when Alice momentarily steps out, suppose that she is allowed to retain access to all the conversations that happened in the room during her presence. When she re-joins the room, it is natural that she does not lose access to those past conversations. However, if Alice joins in a Lossy manner, she may be forced to relinquish access to past conversations at re-join time.

Similarly, a Join operation could be Restorative or Non-Restorative. A Restorative Join explicitly restores accesses authorized during past membership period while a Non-Restorative Join does not. Suppose Alice steps out and she is not allowed to access any conversation when she leaves the room. When Alice re-joins the conversations in the room, a Restorative Join explicitly restores her past accesses (irrespective of what exactly the current Join enables). But in a Non-Restorative Join, the Join operation does not explicitly restore her past access. If the nature of her current Join allows her to access past conversations, she may access them—not otherwise. Finally, the group operations could be Strict or Liberal. A Strict operation is more restrictive in terms of access relative to its Liberal counter part. For example, suppose Bob joins the room for the first time. A Strict Join would allow Bob to access only new conversa-

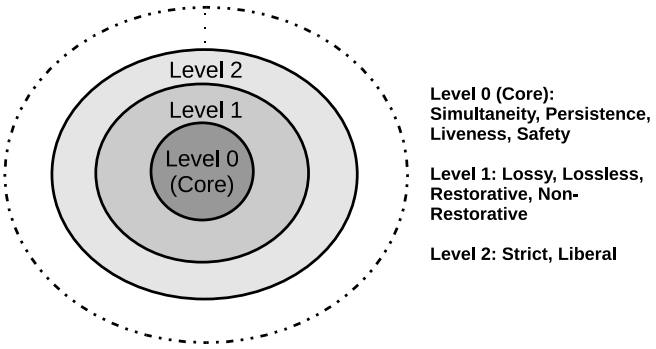

Figure 1: Layered g-SIS Properties.

tions. On the contrary, a Liberal Join, in addition, would allow Bob to access older conversations in the room that happened before he joined. In general, there may be any number of such variations of group operations.

Figure 1 shows a layered set of properties. At the center is the Core or Level 0 properties that must be satisfied by any g-SIS model. Next, we have Level 1 properties that must be satisfied (in addition to the core properties) by models that support specific variations of group operations (Lossy/Lossless, Gainless/Gainful, Restorative/Non-Restorative, etc.). Finally, we have Level 2 properties that are based on next level of variations of group operations (Strict/Liberal). In general, indefinite levels of such properties are possible (illustrated in the figure as a dotted circle) but we anticipate two or three levels would suffice in practice.

The properties we discuss in this section are Core or Level 0 properties that any g-SIS model should satisfy regardless of the additional properties that it may support at Levels 1 or 2. A model is admitted as g-SIS only if it satisfies *all* of the core properties. On the contrary, a g-SIS model need not satisfy every Level 1 or 2 property. These properties allow various flavors of g-SIS models since they are based on specific variations of group operations. At level 1, we propose that such variations be fixed for all subjects in a given g-SIS model. That is, a g-SIS model should pick a specific subset of Level 1 properties that it wants to support depending on the requirements posed by the application. However, this subset may vary from one model to another. For example, in a given g-SIS model, we pick Lossy or Lossless Join for all subjects. Level 2 properties are based on specific variations of Level 1 group operations. Unlike Level 1 properties, a g-SIS model is allowed to dynamically choose any variation of Level 2 group operartions. For example, subject s1 may be given Strict Join while s2 may be given Liberal Join. However Level 1 operation type is fixed for both s1 and s2.

3.2 Formal Specification of Core Properties

We now formally specify the core g-SIS properties. We use Linear Temporal Logic [22] to specify group properties that the g-SIS models should satisfy. Temporal logic is a specification language for expressing properties related to a sequence of states in terms of temporal logic operators and logic connectives (e.g., $\wedge$ and $\vee$). Temporal logic operators are of two types: Past and Future. A brief overview of temporal operators used in this paper is given in table 1 which can be used as a reference to interpret the upcoming formulas. A model will be admitted as g-SIS only if it satisfies *all* of the core properties. Thus, these properties should be satisfied by models that support any type of group operation.

Notations and Conventions.

We now discuss a few notations and conventions that will be used in specifying the formulas in this paper. We mentioned earlier that the group operations could be of many different types. For example, join_i, leave_j, add_k and remove_ℓ are a specific type of Join, Leave, Add and Remove respectively. In all of our formulas, Join, Leave, Add and Remove are meta-variables that denote a disjunction of those types as indicated below. We assume that any reference to Join, Leave, Add and Remove operations are interpreted with the following semantics:

$$\text{Join}(s) \equiv (\text{join}_1(s) \vee \text{join}_2(s) \vee ... \vee \text{join}_\text{m}(s))$$
$$\text{Leave}(s) \equiv (\text{leave}_1(s) \vee \text{leave}_2(s) \vee ... \vee \text{leave}_\text{n}(s))$$
$$\text{Add}(o) \equiv (\text{add}_1(o) \vee \text{add}_2(o) \vee ... \vee \text{add}_\text{p}(o))$$
$$\text{Remove}(o) \equiv (\text{remove}_1(o) \vee \text{remove}_2(o) \vee ... \vee \text{remove}_\text{q}(o))$$

We characterize the group operations as events in the LTL formulas. That is, Join is "True" only if any of the join_i events (for $1 \leq \text{i} \leq \text{m}$) occur in a temporal state. Similarly, Leave, Add and Remove are "True" only if any of the respective events occur.

Authz(s,o,r) is a predicate on states that when "True", indicates that the r operation is authorized in that state. That is, the "True" or "False" value of Authz depends on the previous trace of events. We concern our attention to authorizations involving read access and the properties we specify can be extended, if need be, to other types of accesses. Note that objects can be added to the group but Authz is only concerned about read access to those objects. In summary, we have the following:

Join (s)	Subject s joins the group. Represents occurrence of one of join_i events.
Leave (s)	s leaves the group. Represents occurrence of one of leave_j events.
Add (o)	Object o is added to group. Represents occurrence of one of add_k events.
Remove (o)	o is removed. Represents occurrence of one of remove_l events.
Authz (s,o,r)	s is authorized to exercise a right r on o.

We drop the parameters in all of the predicates above for convenience and clarity. Thus a formula Authz $\rightarrow$ (Join $\wedge$ ($\neg$(Leave$\vee$Remove) $\mathcal{S}$ Add)) actually means $\text{Authz}(s,o,r) \rightarrow (\text{Join}(s) \wedge (\neg(\text{Leave}(s) \vee \text{Remove}(o)) \; \mathcal{S} \; \text{Add}(o)))$. Note that Join and Leave refer to the same subject and Add and Remove refer to the same object. Further, Authz refers to the same specific pair of s and o.

A g-SIS policy model is an FSM as described below:

$$\mathcal{M} = \langle \text{E}, \text{Authz}, \sigma_0, \Sigma, \Delta \rangle$$
$$\text{E} = \{\text{Join} \cup \text{Leave} \cup \text{Add} \cup \text{Remove}\}$$
$$\text{Authz} : \Sigma \rightarrow \{\text{True}, \text{False}\}$$
$$\Delta : \Sigma \times 2^\text{E} \rightarrow \Sigma$$

A g-SIS model, $\mathcal{M}$, is a 4-tuple: E is a union non-empty sets of group events (i.e., each set of Join, Leave, Add and Remove is non-empty), Authz is the authorization predicate that is either True or False in each state, σ_0 is the initial

Table 1: Temporal Operators

Future/Past	Operator	Read as	Explanation
Future	$\bigcirc$	Next	($\bigcirc$ p) means that the formula p holds in the next state.
	$\Box$	Henceforth	($\Box$ p) means that the formula p will continuously hold in all future states starting from the current state.
	$\mathcal{U}$	Until	(p $\mathcal{U}$ q) means that q will occur sometime in the future and p will hold at least until the first occurrence of q.
	$\mathcal{W}$	Unless	(p $\mathcal{W}$ q) is a weaker form of (p $\mathcal{U}$ q). It says that p holds either until the next occurrence of q or if q never occurs, it holds throughout the sequence.
Past	$\ominus$	Previous	($\ominus$ p) means that formula p held in the previous state.
	$\blacklozenge$	Once	($\blacklozenge$ p) means that formula p held at least once in the past.
	$\mathcal{S}$	Since	(p $\mathcal{S}$ q) means that q happened in the past and p held continuously from the position following the last occurrence of q to the present.

state, Σ is the set of all states and Δ is the transition relation which specifies the transition that the FSM should take when an event occurs.

Well-Formed Traces.

The following g-SIS assumptions separate well-formed traces of group events from all possible traces. A g-SIS model, $\mathcal{M}$, ignores any trace of group events that does not satisfy the following formulas. Thus our properties apply only to well-formed traces.

A. Two events for the same subject cannot occur at the same time. Similarly, two events for the same object cannot occur at the same time. There are two cases:

A.1 An object cannot be Added and Removed and a subject cannot Join and Leave at the same time.

$$\Box(\neg(\text{Add} \wedge \text{Remove}) \wedge \neg(\text{Join} \wedge \text{Leave}))$$

A.2 For any given subject or object, two types of operation cannot occur at the same time [1].

$$\forall i, j\ \Box((i \neq j) \rightarrow \neg(\text{join}_\text{i} \wedge \text{join}_\text{j}))$$
$$\forall k, \ell\ \Box((k \neq \ell) \rightarrow \neg(\text{leave}_\text{k} \wedge \text{leave}_\ell))$$
$$\forall s, t\ \Box((s \neq t) \rightarrow \neg(\text{add}_\text{s} \wedge \text{add}_\text{t}))$$
$$\forall x, y\ \Box((x \neq y) \rightarrow \neg(\text{remove}_\text{s} \wedge \text{remove}_\text{y}))$$

B. If a subject s joins a group, s cannot join again unless s first leaves the group. That is, any two Join's should be separated by a Leave. A similar rule applies for other operations.

$$\Box(\text{Join} \rightarrow \bigcirc\ (\neg\text{Join}\ \mathcal{W}\ \text{Leave}))$$
$$\Box(\text{Leave} \rightarrow \bigcirc\ (\neg\text{Leave}\ \mathcal{W}\ \text{Join}))$$
$$\Box(\text{Add} \rightarrow \bigcirc\ (\neg\text{Add}\ \mathcal{W}\ \text{Remove}))$$
$$\Box(\text{Remove} \rightarrow \bigcirc\ (\neg\text{Remove}\ \mathcal{W}\ \text{Add}))$$

A g-SIS policy model should satisfy *all* of the following core properties.

1. Overlapping Membership (Simultaneity) Property: We begin with the simplest and most fundamental property. Clearly, a group subject can access a group object only if they were both members of the group at least once: $\Box(\text{Authz} \rightarrow (\blacklozenge\text{Add} \wedge \blacklozenge\text{Join}))$. This is too weak to be a core property since the subject and object could have been members at different time periods. We first strengthen this property by requiring that the subject and object were *simultaneously* members of the group at some point in the past. That is, the subject and object had an overlapping membership period as specified below:

$$\Box(\text{Authz} \rightarrow \blacklozenge(\text{Add} \wedge (\neg\text{Leave}\ \mathcal{S}\ \text{Join})) \vee \blacklozenge(\text{Join} \wedge (\neg\text{Remove}\ \mathcal{S}\ \text{Add})))$$

This formula can be interpreted as follows. In a g-SIS model if a subject is able to access an object then there exists a point in the past (indicated by the $\blacklozenge$ operator) where (a) the object was added and the subject was part of the group at that point in time ($\neg$ Leave Since Join) *or* (b) the subject joined the group and the object was part of the group at that point in time ($\neg$ Remove Since Add).

Next, we further strengthen this property by requiring that only membership can enable authorizations. That is, a subject is able to access an object only by joining the group. A Join operation can enable new access while a Leave may at most maintain the authorizations enabled by Join or disable some or all authorizations. We now state the Overlapping Membership Property as below:

$$\Box(\text{Authz} \rightarrow \blacklozenge(\text{Join} \wedge ((\neg\text{Leave} \wedge \neg\text{Remove})\ \mathcal{U}\ \text{Authz}) \wedge (\neg\text{Remove}\ \mathcal{S}\ \text{Add})) \vee \blacklozenge(\text{Add} \wedge ((\neg\text{Leave} \wedge \neg\text{Remove})\ \mathcal{U}\ \text{Authz}) \wedge (\neg\text{Leave}\ \mathcal{S}\ \text{Join})))$$

This property strengthens the earlier formula by requiring that Authz holds in a state only if it held at least once during an overlapping membership period [2].

2. Persistence Properties: These properties state that authorization value (True, False) can change only when a group event (for the subject and object in question)

[1] Note that LTL does not allow an event to occur more than once in the same state. For example, join_1 remains True if it occurs once or twice in the same state.

[2] Thus if a subject is able to access an object during non-membership period then the access should have been authorized during a membership period in the past. This can be stated as: $\Box((\text{Leave} \wedge (\neg\text{Join}\ \mathcal{U}\ \text{Authz})) \rightarrow \blacklozenge(\text{Join} \wedge (\neg\text{Leave}\ \mathcal{U}\ \text{Authz})))$.

occurs.

Authorization Persistence: When a subject s is authorized to access an object o, it remains so at least until a group event involving s or o occurs [3].

$$\Box(\text{Authz} \rightarrow (\text{Authz } \mathcal{W} \ (\text{Join} \vee \text{Leave} \vee \text{Add} \vee \text{Remove})))$$

Revocation Persistence: When a subject s is not authorized to access an object o, it remains so at least until a group event involving s or o occurs.

$$\Box(\neg\text{Authz} \rightarrow (\neg\text{Authz } \mathcal{W} \ (\text{Join} \vee \text{Leave} \vee \text{Add} \vee \text{Remove})))$$

3. **Liveness Property:** When a subject joins a group any object that is added to the group after Join time must be authorized unless the subject or object exits the group. This is a Liveness Property because it specifies the conditions under which authorization *must* succeed.

$$\alpha_1 \equiv (\text{Add} \rightarrow (\text{Authz } \mathcal{W} \ (\text{Remove} \vee \text{Leave})))$$
$$\alpha \equiv \Box(\text{Join} \rightarrow (\alpha_1 \ \mathcal{W} \ \text{Leave}))$$

Formula α above characterizes this property. It states that if a Join occurs then the implication α_1 should hold continuously unless a Leave occurs. Formula α_1 says that if an Add occurs, Authz should hold continuously unless a Remove or a Leave occurs. Effectively, α states that if a subject joins the group and subsequently an object is added (while the subject is still a member), the subject will be authorized to access the object unless and until the subject or the object exits the group.

Note that this property also implies when an object is added to the group, any subject who joined the group prior to object Add time can access the object unless the subject or object exits the group. Thus the above property can be equivalently stated as follows:

$$\Box((\text{Add} \wedge (\neg\text{Leave } \mathcal{S} \ \text{Join})) \rightarrow (\text{Authz } \mathcal{W} \ (\text{Leave} \vee \text{Remove})))$$

4. **Safety Properties:** These safety properties specify the conditions under which a subject *cannot* access an object. They specify conditions in which authorization *must* be denied.

Subject Safety: After a subject leaves a group, if an object is added for the first time, the past subject cannot access the object [4].

$$\Box((\text{Leave} \wedge (\neg\blacklozenge\text{Add})) \rightarrow((\neg\text{Authz } \mathcal{W} \ \text{Join}) \wedge (\neg\text{Authz } \mathcal{W} \ \text{Add})))$$

Object Safety: A removed object cannot be accessed by subjects joining the group for the very first time [4].

$$\Box((\text{Remove} \wedge (\neg\blacklozenge\text{Join})) \rightarrow((\neg\text{Authz } \mathcal{W} \ \text{Join}) \wedge (\neg\text{Authz } \mathcal{W} \ \text{Add})))$$

Bounded Subject Authorization: The set of all objects that a subject can access during non-membership period is bounded at Leave time. This set cannot grow until the subject rejoins the group.

$$\Box((\text{Leave} \wedge \neg\text{Authz}) \rightarrow (\neg\text{Authz } \mathcal{W} \ \text{Join}))$$

Bounded Object Authorization: This is a dual of the earlier property. The set of all subjects who can access a removed object is bounded at Remove time. This set cannot grow until the object is re-added to the group.

$$\Box((\text{Remove} \wedge \neg\text{Authz}) \rightarrow (\neg\text{Authz } \mathcal{W} \ \text{Add}))$$

4. G-SIS PROPERTIES (LEVELS 1 AND 2)

In this section, we specify additional properties based on the various flavors that a g-SIS model may support. We first specify the Level 1 properties which are concerned with the first level of variations—Lossy, Lossless, Restorative and Non-Restorative. Next, we specify Level 2 properties that are next level of variations: Strict and Liberal.

We first specify additional terminology that will be used in the rest of this paper. Figure 2 shows a sample trace of events on a timeline. Starting from left, subject s1 joins and leaves the group and re-joins again in the future. During this period, objects are added and removed from the group. Relative to s1's most recent join, we refer to objects o2 and o3 as *existing* group objects. Objects o4 and o5 are referred to as *new* objects (relative to s1) to be added in the future. Similarly, in Figure 3, when o2 is added, s1 is referred to as *existing* group subject relative to o2. s1 and s3 are referred to as *new* group subjects (relative to o2) who join the group after o2 is added.

Further, a subject is called a *current subject* if he/she joined the group sometime in the past and has not left the group since then. A subject is called *past subject* if he/she joined the group sometime in the past, subsequently left the group and has not re-joined since then. The terms *current object* and *past object* have similar semantics.

4.1 Level 1 Properties

4.1.1 Lossless Vs Lossy Join

The Join operation could be either Lossless or Lossy. In Lossless Join, the joining subject does not lose access to objects that were authorized prior to Join time. For example, in figure 2, suppose that when s1 leaves the group he/she is allowed to retain access to objects o1 and o2. From the Overlapping Membership Core property, o3 is not accessible

[3]Note that a subject may Join a group, Leave subsequently and retain access to some objects. It is possible that this subject may lose access to these objects by Joining the group again. We call this a Lossy Join where joining a group may actually require a subject to relinquish all prior access. In general, there are many flavors of group operations that will be discussed in detail in the following sections. This is the reason we include enabling operations such as Join and Add in the $\mathcal{W}$ (unless) predicate in the above formula.

[4]This property is entailed by the Overlapping Membership Property as demonstrated in appendix A. Nevertheless, we believe it is an important observation for safety and hence explicitly recognize it here as a core property.

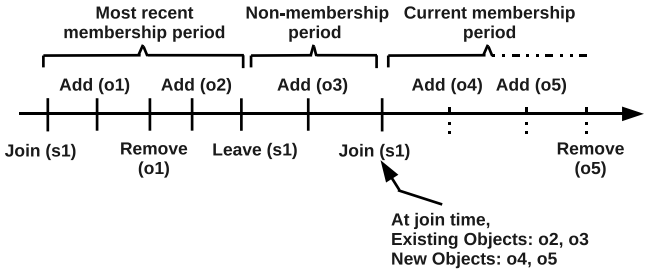

Figure 2: Subject Operations Illustration.

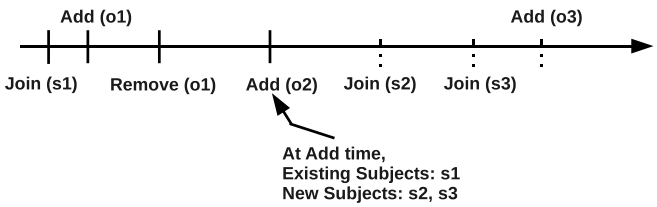

Figure 3: Object Operations Illustration.

since it is added after s1 leaves the group. When s1 rejoins the group, Lossless Join does not revoke access to o1 and o2. On the contrary, a Lossy Join can revoke access to some or all of the objects authorized prior to Join time. Following our example, when s1 rejoins the group, a Lossy Join revokes access to objects o1 and/or o2 that s1 had access prior to Join time. Note that when a subject joins a group for the first time, Lossy and Lossless Join operations behave exactly the same way. They differ only when subjects re-join.

A Lossy Join may sound counter-intuitive at first thought but there are real-world scenarios where such a behavior is desirable. Revisiting our meeting room scenario, during Alice's presence in the room she is allowed to take personal notes of the discussions and keep it when she leaves the room. But when re-joins the room at a later period, she may not be allowed to bring her personal notes. This is applicable in many scenarios such as students taking exams. Students take notes and can access course materials when they register for the course. Once the course is finished, they retain access to course materials to prepare for the exam. But when they take the exam (re-enter the room), they may not be allowed to access course materials.

4.1.2 Gainless Vs Gainful Leave

Similar to Join, the Leave operation could either be Gainless or Gainful. In Gainless Leave, a subject does not gain additional access to group objects by leaving a group. This means that if a subject is able to access an object after leaving the group, the access should have been authorized during the most recent membership period. On the contrary, in Gainful Leave, the subject may gain new access by leaving the group. For example, a subject may gain access to objects that were authorized during some prior membership and not necessarily the most recent membership. Such a behavior is desirable in situations where an incentive is provided to leave a group—commonplace in voluntary retirement or severance packages.

4.1.3 Non-Restorative Vs Restorative Join

The Join operation could further be Restorative or Non-Restorative of past accesses. When a subject joins a group, a Restorative Join restores access to objects that were authorized during his/her most recent membership period. On the other hand, a Non-Restorative Join does not restore past access when a subject rejoins the group.

Consider s1's most recent membership period in figure 2. During membership period, s1 is able to access o2. Suppose when s1 leaves the group, o2 cannot be accessed. When s1 rejoins, a Restorative Join will restore access to o2. A Non-restorative Join does not explicitly restore access to o2. However, s1 may be able to access o2 for other reasons. For example, suppose that the rejoin operation allows access to all *existing* objects, s1 may access o2 and o3 (be it Restorative or Non-restorative Join). However, if the rejoin operation allows s1 to access only *new* objects, a Restorative Join will allow s1 to access o2 whereas a Non-Restorative Join will not. A Restorative Join is desirable where a joining subject is given an incentive to re-join a group. This is typical in subscription models—when a customer renews membership, he/she may access all the earlier magazines that were authorized during past subscription in addition to new accesses granted by current subscription.

4.1.4 Non-Restorative Vs Restorative Leave

The Leave operation could also be Restorative or Non-Restorative of access authorized before current membership. Consider the most recent membership period in figure 2. Suppose that when s1 leaves the group he/she is allowed to retain access to o2. Further suppose that when s1 rejoins, access to o2 is lost (for e.g., due to a Lossy Join). During membership period, s1 is able to access o4 and o5. Now when s1 leaves the group, a Restorative Leave allows s1 to regain access to o2. Non-restorative Leave does not restore access to o2.

4.1.5 Formal Specification

Lossless Join Property: In a g-SIS model with Lossless Join, if a subject had access to an object before join time, the subject can continue to access the same after join time.

$$\Box((\text{Join} \wedge \neg\text{Remove} \wedge \ominus \text{Authz}) \rightarrow \text{Authz})$$

This formula states that if a subject joins a group and is authorized to access a group object in the previous state (just prior to Join), the subject can continue to access the object even after Join. Note that as per the Authorization Persistence Property, Authz will continue to hold as long a subject or an object event does not occur.

Lossy Join Property: A g-SIS model with Lossy Join does not satisfy the Lossless Join property. In other words, there exists at least one well-formed trace of subject and object events for which Authz does not satisfy the above property.

Gainless Leave Property: In Gainless Leave, on leaving a group, the subject cannot gain new access.

$$\Box((\text{Leave} \wedge (\neg\text{Join}\ \mathcal{U}\ \text{Authz})) \rightarrow \ominus ((\neg\text{Authz} \wedge \neg\text{Join}) \mathcal{S} (\text{Authz} \wedge (\neg\text{Join}\ \mathcal{S}\ \text{Join}))))$$

This formula says that if the subject is authorized to access an object during non-membership period then

it should have been authorized during the most recent membership period.

Gainful Leave Property: In a g-SIS model with Gainful Leave, there exists at least one well-formed trace of subject and object events for Authz does not satisfy the Gainful Leave property.

Non-Restorative Join Property: In Restorative Join, a joining subject gets access to objects authorized during past membership period irrespective of the type of the current Join operation. In Non-Restorative Join, past authorizations are not necessarily restored by current Join operation. The joining subject is able to access objects from past membership period only if allowed by the current Join operation and not otherwise. Thus if the joining subject is able to access objects that is not enabled by the current Join, then he/she should have had access to the same object just prior to joining the group.

Formalizing this property is complicated because we want our characterization to be independent of the exact semantics of the Join operation in question. Intuitively, we want to require that the Non-restorative Join does not add any authorizations that it would not have added if the subject had a different history. However LTL does not enable one to compare different traces. The solution we take is to consider two different subjects within a single trace. Because the two subjects can have different histories with the same trace, this strategy enables us to formalize the property in LTL as follows:

$$\rho_1 \equiv \text{join}_\text{i}(s1) \wedge \text{join}_\text{i}(s2)$$
$$\rho_2 \equiv (\text{Authz}(s1, o, r) \wedge \neg\text{Authz}(s2, o, r)) \rightarrow \ominus\, (\text{Authz}(s1, o, r) \wedge \neg\text{Authz}(s2, o, r))$$
$$\rho \equiv \forall i \Box(\rho_1 \wedge \rho_2)$$

In formula ρ_1, subjects s1 and s2 both join the group at the same time by means of the same type of Join (specifically join_i, where $1 \leq \text{i} \leq \text{m}$). ρ_2 says that if s1 is authorized to access an object in the current state and s2 is not, this should also be the case in the previous state (and vice-versa). The Non-Restorative Join property is characterized by formula ρ. It states that if two subjects Join the group at the same time with the same type of Join, then any difference in access at Join time is the result of some operation prior to the current Join operation. Let us use formula ρ_2 to understand the intuition. Because both s1 and s2 Join at the same time with same type, any access that is necessarily enabled by this Join for s1, would also be enabled for s2. Any additional access that s1 may have that s2 does not have could arise only because s1 had access to that object before joining the group. This captures the fact that access is not restored from past but is a consequence of the type of Leave operation applied to the subject when he/she left the group in the past.

Restorative Join Property: In Restorative Join, there exists at least one well-formed trace that does not satisfy the Non-Restorative Join property. If a subject joins a group using Restorative Join, some or all of the accesses to objects authorized during past membership period may be restored (unless it has been removed). Note that this is in addition to the authorizations that Join enables. The following formula characterizes a type of Restorative Join where *all* past authorizations are restored.

$$\Box(\text{Join} \wedge ((\neg\text{Leave} \wedge \neg\text{Remove})\ \mathcal{S}\ (\text{Leave} \wedge \ominus\, \text{Authz})) \rightarrow \text{Authz})$$

The above formula can be interpreted as follows. Consider the state at which a subject joins a group. Trace back to the point where the subject last left the group in the past. If the subject was authorized to access an object just prior to that point of Leave (a point that lies within the past membership period), then the subject is authorized to access that object at the point at which he/she joins if the object has not been removed from the group.

Non-Restorative Leave Property: In Non-Restorative Leave, past authorizations prior to current membership period is not explicitly restored when leaving a group. Thus if a subject is able to access an object after leaving a group, then it should have been authorized during the membership period.

$$\Box(\text{Leave} \wedge \text{Authz} \rightarrow \ominus\, \text{Authz})$$

Restorative Leave Property: In Restorative Leave, there exists at least one well-formed trace that does not satisfy the Non-Restorative Leave Property. The following formula characterizes a specific type of Restorative Leave Property where access to *all* objects authorized prior to Join is restored.

$$\Box((\text{Leave} \wedge \neg\text{Remove} \wedge \ominus\, ((\neg\text{Leave} \wedge \neg\text{Remove})\ \mathcal{S}\ (\text{Join} \wedge \ominus\, \text{Authz}))) \rightarrow \text{Authz})$$

The above formula is similar to Restorative Join except that we now trace back to the most recent Join from the point of Leave and if the subject was authorized prior to Join, the subject is also authorized at the time of Leave, provided the object has not been removed.

4.2 Level 2 Properties (Strict Vs Liberal)

We now discuss the next level of g-SIS properties (Level 2) based on additional variations of group operations. These operations apply to variations (Lossy, Lossless, Restorative, Non-Restorative) identified in Level 1. Each group operation can further be Strict or Liberal based on the nature of access that it allows. Thus we have Strict and Liberal versions of Join, Leave, Add and Remove denoted SJ, SL, SA and SR and LJ, LL, LA and LR respectively. In general, a Strict operation is more restrictive of access than its Liberal counterpart.

A g-SIS model allows four group operations: (Join, Leave, Add, Remove). If the type of operations are fixed for all subjects and objects (i.e. a specific type of operation is applied for all group subjects and objects), there are 16 possible models ranging from the most restrictive model allowing only Strict operations: (SJ, SL, SA, SR) to the most permissive model allowing only Liberal operations: (LJ, LL, LA, LR). This is illustrated in figure 4. Parts (a) through (d) show that the Strict operation is more restrictive than the Liberal operation. Parts (e) and (f) show the subject and object model that is obtained by the cartesian product of

subject and object operations respectively. Finally, a lattice of 16 g-SIS models can be obtained by a cartesian product of subject and object models (parts (e) and (f)). An authorization policy exists for each of these 16 models that specify the conditions under which a subject may access an object. On the other hand, a highly flexible g-SIS model could simply allow different types of operations on a case by case basis. For example, SJ for s1, LJ when s1 re-joins, LJ for s2, LL for s1, SL for s2, SJ when s2 re-joins, etc. (similarly for objects). In this case, we have one all encompassing authorization policy that specifies the conditions under which a subject may access an object.

4.2.1 Subject Operations

We discuss the notion of Strict and Liberal for subject operations below.

Strict Join (SJ): In Strict Join, a joining subject can only access *new* group objects as they are added. Suppose that in figure 2 the second Join (s1) is an SJ. Then s1 can access o4 and o5 and cannot access o3. Note that access to o2 will depend on the type of the previous Leave and current Join (Lossy or Lossless).

Liberal Join (LJ): In Liberal Join, a joining subject can access both *existing* and *new* group objects. In figure 2, if the Join was an LJ instead of SJ, s1 can also access o2 and o3 in addition to o4 and o5.

Strict Leave (SL): In Strict Leave, a leaving subject loses access to all group objects. In figure 2, on SL, s1 loses access to all group objects authorized during membership period. Note that this is a specific type of Lossy Leave where all access is lost by leaving a group.

Liberal Leave (LL): In Liberal Leave, a leaving subject can retain access to all objects authorized during his/her recent membership period. Suppose that in figure 2 the Leave is an LL. In this case, s1 retains access to o2. Note that, after Leave, o3 cannot be accessed by s1. This is a Lossless Leave.

4.2.2 Object Operations

We discuss the notion of Strict and Liberal for object operations below.

Strict Add (SA): In Strict Add, the added object can be accessed only by *existing* group subjects. If Add (o2) in figure 3 is an SA, only s1 can access the object. Subjects s2 and s3 joining later cannot access this object.

Liberal Add (LA): In Liberal Add, the added object can be accessed by both *existing* and *new* group subjects. Thus if Add (o2) is an LA in figure 3, *existing* subject s1 and new *subjects* s2 and s3 may access o2.

Strict Remove (SR): In Strict Remove, the removed object cannot be accessed by any group subject. In figure 3, if Remove (o1) is an SR, every group subject (including s1) loses access to o1.

Liberal Remove (LR): In Liberal Remove, subjects who had access to the object at remove time can retain access. However, other subjects will not be able to access the removed object unless the object is added again in the future. In figure 3, if Remove (o1) is an LR, s1 can continue to access o1. However s2 and s3 will not have access to o1.

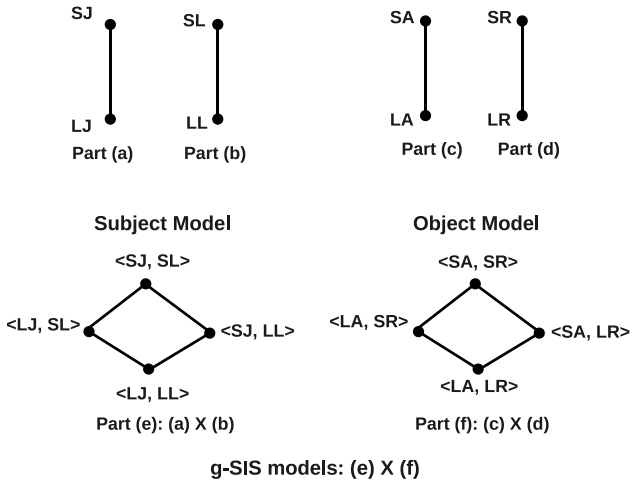

Figure 4: A family of g-SIS models: *The cartesian product of Subject and Object Model results in a lattice of 16 g-SIS models with fixed operation types (products are ordered pointwise).*

4.3 Discussion

Let us now discuss how these operations enable information sharing. We make a few simplifying assumptions to illustrate the mechanics. Consider a g-SIS model with the operation types: (LJ, SL, SA/LA, SR) where all operations are fixed except object Add. Objects can be added to the group by type SA or LA. Let us consider the simplest case of information sharing where a group consists of at most two members at any time but may have any number of objects. The group is mission oriented, so many users may join and leave the group in order to contribute and receive information over time.

Suppose Alice and Bob join the group at the same time. They can share information by adding objects to the group. If Bob wants to ensure that any information he shares with Alice is not accessible to future subjects who may join the group, he can add objects with SA. SA'ed objects are only accessible to *existing* members at add time. This allows current members of the group to share information privately. On the other hand, to the mission's end, information can be made available to future subjects by LA'ing objects to the group. Suppose Alice leaves the group and Cathy replaces her by LJ to the group. Cathy cannot access SA'ed information shared between Alice and Bob before her join time. In other words, Cathy can access existing LA'ed group objects that were added before her join time and newly added objects.

Further, suppose Alice re-joins the group by LJ, replacing Bob. Alice can access LA'ed objects shared between Bob and Cathy. Note that Alice cannot access SA'ed objects that Bob added to the group during her past membership along with Bob. This scenario is illustrative of the need for Restorative Join. If Alice is LJ'ed in a Restorative manner to the group, she can regain access to SA'ed objects during her past membership period.

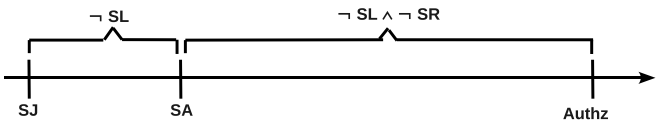

Figure 5: Fixed Operations, Most Restrictive Model: (SJ, SL, SA, SR).

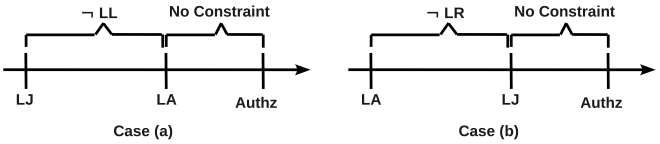

Figure 6: Fixed Operations, Most Permissive Model: (LJ, LL, LA, LR).

In all these steps, a leaving subject (SL'ed) loses access to all group objects. Similarly, a removed object (SR'ed) cannot be accessed by any past or current group subject. If the mission requires access to objects after leaving a group, one needs a model with LL.

5. FORMAL SPECIFICATION OF G-SIS

In this section, we formally specify the authorization policies for g-SIS models using LTL. One can clearly appreciate a multitude of variations of group operations leading to a large number of g-SIS models. In this paper, we specify authorization policies for a sub-family of models involving the following operation types: Lossless Non-Restorative Join and both Lossy and Lossless Non-Restorative Leave. Other g-SIS models can be easily developed by extending this sub-family. To this end, we will later show how an authorization policy specified under these constraints can be extended to accommodate Restorative Join operations. From here on, whenever we use the term Join, we mean Lossless and Non-Restorative Join (unless specifically qualified otherwise). Similarly, the term Leave henceforth refers to Non-Restorative Leave. Note that Leave could be either Lossy or Lossless.

Thus, in the following, Strict Join (SJ) and Liberal Join (LJ) actually refer to Lossless, Non-Restorative SJ and LJ respectively. Similarly, Strict Leave (SL) and Liberal Leave (LL) refer to Non-Restorative SL and LL respectively. Further, SL and LL characterizes Lossy and Lossless Leave respectively.

We first consider the models where the operations are fixed for all subjects and objects. As shown in figure 4, there are 16 such models. Due to space constraints, we discuss only two fixed models in this paper—a model that is most restrictive and another that is most permissive in terms of access granted. Note that if the restrictive model permits a subject to access an object, the permissive model should also grant access to the same object.

Next, we consider a highly flexible g-SIS model where any type of operation is permitted and the type of operation could differ from subject to subject (or from object to object). For example, SJ for s1, LJ for s2, etc. Effectively, the authorization policy for this model with mixed operations should subsume all the 16 models with fixed operations. While this may sound very complex, it will be shown that by systematic analysis, it is possible to write a succinct yet expressive formula that can cover all the cases. We will also illustrate how the two models with fixed operations that we discuss conform precisely to this more general model.

5.1 Models with Fixed Operations

We now specify the authorization policies for the most restrictive and most permissive of the 16 g-SIS models with fixed operations type. The remaining 14 models are more restrictive than this permissive model and more permissive than this restrictive model.

5.1.1 *Most Restrictive Model (*SJ, SL, SA, SR*)*

Recall that on SJ, a subject can only access *new* objects. On SL, the subject loses access to all group objects. Similarly, on SA, only *existing* group subjects can access the added objects. An SR'ed object cannot be accessed by any subject.

Definition 5.1 (Most Restrictive Model). *A g-SIS model is called the Most Restrictive if it satisfies the following LTL formula:*

$$\Box(\text{Authz} \leftrightarrow (\neg\text{SR} \wedge \neg\text{SL})\ \mathcal{S}\ (\text{SA} \wedge (\neg\text{SL}\ \mathcal{S}\ \text{SJ})))$$

The above formula is illustrated in figure 5. The formula says that a subject is authorized to access an object if and only if the subject and object are still part of the group since it was added. Also, at the time the object was added, the subject was a current member of the group. Because of SJ and SL, we only need to consider the case where an object is added after the subject joins the group. Subjects are not authorized to access objects added prior to their join time.

5.1.2 *Most Permissive Model (*LJ, LL, LA, LR*)*

In this model, an LJ'ed subject can access both *existing* and *new* objects. On LL, the subject can continue to access objects that were authorized during the membership period. An LA'ed object can be accessed by both *existing* and *new* subjects. Subjects who were authorized to access an object at the time it is LR'ed can continue to access the object.

Definition 5.2 (Most Permissive Model). *A g-SIS model is called the Most Permissive Model if it satisfies the following LTL formula:*

$$\Box(\text{Authz} \leftrightarrow \blacklozenge(\text{LA} \wedge (\neg\text{LL}\ \mathcal{S}\ \text{LJ}))\ \vee\ \blacklozenge(\text{LJ} \wedge (\neg\text{LR}\ \mathcal{S}\ \text{LA})))$$

Since this model permits LJ, there are two scenarios (figure 6): one where the subject joins prior to the time at which the object in question is added (case (a)) and the other where the subject joins after the object in question is added (case (b)). The first disjunct in the above formula covers the former case and the second disjunct covers the latter case. Note that the $\blacklozenge$ operator in each case says that the respective traces could occur any time in the past. Naturally, the Most Permissive model is exactly the same as the Overlapping Membership Property with an if and only if condition. Effectively, if a subject and an object were members at the same time at any time in the past, this model permits the subject to access the object and the access will never be revoked.

5.2 Models with Mixed Operations

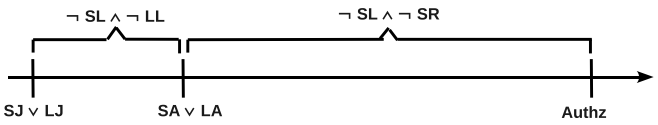

Figure 7: Mixed Operations - Formula φ_1.

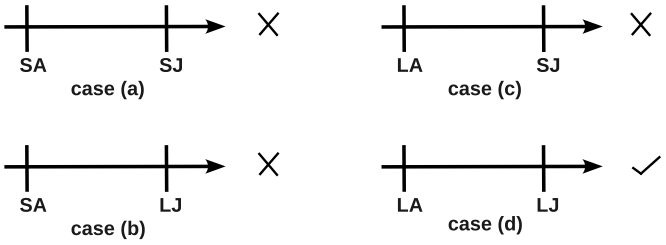

Figure 8: Cases when Add **occurs prior to** Join**.**

We now discuss the specification of authorization policy for a g-SIS model that allows any type of group operation [5]. The formulas that we discussed for models with fixed operation types give us some insight on handling mixed operation models. At a high-level there are two cases to consider when a subject requests access to an object: (a) the subject Join event occurred prior to object Add event and (b) the object add event occurred prior to subject Join event. Intuitively, an authorization policy that correctly addresses these two cases would be complete. We now separately consider these two cases.

$$\varphi_1 \equiv((\neg \text{SL} \wedge \neg \text{SR})\ \mathcal{S}\ ((\text{SA} \vee \text{LA}) \wedge ((\neg \text{LL} \wedge \neg \text{SL})\ \mathcal{S}\ (\text{SJ} \vee \text{LJ}))))$$

Formula φ_1 addresses the scenario where the object is added after the subject joined the group (figure 7). Since Join occurred prior to Add, it does not matter whether the object in question was LA'ed or SA'ed to the group and whether the subject in question SJ'ed or LJ'ed the group. The subject can access the object in both cases as per our Liveness Properties. Formula φ_1 says that the subject has not been SL'ed and the object has not been SR'ed since it was added. Further, when the Add occurred the subject had not left (SL'ed or LL'ed) since Join (SJ or LJ).

In figure 7, an SL or SR since object add time denies access to the requested object. However, it is alright for an LL or LR to occur during that period. Recall that an LR allows subjects who were authorized prior to the remove time to retain access and LL allows a leaving subject to retain access to objects authorized during membership period. Similarly, if the subject was not a current member when the object was added (i.e., joined and left the group before the object was added), access cannot be granted as per Overlapping Membership property. Note that in figure 7 it is possible the subject may Join and Leave before the Add occurs and rejoin after the Add. But this scenario where a Join occurs after Add is addressed by formula φ_2.

An Add occurring prior to Join is a more interesting scenario. As shown in figure 8, there are four possible cases. Let us first consider cases (a) and (b) where the object is SA'ed to the group. Recall that an SA'ed object can be accessed only by *existing* subjects (that is, the subjects who joined the group prior to object Add). Clearly, irrespective of the type of Join, the subject is not authorized to access the objects that were SA'ed prior to the subject Join time. Thus Authz fails in cases (a) and (b).

Consider cases (c) and (b) where the object is LA'ed to the group. In case (c), the object is LA'ed and the subject is SJ'ed. An SJ'ed subject can only access objects added after join time. Thus (c) is also a failed case. Authorization is successful in case (d) where both Add and Join are Liberal operations. An LJ'ed subject can access all *existing* LA'ed objects and *new* group objects. An LA'ed object can be accessed by all *existing and new* group subjects. In short, an LJ'ed subject can access any *new* group object and *existing* LA'ed group objects.

[5]We later show how this formula can be extended to accommodate models that allow Restorative Joins. It is possible to extend this core formula to accommodate models that allow other operations (Lossy Join, Restorative Leave, etc.).

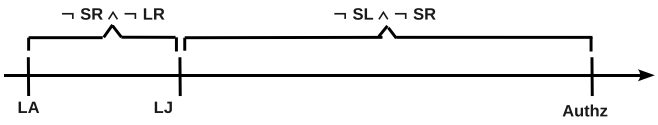

Figure 9: Mixed Operations - Formula φ_2.

$$\varphi_2 \equiv((\neg \text{SL} \wedge \neg \text{SR})\ \mathcal{S}\ (\text{LJ} \wedge ((\neg \text{SR} \wedge \neg \text{LR})\ \mathcal{S}\ \text{LA})))$$

We can now formulate φ_2 as shown above. Figure 9 illustrates φ_2. It says that the subject has not been SL'ed and the object has not been SR'ed since the subject LJ'ed the group. Further, at Join time, the object in question was still part of the group (that is, it has not been LR'ed or SR'ed since it was added).

A Mixed g-SIS Model is a g-SIS model that supports mixed group operations. We now state the authorization policy for this model.

Definition 5.3 (Fixed g-SIS Model). *A g-SIS model is called a Fixed Model if it satisfies the following LTL formula:*

$$\Box(\text{Authz} \leftrightarrow\ \varphi_1 \vee \varphi_2)$$

5.2.1 Deriving Fixed Operation Models

Intuitively, the Mixed g-SIS Model should subsume all the 16 g-SIS models with fixed operations. And indeed, it is straight-forward to derive any fixed operation model by substituting operations that are not supported by that model with "False" in the Mixed Model in definition 5.3.

Consider the Most Restrictive Model (definition 5.1): (SJ, SL, SA, SR). We can derive the formula for this model by substituting all unsupported operations with "False" in formulas φ_1 and φ_2. Unsupported operations never occur (or are ignored), so the corresponding events never occur. Note here that φ_2 fails on the whole since LJ is not allowed in the Most Restrictive Model. By substituting "False" for LJ, LL and LA in formula φ_1, we get the same formula in definition 5.1. We similarly obtain the Most Permissive Model (definition 5.2): (LJ, LL, LA, LR) by substituting "False" for SJ, SL, SA and SR in formula φ_1 and φ_2. Similarly, we easily derive another model with fixed operations, (LJ, SL,

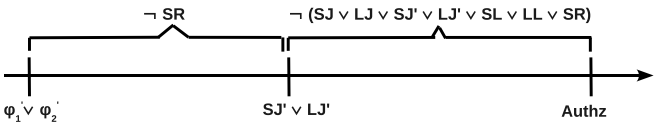

Figure 10: Mixed g-SIS Model with Restorative Join.

LA, SR), as below:

$$\lambda_1 \equiv (\neg \text{SL} \wedge \neg \text{SR})\ \mathcal{S}\ (\text{LA} \wedge (\neg \text{SL}\ \mathcal{S}\ \text{LJ})) \qquad (\text{From } \varphi_1)$$
$$\lambda_2 \equiv (\neg \text{SL} \wedge \neg \text{SR})\ \mathcal{S}\ (\text{LJ} \wedge (\neg \text{SR}\ \mathcal{S}\ \text{LA})) \qquad (\text{From } \varphi_2)$$

And Authz for this model is stated as below:

$$\Box(\text{Authz} \leftrightarrow\ \lambda_1 \vee \lambda_2)$$

Recall that we mentioned g-SIS models subsume the Secure Multicast policies. In Secure Multicast, when a node joins the group it can only access newly transmitted data and on leave it may retain access to those received during membership period. It is not clear if the notion of object operations exist in multicast. Thus a g-SIS model that supports (SJ, LL, SA/LA, SR/LR) operations is similar to the policies allowed by multicast models.

5.2.2 *Restorative Join*

We now show how our Mixed g-SIS Model can be extended to support Restorative Join operations. Recall that a Restorative Join explicitly restores access to objects authorized during previous membership period as long as those objects are still part of the group. A Mixed Restorative g-SIS Model supports four Join operations: SJ and LJ (the Non-Restorative Join operations) and SJ′ and LJ′ (their Restorative counter-parts). Thus SJ′ is an SJ which further restores past accesses (similarly LJ′)[6]. First, we re-write φ_1 and φ_2 to include Restorative Join operations (SJ′ and LJ′) as shown below. Formula δ_3 (see figure 10) says that the subject has not joined or left or the object has not been SR'ed (formula δ_2) since the subject joined the group in a restorative manner (this check simply ensures that we are referring to the most recent Join and that the Join is Restorative). Further, the object has not been SR'ed since a point in the past at which the subject was authorized ($\varphi_1' \vee \varphi_2'$) to access the object. It is easy to understand this formula by visualizing a subject who joins the group for the second time. The first time the subject joins, the subject would be authorized to access objects as per δ_1. When the subject leaves and re-joins the second time, it is possibly a Restorative Join. At this point, formula δ_3 allows access to objects authorized during the first membership period.

$$\varphi_1' \equiv ((\neg \text{SL} \wedge \neg \text{SR})\ \mathcal{S}\ ((\text{SA} \vee \text{LA}) \wedge ((\neg \text{LL} \wedge \neg \text{SL})\ \mathcal{S}\ (\text{SJ} \vee \text{SJ}' \vee \text{LJ} \vee \text{LJ}'))))$$
$$\varphi_2' \equiv ((\neg \text{SL} \wedge \neg \text{SR})\ \mathcal{S}\ ((\text{LJ} \vee \text{LJ}') \wedge ((\neg \text{SR} \wedge \neg \text{LR})\ \mathcal{S}\ \text{LA})))$$
$$\delta_1 \equiv (\varphi_1' \vee \varphi_2')$$
$$\delta_2 \equiv (\text{SJ} \vee \text{LJ} \vee \text{SJ}' \vee \text{LJ}' \vee \text{SL} \vee \text{SR})$$
$$\delta_3 \equiv (\neg\delta_2\ \mathcal{S}\ ((\text{SJ}' \vee \text{LJ}') \wedge (\neg \text{SR}\ \mathcal{S}\ \delta_1)))$$

We now define a g-SIS Model that supports Restorative Join operations.

Definition 5.4 (Mixed Restorative g-SIS Model). *In a Mixed Restorative g-SIS Model, a subject is allowed to access an object if it satisfies the following LTL formula:*

$$\Box(\text{Authz} \leftrightarrow\ \delta_1 \vee \delta_3)$$

In this model, a subject is authorized to access an object if: (a) the access is authorized by the current membership ($\varphi_1 \vee \varphi_2$) or (b) the access was authorized during past membership period and there has been no SR since (δ_3).

We have shown how the Mixed Model can be extended to support Restorative Join. It is also possible to extend this model to support other variations such as Lossy Join, Restorative Leave, etc.

5.3 Verification of Core Properties

In this section, we discuss how we verified that the specification of the Mixed g-SIS Model (Definition 5.3) entails the core g-SIS properties specified in section 3 using model checking. Model checking [13, 14] is an automated verification technique that analyzes a finite model of a system (i.e., a finite state machine (FSM) that produces computation traces consisting of infinite sequences of states) and exhaustively explores the state space of the model to determine whether desired properties hold in the model. In the case that a property is false, a model checker produces a counterexample consisting of a trace that violates the property, which can be used to correct the model or modify the property specification.

Symbolic Model Verifier (SMV) [24, 12] is a family of model checking tools based on Binary Decision Diagrams (BDDs). BDDs represent states very compactly. In SMV, models are represented by using variables that are assigned values in each step of the FSM. Properties to be checked are specified by temporal logic [22] formulas. SMV verifies the properties against the model and returns a counter-example if the model does not satisfy the property.

The FSM we need to construct for the purpose of our proof is very simple. The FSM (appendix A) simply models a system where Join, Leave, Add and Remove events are allowed to occur concurrently, non-deterministically and in any order. Such an FSM produces all possible traces of group events. We use the model checker to prove that our LTL specification entails the core properties. We formalize an implication having the well-formedness conditions and the specification of the Mixed g-SIS model in the antecedent. The consequence contains the required properties identified here in earlier sections. Thus we verify that all traces that satisfy the specification also satisfy the required properties. This is a non-traditional use of the model checker in which we use the tool as a theorem prover. Further, we force an

[6]Note that the difference between LJ and LJ′ is subtle. They both allow access to all *existing and new* group objects at Join time. But they may differ in other ways. For example, if an object has been LR'ed and the subject had past access to it, LJ′ restores access whereas LJ does not.

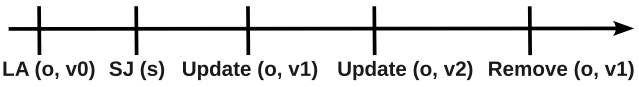

Figure 11: Read-Write g-SIS model with versioning.

additional constraint that in the initial state, only a Join or an Add can occur. This is a reasonable constraint because at system initialization, there could be no subject or object in the system for a Leave or Remove to occur. Suppose σ_0 denotes a conjunction of all of our core properties and σ_1 denotes the set of all well-formed traces from our FSM. Formula σ should hold:

$$\sigma_0 \equiv \text{Core gSIS Properties}$$
$$\sigma_1 \equiv \text{Legal gSIS Traces}$$
$$\sigma_2 \equiv (\text{Authz} \leftrightarrow \varphi_1 \vee \varphi_2) \qquad \text{(Definition 5.3)}$$
$$\sigma \equiv (\sigma_1 \wedge \sigma_2 \rightarrow \sigma_0)$$

Note that σ_2 above is our policy specification. We use the model checker to verify that formula σ always holds. The results from NuSMV (a mature open source model checker) [12] are shown in appendix A. We find that the core properties are in fact satisfied by the g-SIS specification. Further, as mentioned earlier, we show that Safety Properties can be inferred from Overlapping Membership Property by using the same approach. Formalizing our models and properties using LTL allows us to rapidly verify any new property. Using this approach, verification of LTL specifications can be easily automated using model checking.

6. FUTURE WORK

In this section, we discuss some of the research challenges involved in developing this area of work. We identify a few open problems in extending the current model to support Read-Write operations and multiple groups.

6.1 Read-Write g-SIS Model

In the models we considered so far, we assumed that subjects can Add new objects and perform only Read operations on those objects. Let us now discuss some of the issues in Read-Write g-SIS models. In a sense, a Read-only model can simulate a Read-write model by removing, updating and re-adding the object. While this is a feasible approach, there are some critical issues in such a model. If an object is removed and re-added, it is possible that subjects joining the group between Remove and Add can access the object. Re-adding changes the state of the object thereby possibly authorizing additional subjects as a consequence. Such issues force us to consider a "Write" or an "Update" operation distinctly.

In general, we believe that a Read-Write model should support versioning due to the distributed nature of the SIS problem. It is unreasonable to expect concurrency control since subjects may have intermittent connectivity to the servers and may be allowed to access objects offline. Further, such a solution does not scale well in an SIS scenario since a large number of subjects may be collaborating on a specific set of objects and locking objects for updating information is highly inefficient and counter-productive. Versioning is crucial to the usability of a Read-Write g-SIS model. However, there are many issues that need to be resolved in the case of Read-Write g-SIS model with versioning.

Consider the scenario in figure 11 where a subject s is SJ'ed to the group and object o is updated resulting in various versions. The first question is whether s should be allowed to access o since v0 of o was added prior to s's Join time. While a simple answer could be no, it would not be the most flexible model—recall that our Liveness property requires that any object that is added after a subject joins the group should be accessible. Practically, we would want to support both options. Suppose s is allowed to access o, is s allowed to access only v1 and v2 or also v0 of o? Similarly, what should a Remove operation mean in such a context? Suppose v1 is removed, can v0 exist? Should the Remove operation have a cascade effect and wipe out all previous versions? Also, can v2 exist after v1 is removed? We believe that a lot of these questions are dependant on the object or information model and so the information sharing model should accommodate all such options. Further, we would have to re-visit the core and additional properties in the context of Read-Write model. Would the core properties be reasonable for Read-Write models? Are there any new core/additional properties?

6.2 Hierarchical g-SIS Model

Let us now consider a g-SIS model in the context of multiple groups. Clearly, it would not be useful (nor interesting) to consider groups that are unrelated—they would simply be independent groups. A natural way to create a structure is to impose a hierarchy similar to Lattice-Based Access Control (LBAC) Models [29] for information flow [15] such as Bell-LaPadula (BLP) [8], Biba [10], Chinese Wall [11], etc. A hierarchical g-SIS model can be configured in many interesting ways. Consider a hierarchy of g-SIS groups. A subject may only read objects at the same group or any lower group. Note that an additional constraint would be subjects can only read objects that are permitted by the g-SIS policies we specified in this paper. Similarly, subjects may only add information at the same group or to higher groups. This is similar to the BLP model for confidentiality (the labels in BLP are similar to groups in g-SIS) but differs in that it has an important temporal element for information sharing within the same or across groups (or labels). This allows a subject to protect information added from other subjects in the same or higher levels. Similarly Chinese Wall policies can be achieved in g-SIS with the notion of Lossy Join. An interesting exercise is to see if the g-SIS models can simulate the LBAC models. An intuitive approach is to see if g-SIS models can express RBAC [30] policies since it has been shown that RBAC can be configured to enforce both MAC and DAC policies [25]. A positive result would show that g-SIS can simulate a whole range of information flow models.

7. CONCLUSION

In this paper, we proposed a group-centric family of models for Secure Information Sharing. We identified a core set of properties that should be satisfied by the g-SIS models. We also identified an additional set of properties in light of many variations of group operations (Lossless, Lossy, Restorative, Non-Restorative, Strict and Liberal). We formally specified the properties using LTL making them suitable to be verified by using model checking which is a highly efficient and re-usable technique. We formally specified a

lattice of 16 g-SIS models with fixed group operations. On the hand, we also specified a highly flexible model that allows any variation of group operations. We showed that the 16 g-SIS models with fixed operations can be derived from single authorization policy for the mixed operation model. Finally, we formally proved by using model checking that the final g-SIS specification semantically entails the core g-SIS properties. Our future research in this line of work is along two major directions as identified earlier. First, we are interested in developing this model to support both read and write operations. Next, we are investigating extensions to g-SIS models in the context of multiple groups. The groups could be highly structured like a hierarchy or completely structure-less yet related.

8. REFERENCES

[1] eXtensible rights Markup Language. *www.xrml.org*.

[2] OASIS eXtensible Access Control Markup Language . *www.oasis-open.org/committees/xacml/*.

[3] SISA | Secure Information Sharing Architecture. *http://www.sisaalliance.com/*.

[4] The Open Digital Rights Language Initiative. *www.odrl.net*.

[5] M. Abrams, J. Heaney, O. King, L. LaPadula, M. Lazear, and I. Olson. Generalized Framework for Access Control: Towards Prototyping the ORGCON Policy. *Proceedings of the 14th National Computer Security Conference*, pages 257–266, 1991.

[6] G.-J. Ahn, B. Mohan, and S.-P. Hong. Towards secure information sharing using role-based delegation. *J. Network and Computer Applications*, 30(1):42–59, 2007.

[7] V. Atluri and J. Warner. Automatic Enforcement of Access Control Policies Among Dynamic Coalitions. *International Conference on Distributed Computing & Internet Technology, Bhubaneswar, India, Dec*, 2004.

[8] D. Bell and L. La Padula. Secure computer systems: Unified exposition and multics interpretation.

[9] V. Bharadwaj and J. Baras. A framework for automated negotiation of access control policies. *DARPA Information Survivability Conference and Exposition, 2003. Proceedings*, 2, 2003.

[10] K. Biba. Integrity considerations for secure computer systems. *Technical Report TR-3153*, April 1977.

[11] D. Brewer and M. Nash. The Chinese Wall security policy. *IEEE Symposium on Security and Privacy*, pages 206–214, 1989.

[12] A. Cimatti, E.M.Clarke, F. Giunchiglia, and M. Roveri. NuSMV: A new symbolic model checker. *International Journal on Software Tools for Technology Transfer*, 2(4):410–425, 2000.

[13] E. M. Clarke, E. A. Emerson, and A. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Language and Systems (TOPLAS)*, 8(2):244–263, 1986.

[14] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press, Cambridge, Massachusetts, USA, 1999.

[15] D. Denning. A Lattice Model of Secure Information Flow. *Communications of the ACM*, 19(5):236–243, 1976.

[16] DoD National Computer Security Center (DoD 5200.28-STD). *Trusted Computer System Evaluation Criteria*, December 1985.

[17] G. Graham and P. Denning. Protection-principles and practice. *Proceedings of the AFIPS Spring Joint Computer Conference*, 40:417–429, 1972.

[18] R. Graubart. On the Need for a Third Form of Access Control. *Proceedings of the 12th National Computer Security Conference*, pages 296–304, 1989.

[19] H. Khurana and V. Gligor. A Model for Access Negotiations in Dynamic Coalitions. *Proceedings of the 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE '04)*, pages 205–210.

[20] H. Khurana, V. Gligor, and J. Linn. Reasoning about joint administration of access policies for coalition resources. In *ICDCS '02: Proceedings of the 22 nd International Conference on Distributed Computing Systems (ICDCS'02)*, page 429, Washington, DC, USA, 2002. IEEE Computer Society.

[21] B. Lampson. Protection. *ACM SIGOPS Operating Systems Review*, 8(1):18–24, 1974.

[22] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems*. Springer-Verlag, Heidelberg, Germany, 1992.

[23] C. McCollum, J. Messing, and L. Notargiacomo. Beyond the pale of MAC and DAC - defining new forms of access control. *Proceedings of the 1990 IEEE Symposium on Security and Privacy*, pages 190–200, 1990.

[24] K. McMillan. *Symbolic Model Checking: An Approach to the State Explosion Problem*. Kluwer Academic, 1993.

[25] S. Osborn, R. Sandhu, and Q. Munawer. Configuring Role-Based Access Control to Enforce Mandatory and Discretionary Access Control Policies. *ACM Transactions on Information and System Security*, 3(2):85–106, 2000.

[26] J. Park and R. Sandhu. Originator control in usage control. *Policies for Distributed Systems and Networks, 2002. Proceedings. Third International Workshop on*, pages 60–66, 2002.

[27] C. Phillips Jr, T. Ting, and S. Demurjian. Information sharing and security in dynamic coalitions. *Proceedings of the 7th ACM Symposium on Access Control Models and Technologies*, pages 87–96, 2002.

[28] S. Rafaeli and D. Hutchison. A survey of key management for secure group communication. *ACM Computing Surveys*, pages 309–329, September 2003.

[29] R. Sandhu. Lattice-Based Access Control Models. *IEEE Computer*, 26(11):9–19, 1993.

[30] R. Sandhu, E. Coyne, H. Feinstein, and C. Youman. Role-Based Access Control Models. *IEEE Computer*, pages 38–47, 1996.

[31] R. Sandhu, K. Ranganathan, and X. Zhang. Secure information sharing enabled by trusted computing and PEI models. In *ASIACCS '06: Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, pages 2–12, New York, NY, USA, 2006. ACM.

[32] J. Warner, V. Atluri, R. Mukkamala, and J. Vaidya.

Using semantics for automatic enforcement of access control policies among dynamic coalitions. In *SACMAT '07: Proceedings of the 12th ACM symposium on Access control models and technologies*, pages 235–244, New York, NY, USA, 2007. ACM.

APPENDIX

A. MODEL CHECKING

The following NuSMV code listing simply allows group events such as SJ, LJ, SR, etc. to occur concurrently in a non-deterministic manner. This produces all possible traces. We discard all the illegal traces based on our assumptions in section 3 and verify that the specification entails the core properties. Due to space constraints, we are only able to show few results from NuSMV. The final output shows that the specification does entail the Overlapping Membership property. Further, the safety properties entail the overlapping membership property as well. A more exhaustive analysis with counter-examples can be found at: **http://sites.google.com/site/nusmv24/**.

Code Listing

```
MODULE main
VAR
SL: boolean;
LL: boolean;
    SA: boolean;
LA: boolean;
SJ: boolean;
LJ: boolean;
SR: boolean;
LR: boolean;
authz: boolean;
DEFINE
join := SJ |LJ;
    leave := SL|LL;
    add := SA | LA;
remove := SR | LR;
xjoin := SJ xor LJ;
xleave := SL xor LL;
    xadd := SA xor LA;
xremove := SR xor LR;
```

–Legal Traces: LT

–(LT & Mixed g-SIS authz) -> Overlapping – Membership Property

```
LTLSPEC
    (G (!(add & remove) & !(join & leave) &
  (xjoin  -> X ((!join U leave) | G !join)) &
(xleave -> X((!leave U join) | G!leave)) &
(xremove -> X((!remove U add) | G!remove)) &
(xadd -> X((!add U remove) | G!add))) &
(!remove U add) &
(!leave U join) &

(G ( authz <->
(((!SL&!SR) S ((SA|LA) & ((!LL&!SL) S (SJ|LJ)))) |
((!SL&!SR) S (LJ& ((!SR&!LR) S LA)))))))
->
(G (authz -> O (add & (!leave S join)) |
O (join & (!remove S add))))
```

–(LT & Mixed g-SIS authz) -> authorization –persistence

```
LTLSPEC
    (G (!(add & remove) & !(join & leave) &
  (xjoin  -> X ((!join U leave) | G !join)) &
(xleave -> X((!leave U join) | G!leave)) &
(xremove -> X((!remove U add) | G!remove)) &
(xadd -> X((!add U remove) | G!add))) &
(!remove U add) &
(!leave U join) &

(G ( authz <->
(((!SL&!SR) S ((SA|LA) & ((!LL&!SL) S (SJ|LJ)))) |
((!SL&!SR) S (LJ& ((!SR&!LR) S LA)))))))

->
(G (authz -> X ( (authz U (join | leave |
add | remove)) | G authz)))
--
```

–(LT & Mixed g-SIS authz) -> Subject liveness

```
LTLSPEC
    (G (!(add & remove) & !(join & leave) &
  (xjoin  -> X ((!join U leave) | G !join)) &
(xleave -> X((!leave U join) | G!leave)) &
(xremove -> X((!remove U add) | G!remove)) &
(xadd -> X((!add U remove) | G!add))) &
(!remove U add) &
(!leave U join) &

(G ( authz <->
(((!SL&!SR) S ((SA|LA) & ((!LL&!SL) S (SJ|LJ)))) |
((!SL&!SR) S (LJ& ((!SR&!LR) S LA)))))))

->
(G (join -> ( ( ( add -> ((authz U (leave | remove)) |
G authz) ) U leave ) | G( add -> ((authz U (leave |
remove)) | G authz)))))
--
```

–(LT & Mixed g-SIS authz) -> Object liveness

```
LTLSPEC
    (G (!(add & remove) & !(join & leave) &
  (xjoin  -> X ((!join U leave) | G !join)) &
(xleave -> X((!leave U join) | G!leave)) &
(xremove -> X((!remove U add) | G!remove)) &
(xadd -> X((!add U remove) | G!add))) &
(!remove U add) &
(!leave U join) &

(G ( authz <->
(((!SL&!SR) S ((SA|LA) & ((!LL&!SL) S (SJ|LJ)))) |
((!SL&!SR) S (LJ& ((!SR&!LR) S LA)))))))

->
(G ((add & (!leave S join)) -> ((authz U (leave |
remove)) | G authz)))
--
```

–(LT & Mixed g-SIS authz) -> subject safety property

```
LTLSPEC
    (G (!(add & remove) & !(join & leave) &
  (xjoin  -> X ((!join U leave) | G !join)) &
(xleave -> X((!leave U join) | G!leave)) &
(xremove -> X((!remove U add) | G!remove)) &
(xadd -> X((!add U remove) | G!add))) &
(!remove U add) &
(!leave U join) &

(G ( authz <->
(((!SL&!SR) S ((SA|LA) & ((!LL&!SL) S (SJ|LJ)))) |
((!SL&!SR) S (LJ& ((!SR&!LR) S LA)))))))
->
(G (leave -> ((H!add) ->  ( ((!authz U join) |
G !authz) & ((!authz U add) | G !authz)))))
--
```

–(LT & Mixed g-SIS authz) -> object safety property

```
LTLSPEC
    (G (!(add & remove) & !(join & leave) &
  (xjoin  -> X ((!join U leave) | G !join)) &
(xleave -> X((!leave U join) | G!leave)) &
(xremove -> X((!remove U add) | G!remove)) &
(xadd -> X((!add U remove) | G!add))) &
(!remove U add) &
(!leave U join) &

(G ( authz <->
(((!SL&!SR) S ((SA|LA) & ((!LL&!SL) S (SJ|LJ)))) |
((!SL&!SR) S (LJ& ((!SR&!LR) S LA)))))))
->
(G ((remove & (! O join)) -> ( ((!authz U join) |
G !authz) & ((!authz U add) | G !authz))))
--
```

–(LT & Overlapping Membership) -> subject safety property

```
LTLSPEC
    (G (!(add & remove) & !(join & leave) &
  (xjoin  -> X ((!join U leave) | G !join)) &
(xleave -> X((!leave U join) | G!leave)) &
(xremove -> X((!remove U add) | G!remove)) &
(xadd -> X((!add U remove) | G!add))) &
(!remove U add) &
(!leave U join) &

(G ( authz <->
(((!SL&!SR) S ((SA|LA) & ((!LL&!SL) S (SJ|LJ)))) |
((!SL&!SR) S (LJ& ((!SR&!LR) S LA)))))))

->
(G (leave -> ((H!add) -> (((!authz U join)
| G !authz) & ((!authz U add) | G !authz)))))
```

–(LT & Overlapping Membership) -> object safety property

```
LTLSPEC
    (G (!(add & remove) & !(join & leave) &
  (xjoin  -> X ((!join U leave) | G !join)) &
(xleave -> X((!leave U join) | G!leave)) &
(xremove -> X((!remove U add) | G!remove)) &
(xadd -> X((!add U remove) | G!add))) &
(!remove U add) &
(!leave U join) &

(G ( authz <->
(((!SL&!SR) S ((SA|LA) & ((!LL&!SL) S (SJ|LJ)))) |
((!SL&!SR) S (LJ& ((!SR&!LR) S LA)))))))
->
(G ((remove & (! O join)) -> ( ((!authz U join) |
G !authz) & ((!authz U add) | G !authz))))
```

[User@localhost Desktop] NuSMV -int gsis.smv
***** This is NuSMV 2.4.3 (compiled on Mon May 5 02:33:40 UTC 2008)**
***** For more information on NuSMV see <http://nusmv.irst.itc.it>**
***** or email to <nusmv-users@irst.itc.it>.**
***** Please report bugs to <nusmv@irst.itc.it>.**

NuSMV > go
NuSMV > check_ltlspec

```
-- specification (((( G (((((!(add & remove) &
!(join & leave)) & (join ->  X ((!join U leave)
|  G !join))) & (leave ->  X ((!leave U join) |
G !leave))) & (remove ->  X ((!remove U add) |
G !remove))) & (add ->  X ((!add U remove) |
G !add))) & (!remove U add)) & (!leave U join)) &
G (authz <-> (((!SL & !SR) S ((SA | LA) &
((!LL & !SL) S (SJ | LJ)))) | ((!SL & !SR) S
 (LJ & ((!SR & !LR) S LA)))))) ->  G (authz
-> ( O (add & (!leave S join)) |  O (join &
(!remove S add)))))  is true
```

NuSMV > print_reachable_states

```
############################################
```

system diameter: 1
reachable states: 512 (2^9) out of 512 (2^9)

```
############################################
```

NuSMV >

Appendix B

Stale-Safe Security Properties for Group-Based Secure Information Sharing

ABSTRACT

Attribute staleness arises due to the physical distribution of authorization information, decision and enforcement points. This is a fundamental problem in virtually any secure distributed system in which the management and representation of authorization state is not centralized. This problem is so intrinsic, it is inevitable that access control will be based on attribute values that are stale. While it may not be practical to eliminate staleness, we can *limit* unsafe access decisions made based on stale subject and object attributes. In this paper, we propose and formally specify four stale-safe security properties of varying strength which limit such incorrect access decisions. We use Linear Temporal Logic (LTL) to formalize these properties making them suitable to be verified by using model checking. We show how these properties can be applied in the specific context of group-based Secure Information Sharing (g-SIS) as defined in this paper. We specify the authorization decision/enforcement points of the g-SIS system as a Finite State Machine (FSM) and show how this FSM can be modified so as to satisfy one of the stale-safe properties. We formally verify that this FSM satisfies the stale-safe property using a mature model checker called Symbolic Model Verifier (SMV).

1. INTRODUCTION

The concept of a stale-safe security property is based on the following intuition. In a distributed system authoritative information about subject and object attributes used for access control is maintained at one or more secure authorization information points. Access control decisions are made by collecting relevant subject and object attributes at one or more authorization decision points, and are enforced at one or more authorization enforcement points. Because of the physical distribution of authorization information, decision and enforcement points, and consequent inherent network latencies, it is inevitable that access control will be based on attributes values that are stale (i.e., not the latest and freshest values). In a highly connected high-speed network these latencies may be in milliseconds, so security issues arising out of use of stale attributes can be effectively ignored. In a practical real-world network however, these latencies will more typically be in the range of seconds, minutes and even days and weeks. For example, consider a virtual private overlay network on the internet which may have intermittently disconnected components that remain disconnected for sizable time periods. In such cases, use of stale attributes for access control decisions is a real possibility and has security implications.

We believe that, in general, it is not practical to eliminate the use of stale attributes for access control decisions.[1] In a theoretical sense, some staleness is inherent in the intrinsic limit of network latencies, of the order of milliseconds in modern networks. We are more interested in situations where staleness is at a humanly meaningful scale, say minutes, hours or days. In principle, with some degree of clock synchronization amongst the authorization information, decision and enforcement points, it should be possible to determine and bound the staleness of attribute values and access control decisions. For example, a SAML assertion produced by an authorization decision point includes a statement of timeliness, i.e., start time and duration for the validity of the assertion. It is upto the access enforcement point to decide whether or not to rely on this assertion or seek a more timely one. Likewise a signed attribute certificate will have an expiry time and an access decision point can decide whether or not to seek updated revocation status from an authorization information point.

Given that the use of stale attributes is inevitable, the question is how do we safely use stale attributes for access control decisions and enforcement? The central contribution of this paper is to formalize this notion of "safe use of a stale property" in the specific context of group-based secure information sharing (g-SIS) as defined in this paper. We also demonstrate specifications of systems that provably do and do not satisfy this requirement as revealed by model checking of the specifications. We believe this formalism can be extended to more general contexts beyond the group-based secure information sharing considered in this paper, but this is beyond the current scope. We believe that the requirements for "safe use of a stale property" identified in this paper represent fundamental security properties the need for which arises in virtually any secure distributed systems in which the management and representation of authorization state is not centralized. In this sense, we suggest that we have identified and formalized a *basic security property*, in the same sense that non-interference [21] and safety [11] are basic security properties that are desirable in a wide range of secure systems.[2]

Specifically, we present formal specifications of four "stale-safe" properties. The most basic and fundamental requirement we consider deals with ensuring that while authorization data cannot be propagated instantaneously throughout the system, it is highly desirable to ensure that a requested action was definitely authorized at some point in the recent past. With staleness we may allow the authorization to hold

[1] Staleness of attributes as known to the authoritative information points due to delays in entry of real-world data is beyond the scope of this paper. For example, if an employee is dismissed there may be a lag between the time that action takes effect and when it is recorded in cyberspace. The lag we are concerned with arises when the authoritative information point knows that the employee has been dismissed but at some decision point the employee's status is still showing as active.

[2] The work of Lee et al [15, 16] is the closest to ours that we have seen in the literature, but focuses exclusively on the use of attribute certificates, called credentials, for assertion of attribute values. Lee et al focus on the need to obtain fresh information about the revocation status of credentials to avoid staleness. As we will see our formalism is based on the notion of a "refresh time," that is the time when an attribute value was known to be accurate. We believe the notion of refresh time is central to formulation of stale-safe properties. Because Lee et al admit only attribute certificates as carriers of attribute information there is no notion of refresh time in their framework.

for longer than it should have, but there is no doubt that the access was authorized in the past. Two additional requirements can be added to obtain properties that are stronger with respect to when it is required that the action be authorized. The first is that, to be permitted, it must be confirmed that a requested action is authorized at a point in time after the request and before the action is performed. The second requirement bounds the elapsed time between the point at which the authorization is confirmed and the point at which the action is performed. Because both of these requirements can be added singly or in combination, we obtain four different stale-safe properties.

We formalize these four properties in Linear Temporal Logic (LTL), making them suitable to be verified by using model checking. We show how these properties can be applied in the specific application domain of group-based secure information sharing (g-SIS). We specify one component of a g-SIS system as a finite state machine (FSM). We present two FSM's—one that does not and one that does satisfy the weakest of our state-safe properties. We formally verify the correct specification using model checking. We also use the model checker to obtain an execution trace of the incorrect FSM, which could be used by a designer to correct that FSM.

In section 2, we discuss the group-based Secure Information Sharing problem which will be used throughout the paper to illustrate the stale-safe properties. In section 3, we formalize the stale-safe security properties using Linear Temporal Logic. We specify a weak and strong version of the properties each of which is further restricted with a notion of elapsed time between the time at the which the operation is authorized and performed. In section 4, we construct a Finite State Machine (FSM) that enforces a specific policy for g-SIS. We formally verify the FSM against the weak property stated in section 3 using Model Checking. We also specify an FSM that appears to be a natural candidate for g-SIS but fails this property. In section 5, we list related work and we conclude in section 6.

2. GROUP-BASED SECURE INFORMATION SHARING (g-SIS)

Secure Information Sharing (SIS) or sharing information *while* protecting it is one of the earliest problems to be recognized in computer security, and yet remains a challenging problem to solve. A detailed discussion of SIS problem motivation and solution approaches can be found in [14]. The central problem is that copies of digital information are easily made and controls on the original typically do not carry over to the copies. One approach tried in the past has been to tie access control to each copy also so that copies are as tightly controlled as the original. The most common form of this approach is so-called mandatory or lattice-based access control [28] where copies are also labeled to reflect security sensitivity of the original. More recently, an alternate approach has emerged wherein plaintext unprotected copies are prohibited, while encrypted protected copies can be freely made. This implies that access controls need to be enforced on the client machines where the content is decrypted and displayed, so as to ensure that only authorized users get to see the content and that they are unable to make plaintext unprotected copies. There has been considerable interest in this approach, initially driven by the forces of digital rights management for entertainment content seeking to protect revenue but more generally seeking to protect content for its sensitivity.

2.1 Objectives

The group-based SIS (g-SIS) problem [14] is motivated by the need to share sensitive information amongst a group of authorized users. For simplicity we only consider the case of read access to the objects in the group. Every member of the group is authorized to read group objects. For purpose of this paper, we specify the following objectives for the g-SIS problem. For brevity, the terms subjects and objects refer to subjects and objects that belong to the group.

1. Objects are always protected (encrypted) and never exists in plain text except when viewed.

2. Objects are assumed to be available via super-distribution. This simply means that the objects are protected once and subjects may access them when authorized—objects are not individually prepared for each subject. We limit super-distribution to occur within a group.

3. Subjects can access objects off-line without involving the server using trusted access machines. The degree of trust required on the access machines may vary depending on the application and policy. In one case, the access machines may be implicitly trusted because of its physical location (e.g. access machines in an organization). In a completely distributed setting, a Trusted Reference Monitor (TRM) needs to be present on the access machines that can verify the integrity of the system and enforce the authorization policies in a trustworthy manner [27, 20]. This can be achieved using integrity measurements, remote attestation and other features enabled by Trusted Computing Technology [2] or software analogs of this technology.[3]

4. Each group has a Group Administrator (GA) who controls group membership and policies. The GA can add or remove subjects and objects from the group. We do not specify how an admin is appointed. The admin may or may not be a member of the group. Each group also has a Control Center (CC), a server that maintains authoritative subject and object attributes and provides group credentials to new members. Changes in subject and object attributes or group policies are updated by the GA at the CC and this change will eventually be propagated to the subject's access machines (discussed later in detail).

We now digress briefly to compare g-SIS with a related problem –broadcast/multicast encryption. Member management in g-SIS scenario sharply differs from Secure Internet Multicast. In multicast, as users join and leave a group, remaining members go through a re-key process thereby refreshing the group key [24]. However, for secure information sharing, such a requirement is extremely un-friendly because members need not be always connected to a server to access the objects. Thus, continuing our list of objectives, we have the following.

[3]It is generally accepted that software-only solutions will provide a lower degree of assurance than solutions with a hardware root of trust. The issues discussed in this paper are orthogonal to assurance so will apply to both software and hardware based solutions.

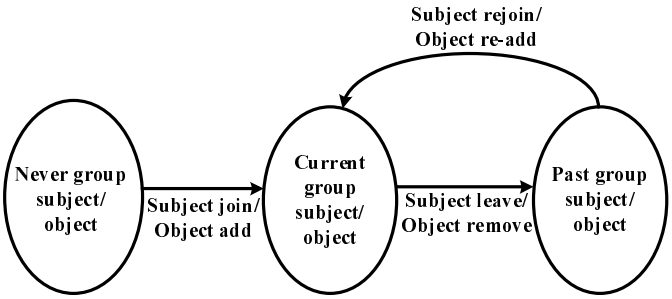

Figure 1: Subject and object membership states.

5. When a subject joins or leaves the group, remaining members should not be affected. In other words, join and leave of a subject should be completely oblivious to other subjects. Remaining subjects should not be forced to be online or go through a re-key process[4].

6. Secure multicast focuses on maintaining forward and backward secrecy of data [24]. Forward-secrecy requires that a leaving subject should not be able to read data that will subsequently be exchanged in the future. Backward-secrecy requires that a joining subject should not be able to read data exchanged amongst the remaining subjects in the past. However, *information* sharing may not be limited to forward and backward secrecy. For g-SIS flexible membership policies may be required. When a new user joins the group, whether he can access any group objects created prior to his membership is policy-dependant. Objects created after he joins the group are accessible. When a member leaves the group, whether he can continue to access objects created during his membership period is policy-dependant. However, he cannot access any object exchanged in the future within the group.

2.2 Group Management and Policy Enforcement

Subjects and objects in a group go through various states as shown in figure 1. Different access policies are possible depending on the relative state of subjects and objects . For example, a current subject could be allowed access only to current objects or also to objects created before the subject joined the group. Similarly, a past subject may lose access to all objects or retain access to objects created during his membership period. When a subject rejoins the group, he may either gain access to objects created during his past membership or simply join the group as a new subject. Similarly, many different object policies are possible. Detailed discussions can be found in [14]. Each group may thus pick a specific set of group-level access policies for subjects and objects.

[4]Members should not be asked to re-key or contact a server to get a new key. Note that re-keying is not an efficient solution in SIS as the member needs to keep track of which document was encrypted with which key. As users join and leave a group, the remaining members will need to go through a re-key process resulting in encrypting documents with different keys along the time line. One cannot discard the old key (as done in multicast) as disseminated documents encrypted with the old key continue to persist.

Figure 2 shows one possible enforcement model for the g-SIS problem and illustrates the interaction of various components in g-SIS. The Group Administrator (GA) controls group membership and policies. The Control Center (CC) is responsible for maintaining authoritative group credentials and attributes of group subjects and objects on behalf of the GA.

- *Subject Join*: Joining a group involves obtaining authorization from the GA followed by obtaining group credentials from the CC. In step 1.1, the subject contacts the GA using an access machine and requests authorization to join a group. The GA verifies that the subject is not already a member and authorizes the subject in step 1.2 (by setting AUTH to TRUE). The subject furnishes the authorization to join the group and the evidence that the access machine is in a good software state to the CC in step 1.3. The CC remotely verifies GA's authorization, if the subject's access machine is trustworthy (using the evidence) and has a known Trusted Reference Monitor (TRM) that is responsible for enforcing policies. In step 1.4, the CC provisions the attributes. sid is the Subject Id, Join_TS is the time-stamp of subject join (set to a non-NULL value), Leave_TS is the time at which a subject leaves the group (initially set to NULL), gKey is the group key using which group objects can be decrypted, Policy is the group's access policy, ORL is the Object Revocation List which lists the objects removed from the group.

- *Policy Enforcement*: From here on, the subject is considered a group member and may start accessing group objects (encrypted using the group key) as per the group policy and using the credentials obtained from the CC. This is locally mediated and enforced by the TRM. Note that the objects are available via super-distribution and because of the presence of a TRM on subject's access machines, objects may be accessed offline conforming to the policy. For example, the TRM on an access machine may allow the subject to access objects added after subject joined the group and disallow access to objects added before he/she joined the group. Such decisions can be made by using the join and leave time-stamps of subject, add and remove time-stamps of object and comparing their relative values. Objects may be added to the group by subjects by obtaining an add time-stamp (setting an Add_TS attribute for the object) from the CC. We assume object attributes are embedded in the object itself. Note that due to super-distribution, the remove time-stamps for objects cannot be embedded in the object (since there could be many copies of the same object). Instead, an Object Revocation List (with the remove time-stamps of object ids) is provisioned on the access machine.

- *Attribute Refresh*: Since subjects may access objects offline, the access machines need to connect to the CC and refresh subject attributes periodically. How this is done is a matter of policy and/or practicality. For example, a refresh could take effect in an access machine based on time or a usage count. Offline access to secure clock may be impractical in many circumstances. Usage count is a practical approach when using Trusted Computing Technology. A discussion on

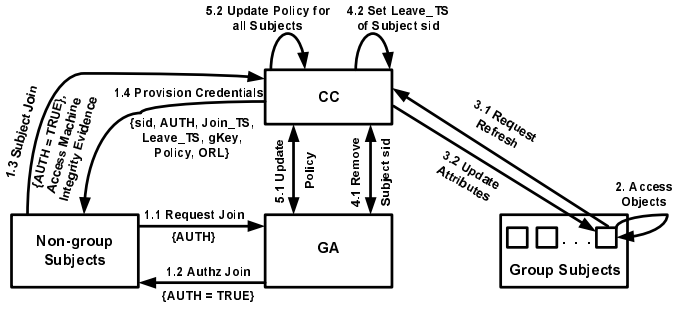

Figure 2: g-SIS System.

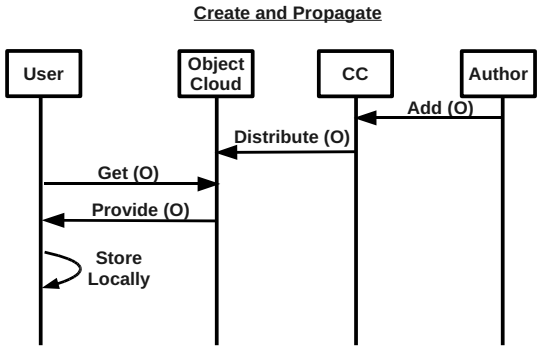

Figure 3: Super-distribution.

using the monotonic counters in the Trusted Platform Module can be found in [30]. The usage count limits the number of times the credentials may be used to access group objects (like consumable rights). Thus objects may be accessed until the usage count is exhausted and the access machine will be required to refresh attributes in step 3.1 and 3.2 before any further access can be granted. Attributes RT and N represent the refresh time-stamp and usage count of the subject respectively.

- *Administrative Actions*: The GA may have to remove a subject or object from the group or update group policy. In step 4.1, the GA instructs the CC to remove a subject. The CC in turn marks the subject for removal by setting the subject's Leave_TS attribute in step 4.2. This attribute update is communicated to the subject's access machine during the refresh step 3.1 and 3.2. In the case of object removal, the ORL is updated with the object's id and Remove_TS. Policy updates (or any other update for that matter) are handled in a similar manner as shown in step 5.1 and 5.2.

As you can see, there is a delay in attribute update in the access machine that is defined by the refresh window. Although a subject may be removed from the group at the CC, the access machines will let subjects access group objects until the subject attributes are refreshed at the next refresh step. This access violation is due to attribute staleness that is inherent to any distributed system however short the refresh window is. We discuss this topic in detail in the subsequent sections. This paper does not focus on building trusted systems to realize the architecture in figure 2 and it is a work in progress. This is an area of active work and we direct interested readers to [27] and [20], to cite a few.

3. STALE SAFE SECURITY PROPERTIES FOR g-SIS

As discussed earlier, in distributed systems access decisions are almost always based on stale-attributes and stale-attributes lead to critical access violations. In this section we propose Stale-safe Security Properties that limit such access violations. Note that it is impossible to completely eliminate staleness in practice and thus our intension here is best effort. We first discuss a few scenarios were stale attributes lead to access violations using the g-SIS example and informally discuss the stale-safe properties. We formalize them next.

3.1 System Characterization

The g-SIS system consists of subjects and objects, trusted access machines (using which objects are accessed), a GA and a CC. Access machines maintain a local copy of subject attributes which they refresh periodically with the CC. Object attributes are part of the object itself. A removed object is listed in the Object Revocation List (ORL), which are provided to access machines as part of refresh. To easily illustrate the properties, we assume that each subject is tied to an access machine from which objects are accessed and there is a single GA and single CC per group. Also, we assume that the refresh is based on usage count. Suppose a policy that a subject is allowed to access an object as long as both the subject and object are current members of the group and the object was added after the subject joined the group. Thus the g-SIS system can be characterized as follows:

Subject attributes	$\{\text{id}, \text{Join\_TS}, \text{Leave\_TS}, \text{ORL}, \text{gKey}, \text{RT}, \text{N}\}$
Object attributes	$\{\text{id}, \text{Add\_TS}\}$.
Refresh Time (RT)	Access machine contacts CC to refresh subject attributes and ORL.
Refresh Window (RW)	Time interval between two RT's (depends on how quick the usage count is exhausted).
Access Policy	$\text{Authz}_\text{P}(S, O, OP) \rightarrow O \notin \text{ORL}(S) \wedge$ $\text{Leave\_TS}(S) = \text{NULL} \wedge$ $\text{Join\_TS}(S) \leq \text{Add\_TS}(O)$.

Figure 3 illustrates super-distribution. An Author (a group subject) creates an object, encrypts the object using the group key (mediated by TRM) and sends it to the CC for approval and distribution. The CC (or possibly a GA) approves the object, time-stamps object add and releases this protected object for distribution. Since the object is protected, it is not necessarily guarded by the CC. Instead it is made available to subjects by distribution through networks such as WWW, email, etc. This infospace is called the Object Cloud in figure 3. The User (another group subject) can obtain these encrypted objects and store them locally in his/her access machine. The sequence diagram in Figure 4 illustrates the staleness problem. The User and the TRM interacts with the GA and CC in steps 1 to 5 to join the group. The TRM refreshes attributes with the CC in steps 6 and 7. Briefly after the refresh, the GA removes this subject by setting his/her Leave_TS attribute at the CC (a

non-null value). Note that this step is not visible to the TRM until the next refresh steps 11 and 12. In the mean time, the User may request access to objects the were obtained via super-distribution (step 9). "Create and Propagate" refers to the scenario in figure 3. At this point, the TRM evaluates the policy based on the attributes that it maintains. This should be successful and the object is displayed to the user in step 10. Note the difference in Leave_TS values between the CC and TRM. Only after the following refresh (steps 11 and 12) does the TRM notice that the subject has been removed from the group and denies any further access (steps 13 and 14).

Figure 5 shows a timeline of events involving a single group. Subject $S1$ joins the group and the attributes are refreshed with the CC periodically. RT represents the time at which refreshes happen. The time period between any two RT's is a Refresh Window, denoted RW_i. After join, RW_0 is the first window, RW_1 is the next and so on. Suppose RW_4 is the current Refresh Window. Objects $O1$ and $O2$ were added to the group by *some* group subject (or the GA) during RW_2 and RW_4 respectively and they are available to $S1$ via super-distribution. In RW_4, $S1$ requests access to $O1$ and $O2$. An access decision will be made by the TRM in the access machine as per the attributes obtained at the latest RT.

As you can see, our access policy will allow access to both $O1$ and $O2$. However it is possible that S1 was removed by the GA right after the last RT and before $Request(S1, O1, access)$ in RW_4 (see figure 4). Ideally, S1 should not be allowed to access both $O1$ and $O2$.

From a confidentiality perspective in information sharing, granting $S1$ access to $O1$ is relatively less of a problem than granting access to $O2$. This is because the CC or the GA can assume that $S1$ was always authorized access to $O1$ and hence information has already been released to $S1$. In the worst case, $S1$ continues to access the same information ($O1$) until the next RT. However, $S1$ never had an authorization to access $O2$ and letting $S1$ access $O2$ means that $S1$ has gained knowledge of new information. This is a critical violation and should not be allowed. Such scenarios are what our stale-safe security properties address. A subject cannot access an object if it was added to the group after the last refresh time even if the authorization policy allows access. This can be achieved by comparing the Add_TS (O) with the most recent refresh time-stamp ($\text{RT}_{\text{recent}}$). Thus the access decision for a stale-safe g-SIS system should be made as follows:

Access Policy	$\text{Authz}_{\text{P}}(S, O, OP) \rightarrow O \notin \text{ORL}(S) \wedge$ $\text{Leave\_TS}(S) = \text{NULL} \wedge$ $\text{Join\_TS}(S) \leq \text{Add\_TS}(O)$.
Stale-safe Property	$\text{SafeP}\ (S, O) \rightarrow \text{Add\_TS}\ (O) < \text{RT}_{\text{recent}}$
Stale-safe Access Policy	$\text{Authz}_{\text{P}}(S, O, OP) \wedge \text{SafeP}(S, O)$

The property we discussed considers attributes to be stale if it is time-stamped later than the last refresh time-stamp of the access machine. A more strict property may require the access machine to refresh attributes before granting any access. That is, when $S1$ requests access to $O1$, the stricter version of the stale-safe property mandates that the access machine refreshes the subject attributes before making an authorization decision. Further, it is natural to consider

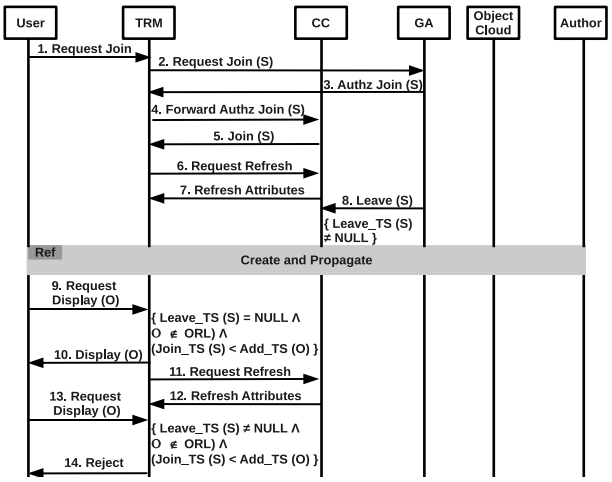

Figure 4: Staleness Illustration.

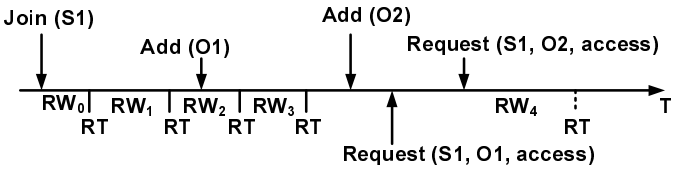

Figure 5: Events on a time line to illustrate stale-safe properties in g-SIS.

elapsed time since the last refresh to be an important issue in limiting staleness of authorization data. We formalize these notions in the following subsection.

3.2 Formal Property Specification

In this section we use Linear Temporal Logic (LTL) [18] to specify four different formal stale-safety properties of varying strength. Temporal logic is a specification language for expressing properties related to a sequence of states in terms of temporal logic operators and logic connectives (e.g., $\wedge$ and $\vee$). Temporal logic operators are of two types: Past and Future. The past operators $\ominus$ and Since (read previous and since respectively) have the following semantics. $\ominus\ p$ means that the formula p was true in the previous state. Note that $\ominus\ p$ is false in the very first state. p Since q means that q has happened sometime in the past and p has held continuously following the last occurrence of q to the present. The future operators $\bigcirc$, $\diamond$, and $\Box$ represent next state, some future state, and all future states respectively. For example, $\Box p$ means that formula p is true in all future states. Also, the formula p *until* q (read p until q) means that q will occur sometime in the future and p will remain true at least until the first occurrence of q.

Our formalization uses the following predicates:

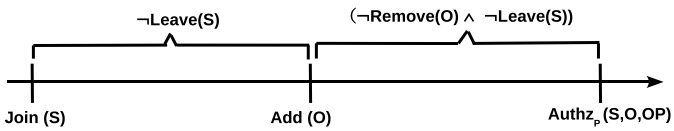

Figure 6: Access Policy (Authz_P).

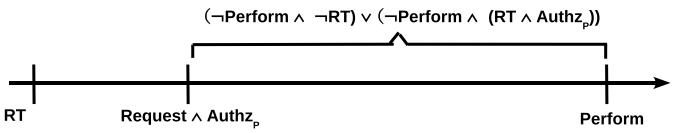

Figure 7: Formula φ_0.

Figure 8: Formula φ_1.

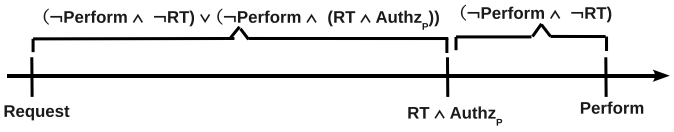

Figure 9: Formula φ_2.

request (S, O, OP)	The subject requests to perform an action OP on an object.
Authz_P (S, O, OP)	S is authorized to perform an action OP on O.
Join (S) and Leave (S)	Subject Join and Leave events.
Add (O) and Remove (O)	Object Add and Remove events.
perform (S, O, OP)	S performs an action OP on O in the current state.
RT (S)	The TRM contacts the CC to update subject attributes.

In the forthcoming formulae (φ_0, φ_1 and φ_2) and throughout this paper, we drop the corresponding parameters S, O and OP in these predicates for clarity. They should however be interpreted with the respective semantics described above.

3.2.1 Stale-unsafe Access Decision

We first formalize a stale-unsafe access decision using the access policy discussed in section 3.1 as an example. Authz_P below is the same policy represented using LTL. Formula φ_0 formalizes an access decision that is stale-unsafe.

$$\begin{aligned}\text{Authz}_\text{P} &\equiv (\neg\text{Remove} \wedge \neg\text{Leave}) \text{ Since } ((\text{Add} \wedge \\ &\quad \neg\text{Leave}) \text{ Since Join}) \\ \varphi_0 &\equiv \ominus\, (\neg\text{perform} \wedge (\neg\text{RT} \vee (\text{RT} \wedge \text{Authz}_\text{P}))) \\ &\quad \text{Since } (\text{request} \wedge \text{Authz}_\text{P})\end{aligned}$$

Figure 6 illustrates Authz_P. Authz_P says that S is allowed to perform an action OP on O if prior to the current state the object was added to the group and both the subject and object have not left the group since. Also, the subject joined the group prior to the time the at which the object was added to the group and has not left the group ever since.

Figure 7 illustrates formula φ_0. φ_0 says that the operation was authorized at the time of request. Prior to the current state, the operation has not been performed since it was requested. Also since it was requested, any refreshes that may have occurred indicated that the operation was authorized $(\neg\text{RT} \vee (\text{RT} \wedge \text{Authz}_\text{P}))$.

Definition 3.1 (Staleness Unaware). *A Finite State Machine (FSM) is* staleness unaware *if it satisfies the following LTL formula:*

$$\Box(\text{perform} \rightarrow \varphi_0)$$

Observe that in a *Staleness Unaware* FSM, verifying that Authz_P holds at the time of request will allow the subject access objects that were added during the time between RT and $(\text{request} \wedge \text{Authz}_\text{P})$ in figure 7. This can be clearly seen by converging Authz_P in figure 6 with that in figure 7. As discussed earlier, it is unsafe to let group subjects access these objects before a refresh can confirm the validity of their group membership.

We now specify stale-safe security properties of varying strength. The weakest of the properties we specify requires that a requested action be performed only if a refresh of subject attributes and ORLs has shown that the action was authorized at that time. This refresh is permitted to have taken place either before or after the request was made. The last refresh must have indicated that the action was authorized and all refreshes performed since the request, if any, must also have indicated the action was authorized. This is the *weak stale-safe security property*. By contrast, the *strong stale-safe security property* requires that the confirmation of authorization occur after the request and before the action is performed.

3.2.2 Weak Stale-safe Security Property

Let us introduce two formulas formalizing pieces of stale-safe security properties. Intuitively, φ_1 can be satisfied only if authorization was confirmed prior to the request being made. On the other hand, φ_2 can be satisfied only if authorization was confirmed after the request. Note that weak stale safety is satisfied if either of these is satisfied prior to a requested action being performed.

$$\begin{aligned}\varphi_1 &\equiv \ominus\, (\neg\text{perform} \wedge (\neg\text{RT} \vee (\text{RT} \wedge \text{Authz}_\text{P}))) \\ &\quad \text{Since } (\text{request} \wedge (\neg\text{RT Since } (\text{RT} \wedge \text{Authz}_\text{P}))) \\ \varphi_2 &\equiv \ominus\, (\neg\text{perform} \wedge \neg\text{RT}) \text{ Since } (\text{RT} \wedge \text{Authz}_\text{P} \wedge \\ &\quad ((\neg\text{perform} \wedge (\neg\text{RT} \vee (\text{RT} \wedge \text{Authz}_\text{P}))) \text{ Since request}))\end{aligned}$$

Figure 8 illustrates formula φ_1. φ_1 says that prior to the current state, the operation has not been performed since it was requested. Also since it was requested, any refreshes that may have occurred indicated that the operation was authorized $(\neg\text{RT} \vee (\text{RT} \wedge \text{Authz}_\text{P}))$. Finally, a refresh must have occurred prior to the request and the last time a refresh was performed prior to the request, the operation was authorized.

Observe that formula φ_1 mainly differs from φ_0 on the point at which Authz_P is evaluated. Referring to figure 8, evaluating Authz_P at the latest RT guarantees that requests

to access any object that may be added during the following refresh window will be denied.

Note that φ_1 is satisfied if there is no refresh between the request and the perform. It requires that any refresh that happens to occur during that interval indicate that the action remains authorized. In our g-SIS application, this could preclude an action being performed, for instance, if the subject leaves the group, a refresh occurs, indicating that the action is not authorized, the subject rejoins the group, and another refresh indicates that the action is again authorized. For some applications, this might be considered unnecessarily strict.

Figure 9 illustrates formula φ_2. φ_2 does not require that there was a refresh prior to the request. Instead it requires that a refresh occurred between the request and now. It further requires that the operation has not been performed since it was requested and that every time a refresh has occurred since the request, the operation was authorized.

Note that φ_2 can be satisfied without an authorizing refresh having occurred prior to the request, whereas φ_1 cannot. Thus, though φ_2 ensures fresher information is used to make access decisions, it does not logically entail φ_1 as it is satisfied by traces that do not satisfy φ_1.

We call perform $\rightarrow \varphi_1$ *backward-looking stale safety*, as it does not require that a confirmation of authorization occur after the request has been received. We call perform $\rightarrow \varphi_2$ *forward-looking stale safety*, as it requires that confirmation of authorization is obtained after the request, before the action is performed.

Definition 3.2 (Weak stale safety). *An FSM has the* weak stale-safe security property *if it satisfies the following LTL formula:*

$$\Box(\text{perform} \rightarrow (\varphi_1 \vee \varphi_2))$$

3.2.3 Strong Stale-safe Security Property

Forward-looking stale safety is strictly stronger than weak stale safety. For this reason, and because, unlike backward-looking stale safety, it is a reasonable requirement for controlling many operations, we give it a second name.

Definition 3.3 (Strong stale safety). *An FSM has the* strong stale-safe security property *if it satisfies the following LTL formula:*

$$\Box(\text{perform} \rightarrow \varphi_2)$$

3.2.4 Quantifying "Freshness" of Authorization

Let us now consider how to model requirements that constrain the actual time at which actions such as attribute refresh occur. For this we introduce a sequence of propositions $\{P_i\}_{0 \leq i \leq n}$ that model n time intervals (owing to the propositional nature of LTL, we can model only a finite number of time intervals.). These propositions partition each trace into contiguous state subsequences that lie within a single time interval, with each proposition becoming true immediately when its predecessor becomes false. They can be axiomatized as follows:

$$\begin{aligned} &P_1 \text{ Until } (\Box\neg P_1 \wedge \\ &(P_2 \text{ Until } (\Box\neg P_2 \wedge \\ &(P_3 \text{ Until } (... \\ &\quad \text{Until } (\Box\neg P_{n-1} \wedge \Box P_n)...))))) \end{aligned}$$

We now formulate variants of φ_1 and φ_2 that take a parameter k indexing the current time interval. These formulas use two constants, ℓ_1 and ℓ_2 which represent the number of time intervals since the authorization and the request, respectively, that is considered acceptable to elapse prior to performing the requested action. The formulas prohibit performing the action if either the authorization or the request occurred further in the past than permitted by these constants.

$$\begin{aligned} \varphi_1(k) \equiv \ominus\, &(\neg\text{perform} \wedge (\neg\text{RT} \vee (\text{RT} \wedge \text{Authz}_\text{P}))) \text{ Since} \\ &(\text{request} \wedge \bigvee_{max(0,k-\ell_2) \leq i \leq k} P_i \wedge \\ &(\neg\text{RT Since } (\text{RT} \wedge \text{Authz}_\text{P} \wedge \bigvee_{max(0,k-\ell_1) \leq i \leq k} P_i)))) \\ \varphi_2(k) \equiv \ominus\, &(\neg\text{perform} \wedge \neg\text{RT}) \text{ Since} \\ &(\text{RT} \wedge \text{Authz}_\text{P} \wedge \bigvee_{max(0,k-\ell_1) \leq i \leq k} P_i \wedge \\ &((\neg\text{perform} \wedge (\neg\text{RT} \vee (\text{RT} \wedge \text{Authz}_\text{P}))) \text{ Since} \\ &(\text{request} \wedge \bigvee_{max(0,k-\ell_2) \leq i \leq k} P_i)) \end{aligned}$$

With these formulas, we are now able to state variants of weak and strong stale safety that require timeliness, as defined by the parameters ℓ_1 and ℓ_2.

Definition 3.4 (Timely, weak stale safety). *An FSM has the* timely, weak stale-safe security property *if it satisfies the following LTL formula:*

$$\Box(\bigwedge_{0 \leq k \leq n} (\text{perform} \wedge P_k) \rightarrow (\varphi_1(k) \vee \varphi_2(k)))$$

Definition 3.5 (Timely, strong stale safety). *An FSM has the* timely, strong stale-safe security property *if it satisfies the following LTL formula:*

$$\Box(\bigwedge_{0 \leq k \leq n} (\text{perform} \wedge P_k) \rightarrow \varphi_2(k))$$

3.3 Stale-safe Systems

We discuss the significance of the weak and strong stale-safe properties in the context of stale-safe systems designed for confidentiality or integrity. Confidentiality is concerned about information release while integrity is concerned about information modification. Both weak and strong properties are applicable to confidentiality –the main trade-off between weak and strong here is usability. Weak allows subjects to read objects when they are off-line while strong forces subjects to refresh attributes with the server before access can be granted. Depending on the security and functional requirements of the system under consideration, the designer has the flexibility to choose between weak and strong to achieve stale-safety. In the case of integrity, the weak property can be risky in many circumstances –the strong property is more desirable. This is because objects modified by unauthorized subjects may be used/consumed by other subjects before the modification can be undone by the server. For instance, in g-SIS, a malicious unauthorized subject (i.e. a malicious subject who has been revoked group membership but is still allowed to modify objects for a time period due to stale attributes) may inject bad code and share it with the

group. Other unsuspecting subjects who may have the privilege to execute this code may do so and cause significant damage. In another scenario, a malicious subject may inject incorrect information into the group and other subjects may perform certain critical actions based on faulty information. Thus, although both weak and strong properties may be applicable to confidentiality and integrity, the weak property should be used with a caveat in the case of integrity.

3.4 Extensions

In the earlier section, we discussed the most fundamental stale-safe properties. We now consider stale-safety in the context of a truly distributed system with multiple access machines for a subject, multiple CC'S and GA's and multiple group memberships of subjects and objects. As you can see, this is a broad and complex problem that requires in-depth research and is beyond the scope of this paper. However, we informally articulate the staleness problem and desirable properties in such a scenario.

- *Multiple Access Machines*: Consider a scenario where a group subject may access group objects from multiple access machines. Each machine maintains a local set of attributes and they are not only stale with respect to the CC but also with respect to other access machines. If not careful, a subject may maintain various membership states across multiple access machines. For example, the same subject could be current member in one machine, but could leave the group and rejoin from another machine. As per our access policy earlier, the subject ends up accessing two sets of objects from two different machines –one as a current member and the other as a rejoined member at the same time[5]. Such violations may be serious in the context of mutual exclusion. Thus the stale-safe property for multiple access machines should make sure that the subject's membership state is consistent across all the machines. When a subject attempts to rejoin a group from one access machine, the CC should instruct the TRM's on subject's other access machines to revoke all access until the subject leaves and rejoins on all machines.

- *Multiple CCs*: Recall that the GA updates subject attributes at the CC which in turn is updated at the subject's access machine during a refresh. This property deals with attribute staleness with respect to GA's updates in the case of multiple CCs. A GA may update attributes at one CC and the attributes at other CCs remain stale until this update is propagated across all the CCs. In such a scenario, a refresh from one of the CCs by the access machine may turn out to be stale. The stale-safe property for multiple CCs should make sure that the access machine will be allowed to update attributes only if the attributes at the CC are more recent than that of the access machine. Suppose the CC maintains a subject attribute LU_TS that is the time-stamp of the last update received from the GA for that subject. And the access machine maintains an attribute LSR_TS that is the time-stamp of the lastest refresh when that refresh actually resulted in a real attribute update. Then an access machine can refresh subject attributes only if the LU_TS (S) > LSR_TS (S).

- *Multiple Groups (non-hierarchical)*: In the case of multiple groups, an object from one group could be shared with another. We call the group that owns the object source group. If the object is removed from the source group, attribute staleness could let other group subjects retain access.
  A. If an object is removed from the source group, it should also be removed from other groups with which the object is shared.
  B. If a subject is revoked access to an object from one group, he should also be revoked access to that object from any other membership group with which it is shared.

- *Multiple Groups (hierarchical)*: Consider hierarchical groups where subjects in higher level groups have access to objects at or below its hierarchical level. Thus the subjects in the leaf groups have access only to a single group and subjects at the root groups have access to objects of all group in the hierarchy. Suppose that each level has its own CC. In order to limit staleness, an access policy should use the appropriate subject and object attributes from respective groups in question:
  A. An access decision for the subject should be made based on the object attributes from the source group and subject attributes from the subject's highest hierarchical membership group.
  B. When a subject leaves a group and joins any group at the lower level of the hierarchy, he/she should be revoked access to any object that he/she retains access from past group[6].

4. MODEL CHECKING g-SIS

Model checking [6, 7] is an automated verification technique that analyzes a finite model of a system (i.e., a finite state machine (FSM) that produces computation traces consisting of infinite sequences of states) and exhaustively explores the state space of the model to determine whether desired properties hold in the model. In the case that a property is false, a model checker produces a counterexample consisting of a trace that violates the property, which can be used to correct the model or modify the property specification.

SMV [22, 5] is a family of model checking tools based on binary decision diagrams (BDDs). BDDs represent states very compactly. In SMV, models are represented by using variables that are assigned values in each step of the FSM. Properties to be checked are specified by temporal logic [23] formulas. SMV provides built-in finite data types, such as boolean, enumerated type, integer range, arrays, and bit vectors. In SMV, the initial state is defined by assigning initial values to state variables. State transitions are specified by assigning values to be assumed by each state variable x in the next state, which is denoted by next(x). Each such value is given by an expression over variables in either the current or the next state. The assignments are effectively performed simultaneously to obtain the subsequent state.

[5]Note that this may not be a problem if the policy lets subjects retain access to past objects.

[6]Note that this property may not be applicable to all group policies.

SMV allows nondeterministic assignment, i.e., the value of variable is chosen arbitrarily from the set of possible values.

The set of next assignments execute concurrently in a step to determine the next state of the model. SMV allows nondeterministic assignment, i.e., the value of variable is chosen arbitrarily from the set of possible values. SMV supports macros, which are replaced by their definitions, so they do not increase the system's state space.

The model checker we use in this work supports only future temporal operators ("in the next state," "in all future states," "in some future state," and "until"), so the formulas expressing stale safety in section 3.2 have to be reformulated in this restricted form.

In this section, we use model checking (with SMV) to verify the weak stale-safe property for g-SIS. Model checking the entire g-SIS system with CCs, GAs and multiple groups is out of scope for this paper (please see discussions in Future Work). Instead, we model check the Trusted Reference Monitor (TRM) that is responsible for enforcing the access policies in the subject's access machine. Please see the Appendix for code, the properties that are verified and the results obtained from SMV.

4.1 Formal Verification of the Trusted Reference Monitor

In modeling the TRM, one of the first things to decide is how a refresh of subject attributes is forced so that the TRM periodically updates attributes with CC. Recall that we discussed various approaches: refresh based on timeout, usage count, etc. We could further use rate limits or a combination of these approaches. The TRM we consider models refresh based on usage count[7]. The CC/GA determines a usage count for each subject that specifies the number of times the group credentials (e.g. group key) may be used by the TRM to access objects off-line before a refresh is required. Suppose N is the usage count. Every time a subject accesses an object, N is decremented by the TRM. Once N reaches zero for that subject, the TRM denies access to any object until the attributes are refreshed by the access machine with the CC. As part of this refresh, N is reset to the initial value.

The TRM includes one FSM for each object available at the access machine. We discuss the formal verification of an object machine, $\text{FSM}_{\text{object}}$, against the Weak Stale Safety property (Definition 3.2). Figure 10 shows one possible design of $\text{FSM}_{\text{object}}$ to enforce an authorization policy given by Authz_{E}. Staleness is not considered in this machine. The predicate Authz_{E} is $(\neg\text{Remove\_TS}(O) \wedge \neg\text{Leave\_TS}(S) \wedge (\text{Join\_TS}(S) \leq \text{Add\_TS}(O)))$, indicating that the object O has not been removed from the group, subject S has not left the group, and the subject S joined the group before the object O was added. This is the same as the LTL formula Authz_{P} discussed in section 3 except that we now use attributes that can be directly coded in SMV[8]. We label state transitions using the format $e[C]/A$, in which e is the event, C is the condition that has to be satisfied to enable the transition, and A represents actions that need to be performed when the transition is taken.

The $\text{FSM}_{\text{object}}$ is responsible for mediating request from the subject to access the object to which it corresponds. It remains in the idle state until a request to access the object arrives from the subject. At this point, $\text{FSM}_{\text{object}}$ checks the authorization policy (Authz_{E}) to decide whether the subject can access the requested object. There are then three possible paths the FSM can take, depending on which condition is satisfied:

Request$[\neg\text{Authz}_{\text{E}}]$: If Authz_{E} fails, $\text{FSM}_{\text{object}}$ rejects the request and remains in the idle state. This is the request transition that starts and ends at the idle state. The Refresh transition captures attribute updates received from the CC triggered by other instances of $\text{FSM}_{\text{object}}$ running on behalf of the same subject.

Request$[\text{Authz}_{\text{E}} \wedge N = 0]$: If Authz_{E} succeeds, but the usage count is exhausted, the machine is required to refresh attributes before any access can be granted. A refresh request action ($\text{Refresh}_{\text{REQ}}$) is initiated in this case. This creates a synchronizing event (transitions labeled Refresh in the idle, authorized and refreshed states) for all the $\text{FSM}_{\text{object}}$ instances in the local TRM, which has the effect of updating all subject attributes, as well as the ORL, with the values that are current at the CC. The synchronous event simply ensures that the update is atomic with respect to transitions at every $\text{FSM}_{\text{object}}$. After the refresh, the $\text{FSM}_{\text{object}}$ then enters the refreshed state. It again checks the authorization policy to see whether the subject is now allowed the requested access in light of the updated attribute values. If Authz_{E} holds, the FSM directly enters the authorized state from which it transitions to idle while decrementing the usage count. If Authz_{E} does not hold, the FSM denies access and immediately returns to idle.

Request$[\text{Authz}_{\text{E}} \wedge N > 0]$: If Authz_{E} succeeds and the usage count N is not exhausted, the machine enters the authorized state and waits for the subject to access the object. The requested action is performed only after re-checking the policy Authz_{E} and decrementing the usage count. This re-checking is critical because Authz_{E} checked earlier may no more hold due to updated attributes received from the Refresh transition which could possibly be triggered by another instance of $\text{FSM}_{\text{object}}$. We discuss this in more detail in the following paragraph. $\text{FSM}_{\text{object}}$ thereafter returns to the idle state.

Consider the self-transitions labeled Refresh in idle, authorized and refreshed states. It is needed to allow refreshes initiated by other $\text{FSM}_{\text{object}}$'s to occur atomically with respect to other transitions. A subject could request access to multiple objects; a separate instance of $\text{FSM}_{\text{object}}$ runs for each such object. A problem may arise due to a possible lag between the time at which the access was authorized (after which the $\text{FSM}_{\text{object}}$ is in the authorized state) and the time at which the subject actually performs the access. Suppose $S1$ is allowed to access object $O1$ and that $O1$'s $\text{FSM}_{\text{object}}$ enters and remains in the authorized state until $S1$ performs the action. In the meantime, $S1$ requests and performs access on multiple objects and exhausts the usage count. Finally when an access request for an object $O2$ is initiated, the $\text{FSM}_{\text{object}}$ of which forces a refresh because the usage count ran out. This illustrates that fact that multiple refreshes can occur

[7]This is chosen due to the lack of availability of any practical solution for a secure off-line source of time today. The Trusted Platform Module [2] provides a monotonic counter and hence we choose to develop and model check a usage count based TRM that can be later implemented. Our discussions remain valid irrespective of the approach we take.

[8]Note that the check for the presence of the object in ORL is simplified and simulated as an event in SMV. This event results in setting the boolean attribute $\text{Remove\_TS}(O)$ to TRUE.

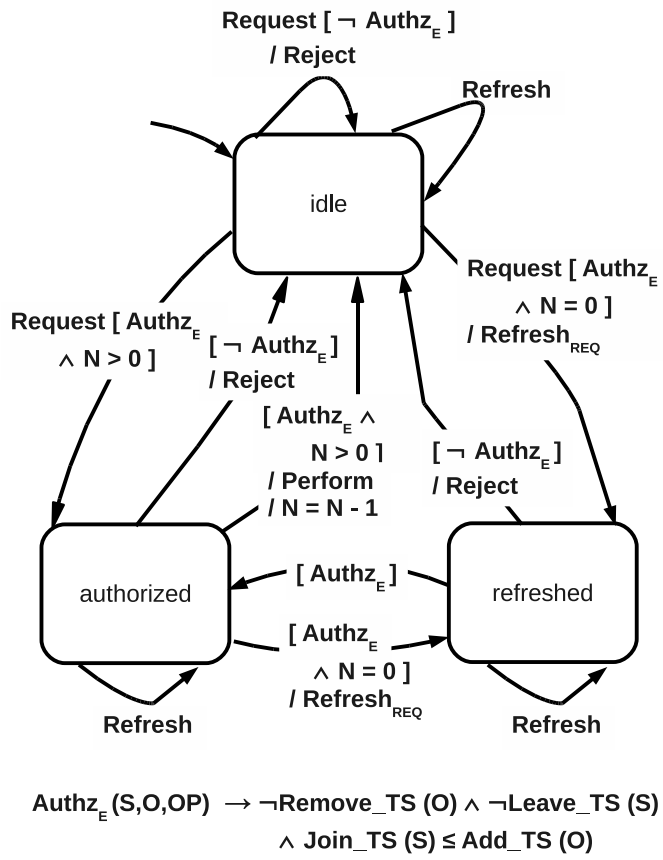

Figure 10: Stale-*unsafe* $\mathrm{FSM}_{\mathrm{object}}$.

when $\mathrm{FSM}_{\mathrm{object}}$ is in the authorized state for $O1$. When $S1$ actually performs the access of $O1$, it may no longer be authorized due to updated attributes. Such refreshes are permitted by the self-transition from the authorized state back to itself. When a refresh response is received by any instance of $\mathrm{FSM}_{\mathrm{object}}$, it is broadcasted to all other instances, and the attributes are updated in every machine. In this way, when a perform is generated, $\mathrm{Authz}_{\mathrm{E}}$ uses the latest attributes to verify policy. Thus when the $\mathrm{FSM}_{\mathrm{object}}$ instance for $O2$ has updated attributes, it sends those updated attribute values to all running instances of $\mathrm{FSM}_{\mathrm{object}}$. Consequently, when $S1$ performs the action, it may not be correct to allow access to $O1$, due to updated attributes and failed $\mathrm{Authz}_{\mathrm{E}}$. Thus checking the policy $\mathrm{Authz}_{\mathrm{E}}$ at both request and perform time is critical for the correctness of $\mathrm{FSM}_{\mathrm{object}}$.

Code listing B in appendix A.2 shows the construction of $\mathrm{FSM}_{\mathrm{object}}$ using SMV and properties that are verified against it. As shown, the backward-looking stale safety property (formula φ_1, section 3.2) does not hold. The model checker immediately detects and reports the problem with a counter example. Note that the property is re-formulated using future temporal operators only which can be model checked using SMV. The $\mathrm{FSM}_{\mathrm{object}}$ in figure 10 is not stale-safe because it allows access to objects that were added after the last refresh time. This problem is fixed in figure 11 which we discuss in the following subsection.

4.1.1 Stale-safe TRM

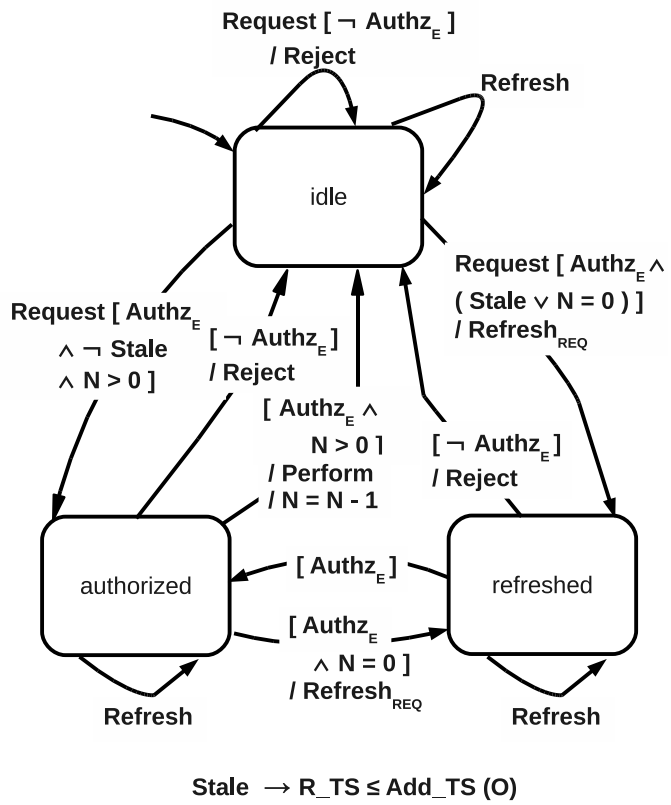

Figure 11: Stale-*safe* $\mathrm{FSM}_{\mathrm{object}}$.

Figure 11 is one of many approaches to build a stale-safe $\mathrm{FSM}_{\mathrm{object}}$. As you can see, the only difference from figure 10 is the extra check for stale attributes (in our case *Stale* is $\mathrm{RT\_TS} \leq \mathrm{Add\_TS}(O)$). The transition from idle to authorized is enabled if the authorization policy succeeds, the usage count is still available *and* the attributes are not stale (i.e., $\mathrm{Add\_TS}(O) < \mathrm{RT\_TS}$). The transition from idle to refreshed is enabled if the authorization policy is successful but either the attributes are stale or the offline usage limit is reached. From refreshed, $\mathrm{FSM}_{\mathrm{object}}$ enters authorized state if $\mathrm{Authz}_{\mathrm{E}}$ still holds with the refreshed attributes else it returns to idle. Thus this machine satisfies formula φ_1 (backward-looking stale-safety) when the attributes are not stale, and it satisfies formula φ_2 (forward-looking stale-safety) when the attributes are stale.

Code Listing A in appendix A.2 shows an SMV implementation of this machine. The result for this machine, which we have verified by using the model checker, is discussed in appendix A.1. It is also possible to construct machines that will satisfy the other properties we discussed earlier.

5. RELATED WORK

Security requirements express the goals for protecting the confidentiality, integrity, and availability of cyber systems. There has been substantial work on developing models and policy languages for addressing these security concerns. Access control lies at the heart of system security [29]. Formal specification and verification techniques and tools, such as model checking, have been increasingly leveraged to verify security properties of access control systems [8, 13, 32, 3, 9,

19, 31, 34]

Zhang et. al. [34] developed a model checking approach to examine the access right of a group of principles. The access control is modeled in the RW language, which is a propositional logic-based policy language to express reading and writing access [10]. May et. al. [19] formalize the rules of Health Insurance Portability and Accountability Act into an extended access control matrix, which can be analyzed by model checker SPIN.

Security analysts of access control systems and policies have increasingly leveraged automated tool support to verify properties in support of security objectives. Jha et al. [13] verify such properties as authorization, availability, and shared access of the SPKI/SDSI policy language through the use of a language containment type of model checking.

Sistla et. al. [32] provides a framework for reasoning about security analysis of dynamic RT policies. Of significant value is their proof of a tight EXPTIME complexity for role containment queries. Additionally, they describe a structure to verify security properties using an explicit model checking approach.

Fisler et al. [8] analyzes the impact of policy changes on role-based access control (RBAC) systems using their Margrave tool. Such policies are represented as multi-terminal BDD's for efficient storage and manipulation. They successfully verify the separation of duty properties in RBAC system.

Schaad et al. [31] also verifies separation of duty properties in RBAC systems, but uses a mature model checking tool called NuSMV.

In addition, security analysis that answers the question whether security stake-holders can cause the authorization system to enter a state, in which certain queries (e.g., safety or liveness properties) hold or fail to hold, has been automatically performed [12, 17, 33, 25, 26], via the SMV family of model checkers.

6. CONCLUSIONS AND FUTURE WORK

Attribute staleness is inherent to any distributed system and can result in serious access violations. In this paper, we proposed stale-safe security properties using the group-based Secure Information Sharing problem as an example. We formalized four stale-safe properties of varying strengths using Linear Temporal Logic amenable to formal verification using Model Checking. Model Checking is a powerful and flexible approach to verify security properties of large and complex systems such as g-SIS. We designed and verified the Trusted Reference Monitor resident in access machines that satisfies the weak stale-safe property. We believe that these properties can be generalized to any distributed applications using Attribute-based Access Control with minor extensions/modifications if any. Our next steps are along three exciting areas:

In section 3, we identified staleness problem in the context of multiple CCs, multiple groups and multiple access machines and proposed extensions. We believe studying and formalizing these extensions is valuable to build systems with flexible stale-safe properties.

Verifying the complete g-SIS system is a major future work. This is a complex problem which is composed of multiple FSMs for TRM, CC and GA. All these machines need to handle various operations such as membership management of subjects and objects, provisioning group credentials, multiple group memberships, etc.

Implementation of g-SIS is a work in progress and many approaches are possible. The access machines need to have a Trusted Computing Base (TCB) that has a hardware and software component. The TPM (although not the only trusted hardware infrastructure required) provides the hardware root of trust. The software component comprises of a trustworthy kernel (possibly a microkernel like L4 [1] or a VMM [4]) and the TRM.

7. REFERENCES

[1] The L4 microkernel family. *http://os.inf.tu-dresden.de/L4/*.

[2] TCG specification architecture overview. *http://www.trustedcomputinggroup.org*.

[3] A. K. Babdara, E. C. Lupu, and A. Russo. Using event calculus to formalise policy specification and analysis. In *Proceedings of the 4th International Workshop on Policires for Distrubuted Systems and Networks (POLICY 2003)*, pages 26–39, 2003.

[4] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. L. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. *Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 164–177, 2003.

[5] A. Cimatti, E.M.Clarke, F. Giunchiglia, and M. Roveri. NuSMV: A new symbolic model checker. *International Journal on Software Tools for Technology Transfer*, 2(4):410–425, 2000.

[6] E. M. Clarke, E. A. Emerson, and A. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Language and Systems (TOPLAS)*, 8(2):244–263, 1986.

[7] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press, Cambridge, Massachusetts, USA, 1999.

[8] K. Fisler, S. Krishnamurthi, L. A. Meyerovich, and M. C. Tshchantz. Verification and change-impact analysis of access-control policies. In *ICSE*, pages 196–205. ACM Press, 2005.

[9] D. Gilliam, J. Powell, and M. Bishop. Application of lightweight formal methods to software security. In *Proceedings of the 14th IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 2005)*, 2005.

[10] D. P. Guelev, M. Ryan, and P. Y. Schobbens. Model checking access control policies. In *Proceedings of the 7th Information Security Conference*, volume 3225 of *Lecture Notes in Computer Science*. Springer-Verlag, 2004.

[11] M. A. Harrison, W. L. Ruzzo, and J. D. Ullman. Protection in operating systems. *Comm. of the ACM*, pages 461–471, August 1976.

[12] S. Jha, N. Li, M. Tripunitara, Q. Wang, and W. Winsborough. Towards formal verification of role-based access control policies. To appear in IEEE Transactions on Dependable and Secure Computing (TDSC), 2008.

[13] S. Jha and T. Reps. Model checking SPKI/SDSI. volume 12, pages 317–353, 2004.

[14] R. Krishnan, R. Sandhu, and K. Ranganathan. PEI models towards scalable, usable and high-assurance information sharing. *Proc. of the 12th ACM Symposium on Access Control Models and Technologies*, pages 145–150, 2007.

[15] A. Lee, K. Minami, and M. Winslett. Lightweight cnsistency enforcement schemes for distributed proofs with hidden subtrees. *Proceedings of the 12th ACM symposium on Access control models and technologies*, pages 101–110, 2007.

[16] A. Lee and M. Winslett. Safety and consistency in policy-based authorization systems. *Proceedings of the 13th ACM conference on Computer and communications security*, pages 124–133, 2006.

[17] N. Li and M. V. Tripunitara. Security analysis in role-based access control. *ACM Transactions on Information and System Security*, 9(4):391–420, November 2006.

[18] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems*. Springer-Verlag, Heidelberg, Germany, 1992.

[19] M. J. May, C. A. Gunter, and I. Lee. Privacy APIs: Access control techniques to analyze and verify legal privacy policies. In *Proceedings of the 19th IEEE Computer Security Foundations Workshop*, 2006.

[20] J. M. McCune, T. Jaeger, S. Berger, R. Caceres, and R. Sailer. Shamon: A system for distributed mandatory access control. *Proc. of the 22nd Annual Computer Security Applications Conference*, pages 23–32, 2006.

[21] J. McLean. Security models. *Encyclopedia of Software Engineering. Wiley & Sons*, 1994.

[22] K. McMillan. *Symbolic Model Checking: An Approach to the State Explosion Problem*. Kluwer Academic, 1993.

[23] A. Pnueli. The temporal logic of programs. In *Proceedings of the 18th IEEE Symposium on Foundations of Computer Science*, volume 526, pages 46–67, 1977.

[24] S. Rafaeli and D. Hutchison. A survey of key management for secure group communication. *ACM Computing Surveys*, pages 309–329, September 2003.

[25] M. Reith, J. Niu, and W. Winsborough. Model checking to security analysis in trust management. In *ICDE, Workshop on Security Technologies for Next Generation Collaborative Business Applications (SECOBAP'07)*, 2007.

[26] M. Reith, J. Niu, and W. Winsborough. Role-based trust management security policy analysis and correction environment (RT-SPACE). In *International Conference on Software Engineering (ICSE) Research Demonstration*, 2008.

[27] R. Sailer, X. Zhang, T. Jaeger, and L. van Doorn. Design and implementation of a tcg-based integrity measurement architecture. *Proc. of the 13th conference on USENIX Security Symposium*, 13:16–16, 2004.

[28] R. Sandhu. Lattice-based access control models. *IEEE Computer*, 26(11):9–19, November 1993.

[29] R. Sandhu. Rationale for the RBAC96 family of access control models. In *RBAC '95: Proceedings of the first ACM Workshop on Role-based access control*, page 9, New York, NY, USA, 1996. ACM Press.

[30] L. Sarmenta, M. van Dijk, C. W. O'Donnell, J. Rhodes, and S. Devadas. Virtual monotonic counters and count-limited objects using a tpm without a trusted os. *Proceedings of the first ACM workshop on Scalable trusted computing*, pages 27–42, 2006.

[31] A. Schaad, V. Lotz, and K. Sohr. A model checking approach to analysis organizational controls. In *Proceedings of the 11th ACM Symposium on Access Control Models and Technologies (SACMAT06)*, pages 139–149, 2006.

[32] A. P. Sistla and M. Zhou. Analysis of dynamic policies. In *Proceedings of Foundations of Computer Security and Automated Reasoning for Security Protocol Analysis*, pages 233–262, 2006.

[33] S. D. Stoller, P. Yang, C. R. Ramakrishnan, and M. I. Gofman. Efficient policy analysis for administrative role based access control. In *CCS'07*, pages 445–455, 2007.

[34] N. Zhang, M. Ryan, and D. P. Guelev. Evaluating access control policies through model checking. In *Proceedings of the 8th Information Security Conference*, volume 3650 of *LNCS*, pages 446–460. Springer-Verlag, 2005.

APPENDIX

A. MODEL CHECKING USING NUSMV

Expression operators !, &, |, and -> represent logical operators "not", "and", "or" and "implies", respectively. Comments follow the symbol "- -". Further, the symbols F, X and U represent the future temporal operators *eventually*, *next*, and *until* respectively (please refer section 3.2).

The set of next assignments execute concurrently in a step to determine the next state of the model. SMV allows nondeterministic assignment, i.e., the value of variable is chosen arbitrarily from the set of possible values. SMV supports macros, which are replaced by their definitions, so they do not increase the system's state space.

A.1 Stale-safe TRM

Code listing A is an SMV implementation of the stale-safe $\text{FSM}_{\text{object}}$ shown in figure 11. Note that the property that is verified is φ_1 (stated as LTLSPEC towards the end). It is re-formulated using the future temporal operators. As reported by SMV, figure 11 satisfies the Weak Stale Safety property. The second property simply makes an additional check that it is always the case, if a subject is able to perform an action on an object then that object was added before the last refresh time. SMV confirms that this is indeed the case.

Code Listing A

```
MODULE main
VAR
--declare attributes
r_ts : 0..100;
leave_ts : boolean;
remove_ts : boolean;
join_ts : {2,18};
--usage count
N : 0..5;
--clock ticks
```

```
ticks : 1..10;
--declare events
--input event from subject
request_event : boolean;
--the latched request_event
request : boolean;
--refresh event received from the CC
refresh : boolean;
--action perform
perform : boolean;
--input event representing if subject has left
leave : boolean;
--input event representing if object has been removed
remove : boolean;
-- declare states
idle : boolean;
authorized : boolean;
refreshed : boolean;

DEFINE
add_ts := 10;
stale := r_ts <= add_ts;
authzE := (add_ts > join_ts) &
(!leave_ts) & (!remove_ts);
authzSS := authzE & !stale & (N>0);

ASSIGN
init(join_ts) := {2,18};
next(join_ts) := join_ts;

init(leave_ts) := 0;
next(leave_ts) := case
idle & refresh & leave : 1;
idle & request & !refresh &
authzE & (stale | N=0) & leave : 1;
authorized & refresh & leave : 1;
refreshed & refresh & leave : 1;
1 : leave_ts;
esac;

init(remove_ts) := 0;
next(remove_ts) := case
idle & refresh & remove : 1;
idle & request & !refresh &
authzE & (stale | N=0) & remove: 1;
authorized & refresh & remove : 1;
refreshed & refresh & remove : 1;
1 : remove_ts;
esac;

init(r_ts) := join_ts;
next(r_ts) := case
idle & refresh & (r_ts <= 90) : r_ts + ticks;
idle & request & !refresh & authzE &
(stale | N=0) & (r_ts <= 90): r_ts + ticks;
authorized & refresh & (r_ts <= 90) : r_ts + ticks;
refreshed & refresh & (r_ts <= 90) : r_ts + ticks;
1 : r_ts;
esac;

init(N) := 5;
next(N) := case
idle & refresh : 5;
idle & request & !refresh &
authzE & (stale | N=0): 5;
authorized & refresh : 5;
refreshed & refresh : 5;
--perform
authorized & !refresh & authzE & (N>0) : N - 1;
1 : N;
esac;

init(request) := 0;
next(request) := case
idle & request_event & !refresh & authzE: 1;
authorized & !refresh & (!authzE | authzE): 0;
1: request;
esac;

init(idle):= 1;
next(idle):= case
idle & refresh : 1;
idle & request & !refresh & !authzE : 1;
idle & request & !refresh & authzSS : 0;
idle & request & !refresh &
authzE & (stale | N=0) : 0;
authorized & !refresh & !authzE : 1;
authorized & !refresh & authzE & (N>0): 1;
refreshed & !authzE : 1;
1: idle;
esac;

init(authorized):= 0;
next(authorized):= case
idle & request & !refresh & authzSS : 1;
refreshed & authzSS : 1;
authorized & refresh : 1;
authorized & !refresh & !authzE : 0;
authorized & !refresh & authzE & N=0: 0;
1 : authorized;
esac;

init(refreshed):= 0;
next(refreshed):= case
idle & request & !refresh &
authzE & (stale | N=0) : 1;
refreshed & !authzE : 0;
refreshed & authzSS : 0;
authorized & !refresh & authzE & N=0 :1;
1 : refreshed;
esac;

init(perform) := 0;
next(perform) := case
authorized & !refresh & authzE : 1;
1 : 0;
esac;

---formula phi1 for WEAK STALE SAFETY
LTLSPEC G ( (refresh & authzE & F request) ->
(X((!refresh | (refresh & authzE) & !request) U
((request & F perform) -> (!perform &
(!refresh | (refresh & authzE)) U perform)))) )

LTLSPEC G ( perform -> add_ts < r_ts )
```

```
[root@localhost TRMobject.smv]# NuSMV
trm_object_safe.smv
*** This is NuSMV 2.4.3 (compiled on Mon May
5 02:33:40 UTC 2008)
*** For more information on NuSMV see
<http://nusmv.irst.itc.it>
*** or email to <nusmv-users@irst.itc.it>.
*** Please report bugs to <nusmv@irst.itc.it>.

-- specification
G ( (refresh & authzE & F request) ->
(X((!refresh | (refresh & authzE) & !request) U
((request & F perform) -> (!perform &
(!refresh | (refresh & authzE))
U perform)))) ) is true
-- specification
G (perform -> add_ts < r_ts)  is true
[root@localhost TRMobject.smv]# NuSMV -int
trm_object_safe.smv
NuSMV > go
NuSMV > print_reachable_states
################################################
system diameter: 19
reachable states: 1.12752e+06 (2^20.1047) out of
2.48218e+07 (2^24.5651)
################################################
NuSMV >
```

A.2 Stale-unsafe TRM

Code Listing B is an SMV implementation of stale-unsafe $\text{FSM}_{\text{object}}$ shown in figure 10. For brevity, we only show the lines that differ from Code Listing A. One can construct this unsafe machine by replacing the corresponding lines in listing A with the ones specified in listing B. A significant change is the missing check for stale-safety. All occurrences of the variable $authzSS$ has been replaced with $authzE$. authzE is just the access policy and authzSS is the stale-safe version of authzE. Also, checks for the variable *stale* are removed. The property that is specified here is formula φ_0 (Staleness Unaware) which is satisfied by this machine. In order to see the time-stamps of objects that are accessible using this machine, we obtain a counter-example by specifying the second property that checks if the objects being accessed were added after the last refresh time. As you can see in the trace, in state 1.4, the subject performs an action on an object whose *add_ts* is 10. But however the last refresh time-stamp *r_ts* at this point for the subject is 2. This is clearly a stale-unsafe access. Note that we use the -bmc option (for Bounded Model Checking) to get a counter-example of minimal length.

Code Listing B

```
ASSIGN

init(leave_ts) := 0;
next(leave_ts) := case
idle & request & !refresh &
authzE & (N=0) & leave : 1;
esac;

init(remove_ts) := 0;
next(remove_ts) := case
idle & request & !refresh &
authzE & (N=0) & remove: 1;
1 : remove_ts;
esac;

init(r_ts) := join_ts;
next(r_ts) := case
idle & request & !refresh & authzE &
(N=0) & (r_ts <= 90): r_ts + ticks;
esac;

init(N) := 5;
next(N) := case
idle & request & !refresh &
authzE & (N=0): 5;
esac;

init(idle):= 1;
next(idle):= case
idle & request & !refresh & authzE : 0;
idle & request & !refresh &
authzE & (N=0) : 0;
esac;

init(authorized):= 0;
next(authorized):= case
idle & request & !refresh & authzE : 1;
refreshed & authzE : 1;
esac;

init(refreshed):= 0;
next(refreshed):= case
idle & request & !refresh &
authzE & (N=0) : 1;
refreshed & authzE : 0;
esac;

--formula phi0, STALENESS UNAWARE
LTLSPEC G( (refresh & F(request & authzE)) ->
(!request U ((request & authzE & F perform) ->
((!perform & (!refresh |
(refresh & authzE))) U perform))))

LTLSPEC G( perform -> add_ts < r_ts)

[root@localhost TRMobject.smv]# NuSMV
trm_object_unsafe.smv
-- specification
G( (refresh & F(request & authzE)) ->
(!request U ((request & authzE & F perform) ->
((!perform & (!refresh |
(refresh & authzE))) U perform)))) is true
-- specification
G (perform -> add_ts < r_ts)  is false
-- as demonstrated by the following execution sequence
Trace Description: LTL Counterexample
Trace Type: Counterexample
.
.
.
[root@localhost TRMobject.smv]# NuSMV -bmc
trm_object_unsafe.smv
-- no counterexample found with bound 0
-- no counterexample found with bound 1
.
```

```
.
.
-- no counterexample found with bound 10

-- no counterexample found with bound 0
-- no counterexample found with bound 1
-- no counterexample found with bound 2
-- specification  G (perform -> add_ts < r_ts) is false
-- as demonstrated by the following execution sequence
Trace Description: BMC Counterexample
Trace Type: Counterexample
-> State: 1.1 <-
  r_ts = 2
  leave_ts = 0
  remove_ts = 0
  join_ts = 2
  N = 5
  ticks = 1
  request_event = 1
  request = 0
  refresh = 0
  perform = 0
  leave = 0
  remove = 0
  idle = 1
  authorized = 0
  refreshed = 0
  authzE = 1
  add_ts = 10
-> Input: 1.2 <-
-> State: 1.2 <-
  request_event = 0
  request = 1
-> Input: 1.3 <-
-> State: 1.3 <-
  idle = 0
  authorized = 1
-> Input: 1.4 <-
-> State: 1.4 <-
  N = 4
  request = 0
  perform = 1
  idle = 1
[root@localhost TRMobject.smv]# NuSMV -int
trm_object_unsafe.smv
NuSMV > go
NuSMV > print_reachable_states
################################################
system diameter: 20
reachable states: 1.02864e+06 (2^19.9723) out of
2.48218e+07 (2^24.5651)
################################################
NuSMV >
```

Part II

Handling Semi-Trustworthy Partners: Game Theoretic Approaches for Information Sharing

USING GAME THEORY TO MITIGATE THE EFFECTS OF A BIOTERRORIST ATTACK

Ryan Layfield, Murat Kantarcioglu and Bhavani Thuraisingham

The University of Texas at Dallas

USING GAME THEORY TO MITIGATE THE EFFECTS OF A BIOTERRORIST ATTACK

by

RYAN PATRICK LAYFIELD, B.S., M.S.

DISSERTATION

Presented to the Faculty of

The University of Texas at Dallas

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT DALLAS

December, 2008

ACKNOWLEDGEMENTS

I want to first start by thanking my friends and family for their support of my research, interest, and pursuits over the years. Without the blessing of their help and encouragement, much of this would not have been possible. In particular, I want to thank my mom and dad for working so hard to provide me with the guidance and opportunities over the years of my childhood. It has become clear to me over the years how much I have benefited from their examples in my life.

A special thanks goes to the professors and fellow students over the years that have made this possible. I thank Dr. Bhavani for providing me the opportunity and funding to do research full-time. I thank Dr. Murat for his guidance, support, and wisdom to helping me develop my ideas. I also want to thank Dr. Khan for giving me the opportunity in the first place to become a PhD student and providing guidance on my initial work. Of course, none of this would be possible without the support and tutelage of the University of Texas at Dallas. I want to thank my colleague Li Liu for helping me through sharing her insights and personal experience in the program, many of which have helped me arrive at this moment.

Last, but certainly not least, I want to thank my friends for their support. I thank my friend Dominic Richards for helping and advising me in several regards through both graduate and undergraduate years at UTD. I thank my good friend and now wife Lakeysha Layfield who has supported me and made sacrifices for me over the years to make this possible. My brother James Layfield deserves thanks as well for being there when I've needed help the most. Indeed, it is through this program that I have found exactly how much those who matter to me have been an essential part of my progress and success.

We owe a considerable amount of gratitude to Dr. Robert Herklotz for his funding of a majority of the work that has gone into this dissertation. On behalf of our team, we wish to express our thanks for his continued support of our research within security. Such funding is an investment in the safety and well being of the collective future of human kind against the ever growing threats present in our time.

November, 2008

USING GAME THEORY TO MITIGATE THE EFFECTS OF A

BIOTERRORIST ATTACK

Publication No. _____

Ryan Patrick Layfield, Ph.D.
The University of Texas at Dallas, 2008

Supervising Professor: Dr. Bhavani Thuraisingham

The goal of our work was to apply proven human-oriented situation analysis with existing simulation techniques to explore new ways of enhancing security through anticipation of human thought processes and activity. In particular, we used social networking to model relationships and game theory to model motivations of those participating. The end result of these studies yielded a combination of methods to anticipate, plan for, and reduce the impact of a biological attack. The SIR model was modified and applied to an individual-level social network through the use of theatres and approximated influences due to relationships, creating a unique, high performance mathematical model to observe the spread of disease. This model was then simulated in a number of scenarios spanning the use of possible biological attack situations, inoculations, and several novel intervention methods. The results of the simulation were then analyzed as a Stackelberg game in order to search for a lower bound to the expected costs and loss of human life under the assumption that the attacker goes last.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Psychology has always played an important role in warfare. From the intimidation of enemy troops to what motivates an international terrorist organization to carry out attacks on civilians, understanding the mindset of an opponent can be crucial to a desirable outcome. In the most ideal situations, a sufficient amount of data can be used not only to anticipate actions but influence them as well. Terrorist organizations, however, have often represented the counterexample to such logical approaches due to the seemingly irrational nature of their choices and motivations. This understood being able to effectively model such a mindset is crucial in countering the potential damage these groups are capable of. The objective of our work is to bring a clearer view of the actions and motivations behind malicious activity through proven models and enhance them to provide a more effective level of detail and, ultimately, security.

One aspect of modeling that we believe warrants investigation is the application of human-centric analysis techniques. In essence, we believe the focus on patterns and data mining have effectively ignored the fact that humans are responsible for a great deal of it. Understanding the perspective which individuals have and the way the interact with others allows us to begin to model security in such a way that prevents undesirable actions by first determining how they arrived at their choices originally. Additionally, understanding that it

is human nature to interact with others provides an opportunity to form a better model of communication surveillance and security planning.

The relationships between individuals often form a complex web of associations that can be captured through the observation of interaction. The field of social networks, which study such graphs, offers a wealth of information in how to observe, model, and derive information from the interactions between people. Relationships and the communication therein can provide important clues when attempting to search for malicious conversations.

These same relationships can also be a liability in security. In epidemiology, there are a number of vectors by which a contagion can potentially pass. When the biological agent in question is particularly contagious through human transmission, the primary vector by which it travels is often via either direct personal contact or proximity between healthy and sick individuals. Understanding the social networks of a victimized population is a vital asset in anticipating and dealing with the use of a biological weapon, as the very relationships which an individual may cherish could be the means by which they become a casualty.

While social networks provide a way to study how humans interact, it does not necessarily expose their personal motivations. To address this problem, we must turn to one of the sciences of how individuals arrive at their decisions: game theory. Game theory is the study of human motivation under the assumption that people whom make a decision wish to maximize their gain from it in a rational manner. Applying this science to situations involving terrorism reveals answers to questions such as the benefit of an attack versus the expected response, the aftermath, and how to achieve the best outcome possible amidst potentially difficult options. Understanding what motivates a choice can assist in understanding where an attack may take place and build a better defense against it. Through

this information, additional steps can be taken to mitigate the overall effect thereof. In the realm of bioterrorism, such effects may include how many human lives are saved and whether or not civic resources are available to properly deal with an attack.

When we combine these fields with an existing epidemic model, we have the potential to generate a comprehensive method of dealing with actual threats. With the right data, such a simulation could yield an accurate prediction of the outcome of an attack with a cost-effective use of resources to form a response. Such results can be further refined with a list of potential targets and it becomes possible to place an upper bound on the damage done by an attack, through selection of the best possible combination of responses. The end result would essentially be the best possible defense with regards to both the present and future threats to human life.

1.1 Motivating Scenario

Our motivating scenario begins with a country which has reason to believe one or more terrorists groups have rallied against its' political ideals and now threatens the lives of the citizens within. The country's governing body recognizes them as a legitimate threat and now wishes to take action to protect itself. Only a few details are known about the group, but one of the most troubling aspects of their capabilities is their investment in weaponized biological agents. The first step that the government wishes to take is to assess the legitimacy of the threat. It is not clear where leadership in the group resides; the organization has chosen to operate as a series of cells to minimize the impact of being discovered. An uninformed move made against one cell could inadvertently trigger an even greater repercussion and invalidate the investment of resources made as plans are changed. One of the positive points in the situation is that due to their physical distribution, they must rely on

public communication networks to avoid detection. In order to gather intelligence, the first step which the government takes is to construct a method of surveillance to monitor the networks within their own borders for cell activity, using existing research on text parsing and anomaly detection.

However, their efforts reveal that the group extends well beyond the authority and capability of their own security agencies to observe them. The group is revealed to be a truly international movement, and despite the completeness of their surveillance solution it is clear that crucial pieces of information are outside of their ability to capture. Through their inquiries, they discover that many other countries share their concern; they too believe they are concerned by the threat of terrorism. It is through their mutual need to defend against such a complex enemy that they form an alliance to share intelligence. Each country has agreed to share what they have with each other as long as they receive the intelligence of their partners in exchange. However, there is no authority to ensure the alliance is enforced, requiring each country to operate within their own means of insuring the information they receive is legitimate and useful.

After time has passed and enough information has been gathered to assess the threat, the government discovers that a biological attack is indeed going to be carried out on their native soil soon. The legislative representatives have determined a course of action is necessary. However, due to a lack of prior data on actual acts of bioterrorism, the government cannot determine the impact of an attack. Multiple targets across several cities are suspected. Many existing simulations are determined to either be too simple to yield useful data or complex to the point that all possible targets cannot be evaluated within the time available with existing

resources. A large search space must be explored with available limited computer resources. Thus, a new simulation-based model is necessary in order to prepare for the attack.

Finally, the government has successfully narrowed down a small number of potential targets in a single city. Assuming that the attacker wishes to maximize the damage to the population to make their point clear, they must take into consideration several possible ways of defending themselves. The circumstances are made more complex due to the cost of each method, the ways in which they can be combined, and using limited resources available. Their goal is to create a cost-effective solution to deal with the threat and ensure an upper bound on the worst case scenario, and implement it before the attack.

The goal of this dissertation is to address each of these needs to form a complete security solution. We have devised solutions to every problem mentioned, building on existing proven work in the field of security while devising our own novel approaches. The ultimate motivation of this work is to understand the threat of bioterrorism and how to prepare for it.

1.2 Related Work

The ambitious nature of our work requires us to address a number of areas. As such, we have performed extensive research on related bodies of work in our desire to create a better solution than currently offered by existing work. We offer the highlights of the works available that most closely resemble our own. The sections are appropriately divided based on the field in which the work was performed.

1.2.1 Communication Surveillance

Surveillance on communications has been a popular avenue of research in the past decade. Understanding how information can be observed and correlated is crucial to developing an

accurate anomaly detection system, which allows us to automate the intelligence gathering process. The work of Ben-Dov et al. demonstrates the ability to enhance link mining of news sites by using available tools to semantically comprehend the contents of a document. [1] One experiment performed by the group successfully discovered correlations between two individuals based simply on their presence within the same sentence. Successful examples can also be found in the field of semantic web analysis. [2]

1.2.2 Game Theory

Game theory has long been a staple of analysis within social and political sciences. We use it to model the motivations of both benevolent and malevolent parties. In particular, we wish to model the conflict of interest between an attacker and a defender in the realm of terrorism. Robert Axelrod in his book "The Evolution of Cooperation" explores the findings of a contest he created using a variation of the repeated Prisoner's Dilemma. In it, all participants were encouraged to submit their own algorithms in an attempt to find the superior approach. Surprisingly, the winner of the contest was a retaliatory algorithm known as Tit-for-Tat, which essentially cooperated with another player unless there was a change of strategy, in which it selected an identical strategy on the next round. [3]

Several related areas of work have already considered the possibilities of game theory as applied to information sharing. In particular, a great deal of work has been done on peer-to-peer networks. Within these file sharing systems, independent players join and leave at their leisure, seeking to download a file or files with the help of other participants. Problems arise when a new participant joins the network and downloads a resource from other peers but never actually contribute to the group. This process, known as leeching, has been a large problem in piece-meal file sharing protocols such as the popular BitTorrent. The work of

Gupta et. al [4] and Buragohain et al. [5] both deal with this behavior by creating a system of incentives for further contribution.

1.2.3 Trust in Peer-based Networks

Trust is crucial to the continued operation of several distributed systems to deal with malicious activity and counter-productive behavior. Information exchange methods have been particularly useful in the formation of ad-hoc networks. Seredynski et al. used the concept of an evolving genetic algorithm to enhance security and trust in a wireless network among multiple nodes relaying data packets. [6] Competition among the nodes occurred between behaviors that were always selfish and evolutionary behaviors that become selfless. Cooperation naturally evolved with a success rate directly dependent on the initial presence of the malicious nodes, achieving an average success rate of over half despite the lack of punishment methods. Other works have taken a purely game-theoretic approach to trust. The work of Cascella performed analysis of an infinitely repeated form of the prisoner's dilemma with regard to persistent players randomly selected. [7] They found the introduction of a reputation system allowed punishment systems based on discriminating between good and bad reputations succeeded as long as players were sufficiently patient to achieve the results. A similar body of research attempts to deal with peer-to-peer trust in scenarios involving military cooperation amidst forces deployed in the field. [8] Given a military body as a dividable resource, they attempt to address the problem of resource allocation in situations where either a central authority isn't robust enough or the resources span multiple international owners. The core issues they addressed were dealing with malicious reports attempting to sabotage trust ratings and attempting to give more control to an agent's own rating.

Our own contributions to trust offer a unique approach in which perfect information about a situation comes at a cost. Thus, trust itself comes at a price, and must be weighed against other needs such as being efficient. We use this cost of trust primarily to augment our game theory in regards to the exchange of information.

1.2.4 Epidemiology Simulations

Understanding how a biological agent can start an epidemic requires a robust understanding of contagion modeling. As such, we have done significant research in the area of simulations to address the need for observing how a disease could spread throughout a population. BioWar is an agent-based model based on a variety of factors that simulate a number of infectious diseases at the lowest level possible. [9] People are represented individually in such a way that travel, human contact, and even physical location are all considered throughout the course of a virtual day. Episims is a product of research efforts at Los Alamos National Laboratory [10] designed to model epidemics at a similar level of detail, with a greater focus on the demographic a person belongs to, their location, and activities. Some of this information is taken directly from a small public survey aimed at gathering information about an individual's daily routine, while the transportation data is derived from TransSims. Users of the model include Eubank et al., whose work suggests that epidemics can be contained with simple vaccination policies targeted at major convergent components of the network. [11] Complexity however is not without cost. Each level of detail adds both a need for more storage and computations resources on a system. Episims, for example, requires high performance computing resources to run a simulation within a reasonable amount of time. [12] More importantly, if a model is designed to address a certain level of realism, the appropriate data must be available to take advantage of it. A model which considers a large

population's daily hygiene would be difficult to justify without an appropriate cross section of surveyed data of the region in question.

We use the SIR model to build an epidemiology simulation as described in chapters 5 through 7. In the original model, the chance of infection and recovery are constants that apply uniformly to every individual. [13] The population total remains constant and can be calculated by adding up the number of members of each state. While powerful, it does not directly provide room for more detailed interactions beyond collapsing multi-dimensional data into simple probabilities. We also believe it cannot accurately portray many of the impacts of real life social networks in how a contagion spreads. Research suggests that the heterogeneity of social networks can yield an intricate propagation of a contagion that cannot be modeled by homogenous state-based models alone. [14] [15] [16]

Several branches of research have attempted to refine the SIR model to enhance its' accuracy and address the increasing need for practicality. Moshe Kress of the Naval Operations Research Department [15] abstracted the equations and applied it to the spread of smallpox thru an urban population of 6 million. A host of states were introduced to reflect infectious conditions associated with the lifecycle of the disease. The social network was constructed across a population of several million, though the household size was fixed and the members of which could visit any gathering area with equal probability. The author asserts this is a basic generalization founded upon the small world property of most real-life networks. Satuma et al. considers the more complex functionality attached to state transitions in both continuous and discrete forms of the model while retaining all of the SIR model's most desirable qualities. [13] The work of Stattenspiel et al. applies the original model to deal with mobility among geographical regions to describe a measles epidemic in Dominica. [17] The

work of Myers et al. [18] added social networking in such a way that allows for both varying strengths of associations as well as heterogeneous bidirectional linking. Myers points out that several kinds of contact due to various roles may result in a different infection for one participant than the other. They also use the concept of percolation within a network coupled with an implementation of the SIR model that groups individuals in the same household as one node. Another extension is that of [19], which introduced a discrete-time model to allow for stepwise simulations. The authors show that much of the model can be extended while retaining many of the SIR's most desirable qualities. These, along with several others, reinforce the notion that the model provides a solid foundation for epidemiology. [20] [21] [22]

The work of [61] takes a look at the general need for models within epidemiology and turns to an agent based software model to provide. Although they do not actually create a specific model for use, they instead take advantage of the existing agent-based behavior system called StarLogo to model the outbreak of tuberculosis within a homeless shelter. The level of detail in which they simulate the situation is notable, as the physical location in question is literally represented within the system in terms of where furniture is within rooms, how it is used, and the movement of people over time. The work of [62] analyzes existing vaccination strategies based on game theory and use a projected dynamic system in an effort to locate Nash equilibrium. They set aside the SIR model in favor of constructing a function which provides a probability of infection based on the current status of the population and the number of vaccinations that have currently been administered. Their entire simulation strategy is rooted in effective theoretical foundations rather than discrete simulation. The work of Banks et al. directly applies game theory to a simulated epidemic of smallpox within the United States

due to bioterrorism. [63] The uncertainty regarding perceived values by an opponent are considered and mitigated via Monte Carlo methods. This particular work is perhaps the closest one to our own, though our approach via Stackelberg games and the customized epidemic modeling provide an important distinction.

CHAPTER 2

MESSAGE CORRELATION IN AUTOMATED COMMUNICATION SURVEILLANCE THROUGH SINGULAR VALUE DECOMPOSITION AND WORD FREQUENCY ASSOCIATION

2.3 Security in Multiplicity

The first step the country must take is to monitor the public communication networks. We assume that these networks result in a large amount of data to be sifted through on a regular basis that would be impractical to do so entirely by hand. However, there are certain aspects of understanding information that data mining alone is currently incapable of. In order to determine when the terrorist cells are exchanging vital information, they must automate their surveillance in such a way that permits them to offload the majority of the work to an automated system that only involves human intervention when necessary.

One of the biggest challenges in automated message surveillance is the recognition of suspicious content. A classic approach to this problem is constructing a set of keywords (i.e. ‘anthrax’, ‘contagion’, ‘smallpox’, ‘spores’) considered malicious. In the event that a communiqué contains one or more of these words, the message is flagged appropriately for further review.

However, there are two drawbacks to this particular approach. First, it is reasonable to assume that such relatively static keywords will not always be present in messages that would otherwise warrant suspicion. Second, there is little guarantee that a sufficiently

intelligent individual will not recognize such surveillance is in place and, instead, use substitute words in place of known keywords.

David Skillicorn of Queens University has suggested a different approach in his work on the Enron e-mail dataset. [23] In his research, he outlines a method for using singular value decomposition, abbreviated SVD, in the interest of recognizing trends in such topics as e-mail and social networks. Within his work, he constructed a matrix composed of word frequencies associated with each message, ranked via a composite of the number of times the words appear and the number of times they are 'expected' in casual conversation. We believe that this work can be utilized in our system by extrapolating some of the techniques he used and applying them to a real-time message monitoring system.

We propose a system that builds on the word frequency techniques outlined in Skillicorn's work. By using the Enron e-mail dataset and altering the role of his use of the SVD, we believe we can achieve a higher degree of message correlation accuracy based on the intersection of uncommon words within conversations.

2.4 Identifying Malicious Information via Association

The goal of the security-conscious country is to essentially automate the recognition of anomalous or suspicious messages within typical communication traffic. [24] Our areas of interest include social network analysis, text processing, and result filtering. This chapter will determine the effectiveness of singular value decomposition applied as a text filtering and correlation system in our system.

One particular approach we consider is the recognition of social context. Given a message that was deemed suspicious by a detection technique, we assume there is an increased probability that future messages bearing similar characteristics to the detected message

warrant further suspicion. For example, a member of a cell mentions the transport of a biological agent to another cell, the members of which discuss the best way to receive the shipment. We track the conversations through the generation and passing of tokens carrying the characteristics of the topics in a model derived from observed social interaction within the message passing network. Currently, these tokens are triggered by the presence of two or more keywords from a list tailored to our dataset.

The Enron e-mail dataset was originally made available by the Federal Energy Regulatory Commission during its investigation of the company. It was purchased by MIT for use in data analysis, and is currently available as both a raw set and a compiled database from Carnegie Melon University, courtesy of William Cohen. This dataset is useful to e-mail surveillance research as it is a freely available collection of messages that has been generated within a 'real world' scenario. After pruning and restructuring for consistency, the overall corpus is composed of approximately 250,000 e-mails generated over a five-year period.

Each e-mail remains in its' original raw ASCII-encoded text format, conforming to the RFC 2822 standard. The simplicity of this specification and the data it requires for proper mail transport assists with analysis in a number of ways. First, every e-mail must contain a unique identification number assigned by the originating server, providing us with a rudimentary way to track threads. Second, each message must have a well defined origin and destination address, with the exception of any group aliases. Third, every message has a time and date stamp that reflects when it was originally sent, providing an explicit ordering to the messages.

Finally, the body text of each e-mail has already been cleaned of any escape sequences or special characters. This greatly simplifies any preprocessing necessary for text parsing. The

content of the Enron dataset is its' most useful aspect. We understand that, at some point in time, e-mail began circulating that eventually led to the investigation of the entire corporation. This has been documented by countless news reports, committees, and even other research groups. Coupled with the official announcement on January 9th, 2002 that the United States Department of Justice was beginning its official inquiry, we have definitive moments and individuals that can be scrutinized to determine the overall effectiveness of our experiment.

```
Message-ID:
<8051748.1075855665834.JavaMail.evans@thyme>
Date: Wed, 13 Dec 2000 08:01:00 -0800 (PST)
From: rebecca.cantrell@enron.com
To: phillip.allen@enron.com
Subject: Re:
Mime-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
X-From: Rebecca W Cantrell
X-To: Phillip K Allen
X-cc:
X-bcc:
X-Folder:  \Phillip_Allen_Dec2000\Notes  Folders\All
documents
X-Origin: Allen-P
X-FileName: pallen.nsf

…
```

Figure 2.1. An excerpt from the Enron e-mail dataset.

Singular value decomposition offers a pattern-centric approach to text-based analysis. Given a m by n matrix A of real or complex numbers, we can decompose it into a series of component matrices: U, S, and V. U represents the patterns among values contained among the objects. In this experiment, the objects are the messages. V embodies the patterns among the attributes of each object. This will usually emerge through the ranks of words within the

messages. The S matrix is a diagonal matrix confirming to the dimensions *n* by *n* that stores the singular values of *A*. Essentially, the most 'interesting' parts of the original matrix are made evident through the breakdown process and isolated for our use. [25]

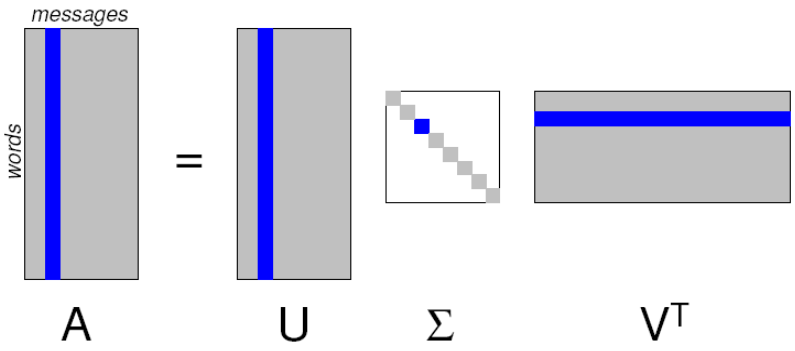

Figure 2.2. Correlation of a single message across the decomposition

Once the SVD process has been carried out, the resulting matrices can be used for a variety of purposes. One particular way to use the results is the elimination of deviants from the patterns to filter out noise. We define noise among our messages as misspelled words, accidentally insertion of punctuation, and anything else a user may inadvertently insert within their message that cannot be removed automatically via preprocessing in a timely and effective manner. Other uses include object correlation, signal processing, and even text retrieval [26]. Most of the uses for SVD stem from the fact that the components, once modified, can be used to build a new matrix that contains only the most 'interesting' qualities of the original.

2.5 Applying the SVD

Our original system [24] has been outfitted with a new detection module based on the SVD method. The algorithm within the module requires a ranked word database, one or more messages that share the same token, and a set of messages that have originated from them representing recent traffic. A matrix is constructed based on the word content of the messages, which is decomposed according to the SVD method. A threshold is set for the noise, and the results are analyzed for any correlation. If a message in the recent traffic set has a strong correlation with one or more of the token holding messages, the token is passed to all recipients of the message.

We have made a number of assumptions based on the characteristics of our dataset. First, we assume that the content among messages is not obscured through covert communication techniques, such as encryption or word replacement. Second, we assume that on any given topic there are a number of words that are rarely used outside of the text. This includes subjects, terminology, and any form of slang. Finally, we assume that subsequent conversations based on an original message will not always carry the same unique nouns that would otherwise make detection simple. For example, a weapons dealer selling weaponized smallpox will likely be flagged on the initial offer to one of the cells, but the subsequent messages regarding price negotiation and leaving out the product name may appear as innocuous as traffic from a public auctioning site. This is a crucial weakness in traditional techniques, as conversations cannot always be tracked via a unique message ID. Additionally, information can potentially be passed from one person to the next in a completely separate e-mail thread. By tracking this information and using prior context, we can potentially bridge the gap between messages.

To determine the rank of a particular word, a brief word history is kept. The 5,000 most common words within the Brown Corpus of Standard American English are used to seed the database. As words are extracted from messages, each word and the number of times it occurred within the message is either inserted into the database if the word is new or used to update the existing word rank.

The matrix constructed is created based on the number of messages involved and the words present. Each message represents a column, while each row represents a word. Thus, each cell represents the number of times a particular word occurs within a particular message. The columns are ordered from most common (left) to least common (right). Mathematically, this is represented by (2.1).

$$\begin{aligned} c_{ij} &= count(w_i, m_j) \\ W &= M_1 \cup M_2 ... \cup M_t \\ w_i &\in W \end{aligned} \tag{2.1}$$

W represents the set of all words that occur in the union of the word sets for each message *M*. The function *count* returns the number of times a particular word *wi* occurs within a message m_j. Note that M_j is derived from the words in m_j; however, m_j may contain the same word more than once while M_j is a list that strictly represents all words only once.

Once the SVD technique has been used on the messages, the noise is removed through the use of a threshold of 0.25, derived from observation within experimentation. After the singular value dimensions falling below this threshold are removed, the matrix is re-constructed from the components. The resulting matrix should assist in simplifying the amount of data that must be analyzed. The correlation of any two messages is based on a total score. Each word that is found in both messages is given a rating according to (2.2).

The variable w_i represents the word that occurs in messages m_j and m_k. The *count* function returns the number of occurrences of the word within a supplied message. The *rank* function is a cumulative count of the occurrence of the word w_i in all messages up to this point. This equation is designed to place emphasis on words that occur less frequently others. It is assumed that rare words that occur in both messages are more likely to be good candidates for a topical match. Note that s_i is naturally normalized to one. Given that the rank of a word is adjusted before these messages are processed with this equation, the maximum score will be one. In order to determine if two messages match, we use (2.3).

$$score(w_i) = \frac{count(w_i, m_j) + count(w_i, m_k)}{rank(w_i)} \tag{2.2}$$

$$\sum_{w_i \in W_j \cap W_k} score(w_i) > \alpha \tag{2.3}$$

where W_j and W_k are the set of all words contained within m_i and m_k, respectively, and a represents a predetermined threshold that determines the minimum. In theory, this should be some value that is determined by statistical analysis of known correlating messages. For our experiment, we assume *a* is 4.00.

2.6 Real Time Communications

With this hypothetical security system, the country in question must now find a means of applying it to actual networks. Monitoring a continuous stream of data in the interest of security is not a trivial problem. In order to properly classify a single message as normal or suspicious, one must parse the contents, determine the origin, identify the recipients, and determine how prior communication traffic affects the context. Whether or not a message is

suspect, it can theoretically affect the semantic meaning of future communications. This implies that, as time progresses, infinite storage for prior messages is necessary for a 'perfect' detection system that can correlate events from any past instance.

While tools exist for message classification with a variety of intent, there are no known systems uncovered through our own research which establish a localized context for each node in the interest of security. While individuals being monitored may share a similar context in a common environment, there are several scenarios in which the same message passed by two different users does not have the same meaning. For example, if the hypothetical malicious individuals speak of symptoms such as fever, chill, and nausea would most likely be relating to a prior e-mail about the effects of smallpox, as opposed to a patient asking their doctor in regards to a bout with the flu. Hence, there is a need for a system to 'personalize' context data for each participant to properly ascertain security threats.

2.6.1 The Challenge of Data Streams

Since its inception, e-mail has become an increasingly popular form of communication. According to a study performed by research group IDC in 2002, e-mail traffic will increase from 31 billion messages a day to 60 billion by 2006. [29] As of the second quarter of 2005, there are roughly 900 million known users of the internet. [30] Assuming that mail traffic increases linearly, each user receives an average of 60 e-mails a day. Manually sifting through the traffic of a group of a hundred people would require an individual to read 6,000 e-mails a day and have a perfect understanding and recollection of all prior data. Clearly, automated methods are necessary to deal with such volumes.

E-mail anomaly detection is not a new concept. One existing topic of interest is the identification and filtering of spam. Using a set of desirable message attributes, a spam

removal system is responsible for removing all unwanted e-mail from a user's inbox. This frequently includes advertisements, fraudulent topics, general bulk mail, and any other messages that do not appear 'relevant'. Ultimately, this will ideally result in a set of messages consisting only of what the user desires. [31] While not always providing enhanced security directly, spam filtering represents a well-defined area to study.

The deployment of a real-time version of our system has a number of advantages over a traditional approach to security involving fully manual threat assessment. However, these hypothetical strengths must also be considered in terms of the requirements, what the system can detect, and what it cannot detect. As with any approach, all of these constraints must be weighed carefully in an assessment of what is appropriate for a given set of security needs.

2.6.2 Strengths

In theory, the deployment of this system offers a great deal of benefits. First, all analysis is performed in real-time. This means that, once deployed, the system is actively monitoring the available text stream for any and all communication activity. In the event that a malicious situation is identified, an observer of the system can either respond immediately or await further messages to decide whether a security issue exists. This is particularly important when time is a factor in the gathering of information.

Second, the system indirectly models the complex social interactions of individuals. Hence, as messages are passed, it is possible to identify groups of people with malicious intent and how they collaborate. The recognition of social sub-networks can uncover the underlying structure of a subversive organization attempting to collaborate, such as helping to unravel the network within terrorist cells and form a more cohesive picture of the threat.

For example, consider the deployment of this system in the interest of catching a group of criminals involved in smuggling biological agents across international borders. Assume that they are using a message passing network to remain in constant contact along with a multitude of innocent people. Using keywords involving the materials and suspected methods, simple text filtering could create a number of false positives from people simply discussing other crimes mentioned in the news. By overlaying detected keyword uses with social network graphs, we could detect a group of individuals using these words exclusively among themselves. Once the group is properly identified, the entire set of individuals connected could be captured and questioned.

Extending upon this scenario, should any of these individuals be held responsible, the system has already generated a set of conversations shared among the guilty parties. These exchanges could easily translate into an evidence exhibit to be used during prosecution. In turn, the data gathered could be used to further track down other terrorist groups.

2.6.1 Weaknesses

Unfortunately, as promising as such as a system may be, it is of note that the proper operation of the tool has a number of important considerations. First, the system requires that it has a roughly omnipotent view of communication among individuals. For example, it assumes that users of an e-mail server will not use any other e-mail server to communicate, nor any other form of communication that falls outside the bounds of what can be observed. Given that groups of individuals will likely communicate in person at some point, one or more semantic gaps could be created. Such gaps would prohibit token passing among nodes, as well as create inaccuracies within the perceived social network, reducing the overall effectiveness of the system.

Second, there are serious ethical implementations for a system with such far reaching observational capability. Regardless of whether or not individuals are engaging in suspicious activity, social models are being created for future reference. Essentially, the data generated can be used to identify how close two individuals are, what they have been talking about, the common points of contact among them, etc. If an individual uses the monitored text stream exclusively for communication, a fairly accurate model of their relationships can be generated.

To fully understand how such data can be used against an individual, consider a government worker in a management position with access to the detection system that has been observing an individual applying for a position. During the evaluation process, the manager could analyze the social net around the applicant and determine the people they are closest to. These individuals could then be contacted and asked a series of questions about the applicant, their habits, prior employment history, etc. While the manager would benefit greatly from being able to have such data, the potential employee would undoubtedly feel their private life had been violated.

Another weakness of the system is the lack of training methods inherent within it to teach it when certain messages are false positives and false negatives. Although it is assumed that the observing agent can distinguish between results, it is much more convenient to filter out the ‘noise’ to focus more on issues that require more attention. Additionally, given the token use of the system, a serious amount of false contexts could be created that would cause multiple complications for the entire social network. In theory, the impact of false tokens could be eliminated by giving an observer the option to delete specific tokens within the constructed graph, but this is only a temporary solution.

Regardless of how effective the system is, the ultimate weakness that this system faces is how much data must be stored. Traditionally, in most message passing networks, messages are stored at the user's terminal, removing the burden from the server. However, in order for the system to properly determine previous context, all messages passed must be stored in an archive after processing for at least some duration of time. Coupled with the data storage of links among users and the presence of tokens, it is possible that the data requirements of the system could multiply exponentially as more users join a network and average traffic flow increases.

2.7 Experimental Results

Despite our efforts and foundations, our experiments did not yield the desired results. After sifting through 900 messages, we found that our system had roughly 3% true positive results, 59% true negative results, 12% false positives, and 26% false negatives. This was an estimation based on spot-checking individual messages as a subsample of the whole. Clearly, much work is needed to make this model successful and accurate.

The first problem we found is that we lacked a proper metric by which to rate results. There was no perceivable method by which a human could offer guidance for more than a small subset of the real data. The number of permutations between all possible messages made it virtually impossible to generate a sensible set of test data to work with in a reasonable amount of time. Our only real alternatives were to either take the time to construct a data set by hand or generate it. The former option could not be constructed large enough without exponentially increasing the amount of data we would have to keep track of, such as subjects and topics which are related but meant to be correlated with only a weak score. The latter

option opens up a whole new set of problems in terms of proper grammar generation and determining how to relate subjects in a realistic manner.

The next problem we found in our model was that larger messages, which have a huge corpus of words to be considered, would often relate to other large messages simply by the sheer size of less interesting words. This created a balancing problem that we were unable to compensate for. One option was to normalize by the number of words in the message, but that created a problem for more interesting words that only occurred once. In essence, isolating the more interesting words presented a larger challenge than we expected.

For example, when comparing message numbers 124 and 125 in the series, the overall score was 46.5. There was no discernable correlation between their subject matter. Several words did appear to have meaning in their lack of frequency to join the messages, but sheer number of less common words substantially inflated the correlation score. As a result, we must classify this as a false positive.

At the same time, smaller messages faced the opposite problem. Messages 268 and 123 were very close in that their diction was unique in how it overlapped, but it only scored 0.67 in our algorithm. Although the relation of the topics is difficult to judge as a false negative, the similarities should have scored higher. In essence, we need some form of weighting the scores based on size.

The token passing did not work as expected. Conversations were not properly tracked in all but a handful of instances. Clearly, we are on the right thought process in terms of the nature of the problem as reflected in reality, but the implementation needs additional work. Tokens need to degrade after a period of time when the context has faded, but not to the point that

they are irrelevant. Additionally, improving our detection techniques may help this aspect of our work directly. Otherwise, it is difficult to judge the effectiveness of social contexts.

Another challenge we faced was the resources necessary to process each message. There was a strain on resources to match the pace a real server would encounter. The SVD approach was particularly difficult on the system, as filling a large matrix was difficult to optimize. In retrospect, we may look into ways in which we can have a running cursor inserting each message into the matrix and extracting results from the combined matrix, but the other challenge was the number of operations involved to simply break it down and reassemble it, which can only be optimized through a keener understanding of the mathematics involved.

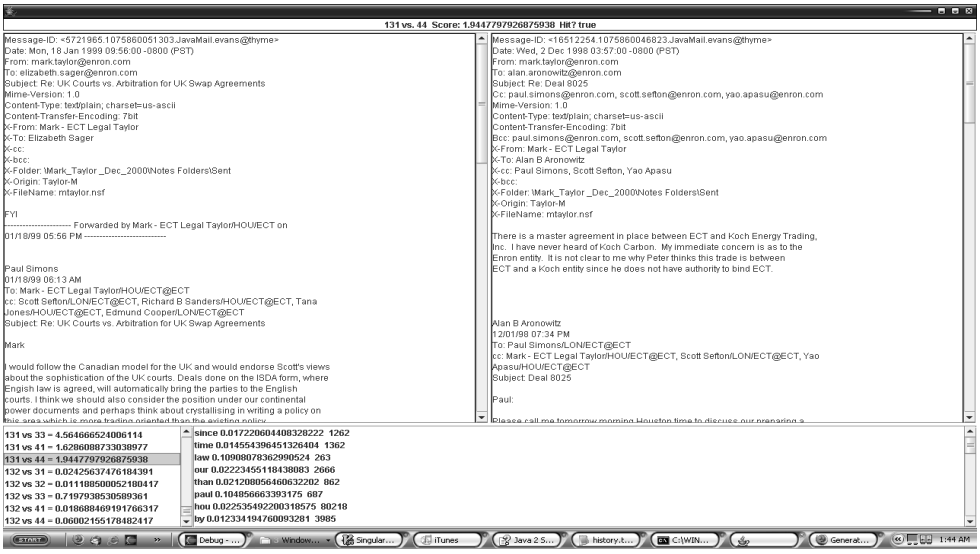

Figure 2.3. A strong false positive example

The number of false positives generated by the algorithm makes our approach particularly difficult to justify use at this point. For example, in figure 2.3, we found two messages analyzed by the system that were reported to have a strong correlation score. Although both

messages should have a heavy amount of similar material, we instead discovered that they spoke of two very different subjects with little in the ways of related content. This is troubling for any security scenario, as even a small amount of false positives could potentially scrutinize a number of otherwise innocent individuals. Clearly, this would defeat the purpose of an automated system beyond simply reducing the task size unless our true negative count was much higher.

One minor troubling issue was the fluctuation of word frequencies at the beginning of our simulation. Until we got past several hundred messages, individual frequencies ranged from 0 to 10 in the global frequency dictionary. Clearly, our system needs to be trained on the linguistics on the target communications group, or simply allowed to calibrate for a few days of traffic. Once enough messages were passed, they settled.

Due to the nature of the problems, we were not able to ascertain whether or not the SVD filtering was successful. We theorize that it was limited by the word frequency problems throughout the experiments, and as such may still be a useful addition. However, the cost of implementation to the resources of the system bring into question whether or not it could be used in a real-time scenario.

2.8 Conclusions

Unfortunately, our system was not a success. While we have learned from this experience in terms of what does not work, we recognize that the problem is inherently difficult. First, the nature of linguistic analysis cannot be boiled down to word choice alone. The grammar of a message appears to be much more important than the content alone, and the traditional bag of words approach does not appear to help with correlation analysis at all.

In order to create a more viable system that can operate successfully on a variety of datasets, one possible direction to explore would be a more thorough word frequency dictionary. By using an available compiled English text corpus, a set of words extracted from an e-mail can be given global word rankings in terms of frequency of use. This would create an adaptive word detection filter that can comprehend when a traditionally unusual word becomes commonplace through widespread use, eliminating potential false positives. This would also address the fluctuation problems we encountered during the experiments.

One problem with this direct approach is that there are a number of words that share the same meaning. For example, the word 'person' has approximately fifteen to twenty meanings, depending on context. While a semantic engine is outside the scope of this system, it is theoretically trivial to use a simple thesaurus database and collapse multiple synonyms into a single common word. Such an addition would make word frequency data much more robust and accurate. This would overcome some of the issues with interpretation of word processing but it is still insufficient to address the grammar issue.

Since such a thesaurus dictionary may not be readily available, an alternative would be the integration of the classic Porter Stemming Algorithm. Developed originally in 1980, the algorithm was developed by Martin Porter in the interest of automatically removing the suffixes attached to words. One scenario involves the conjugation of a simple verb 'run' into the various tenses: 'runs', 'running', 'ran', etc. While the past tense word 'ran' is not properly identified, the rests of the variations are collapsed back into the word 'run' itself. This algorithm alone could eliminate several redundancies in word frequency detection.

Even a perfect keyword recognition algorithm is not sufficient when individuals begin to encode the text of their message. The use of encryption breaking techniques is beyond the

scope of this research. However, assuming that data encryption is never used, there is another alternative to hiding the true topic of a message: word swapping. Keyword analysis is virtually useless when an individual uses unrelated words in place of suspicious words. For example, using the word 'corn' in place of 'bomb' would create a new message that would appear to be innocuously concerned with food.

Given the widespread interest in spam filtering, there are undoubtedly a number of alternative anomaly detection techniques that can be used on text-based communication. Further research is necessary to determine what techniques are available, their effectiveness in detecting characteristics, and whether or not they would be a beneficial addition to the system. For example, the use of artificial intelligence to automatically identify keywords has been discussed in the work of [32].

One area of necessary research can be found in properly modeling temporal decay within the social network. Several questions must be answered: How do implied relationships decay overt time? When does a token become invalid? How does time affect the weights of both nodes and links? What adjustments should be made to the lifespan of a token when used repeatedly? Clearly, much of this research may branch outside of traditional computer science areas.

Another interesting avenue of research is the explicit identification of roles within a network. Within any given social setting, each individual often serves one or more roles in the communication infrastructure. Some may be hubs of information, always kept informed of situations and responsible for informing others. Others may act as brokers between two major parties, a liaison responsible for maintaining a channel of contact. [33] These roles impact how individuals interact, their purpose, and ultimately the way information

disseminates through the network. Such identification could distinguish between low level members of terrorist groups and the leaders themselves. Further research into identification of these roles can enhance anomaly detection by understanding when an individual deviates from the normal purpose they serve.

Groups of individuals, also known as social fields, often form automatically within social environments. [34] Whether by a direct department assignment or simply acquaintance, each individual has a number of associates that he or she frequently communicates with. When a number of individuals share close mutual ties, a group often exists. Recognition of these groups represents a unique challenge in future research. Social groups represent a significant factor in knowledge distribution, and existing techniques for network analysis may prove useful.

A promising older method of analysis is the field of graph clustering. [35] By discovering correlation of nodes through analysis of shared links among tightly grouped users, groups can be discovered. The work of [36], although focusing on the field of internet topology, gives a number of insights on creating such clusters through discovery of how two nodes are indirectly connected.

Regardless of the results, the problem remains interesting in the sense that a comprehension of the social situation may prove to be just as vital as the grammatical comprehension of the information. The sheer amount of data involved in today's communication networks cannot be easily dealt with, and the need for automated surveillance will grow as the traffic continuous to increase. In short, as long as technology continues to advance, there will be a need for research in anomaly detection.

CHAPTER 3

THE EFFECTS OF GLOBAL EIGENTRUST ON LOCAL COMMUNICATION PUNISHMENT STRATEGIES

3.9 Separate Agendas, Common Goals: Forming Alliances for Security

Assume that we have a perfect communications surveillance system that can effectively monitor traffic for any sign of malicious behavior. All traffic within the bounds of the authority in charge of it is being sifted through in real-time for possible threats against national security. Within this ideal scenario, however, there is a serious limitation on the scope of what can be observed. If an attacker chooses to relay information outside of such a network, or relays information of which crosses into the system boundaries but keeps key parts of the context outside of it, the system can still fail to observe the threat. Limitations in the availability of resources to extend the system's scope are further complicated when international boundaries are called into question.

In an era of unprecedented communication infrastructure expansion, the exchange of knowledge has evolved into a market for information. The speed and ease at which data can be shared has prompted many governments, businesses, and other organizations to realize the value of what they know and to guard it appropriately. However, the nature of information also raises the potential that the analysis of two or more pieces of data can yield a far greater value than the combined individual value of each. Thus, there is an additional and often conflicting motivation to collaborate, where multiple organizations to pool their data in the hopes that they will be able to mutually increase the value of what they already have. It is in

this struggle of desires that the challenges of sharing protected knowledge arise. When security itself becomes dependent on advanced knowledge and information, a third motivation complicates the challenge significantly.

Defending against the threats of international terrorism presents this scenario in an alarming manner. Countries that are concerned about an attack on native soil must frequently consider assailants that have collaborators which span the globe. A single country rarely has the necessary resources and jurisdiction to continuously investigate every possible suspect. Even if the resources are available, unless attacks are expected to have a considerable impact, it is not likely that the investment is cost-effective. Thus, a variety of agreements have been created in the course of history to attempt to unite multiple governing bodies under a common threat by exchanging information with their allies.

Unfortunately, in the scope of international politics, the only governing factor that insures members of an alliance will always cooperate fully lies in their individual motivations. Unless data can be verified, there are no guarantees that the information supplied will be truthful. When an exchange of data is made, there exists the potential for increasing gain from a transaction by presenting false knowledge. If the other party's knowledge is truthful, and the malicious party is not caught, a one-sided gain has occurred. Thus, one of the biggest challenges in such endeavors is how to discourage malicious behavior.

The study of game theory deals directly with the motivations of participants, known as players, attempting to achieve some known goal and the choices they must make to do so. Out of all options available, game theory assumes that each participant wishes to maximize their own personal benefit in a rational manner. At any given time in an information exchange, both participants have the option of telling the truth or providing false data. While

it may seem obvious that all parties would collectively benefit from the truth, each individual is often only concerned with their own gain. [37] If that gain comes at the expense of another participant with no threat of retribution, there is little encouragement to do otherwise.

However, when games are repeated, new constraints begin to emerge on a player's strategies. If a participant chooses to lie, they run the risk of being caught, leading to a potential net loss. When a central authority can observe actions and affect the payout a player receives, enforcement of the agreement often becomes a simple manner of finding an appropriate punishment. When considering agreements between multiple sovereign governing entities, there is not necessarily any central authority whatsoever. The burden of ensuring an ideal situation is created is shifted to the collective actions of the group.

One option a player has within such a scenario is to simply refuse to participate with another player. This can include all members, or just a selection of those that are not giving the desirable responses. If one player has information that is highly desirable to the rest, with little dependence on others, they can potentially influence the choices by the entire group. On a level playing field, where no player has information that is significantly more valuable, a single player which no longer communicates with the rest can be sacrificed with little trouble. Clearly, collective action must be taken by a significant number of participants to have any effect on group. Thus, several players must be willing to isolate the same player with undesirable behavior in the hopes that the malicious participant will change their ways.

Another more indirect method of enforcing behavior is to base punishment indirectly on the level of trust shared by the players. Normally, each player already has opinions of the rest, but they lack a broader view of the situation and must assume that how a player deals with them is how they deal with everyone else. EigenTrust provides a means of collectively

allowing each individual player to form accurate opinions of the group by providing ratings of their own experiences.

Several of these factors have already been explored in other works too numerous to mention. However, a factor that has not always been considered is the cost of determining whether or not information is the truth. While there is certainly data that can easily be verified, that nature of certain kinds of information requires a much more in-depth review. Thus, the cost of verification should always be considered in situations where the net gain of the interactions is paramount to the success or failure of an exchange.

The goal of this chapter is to explore the potential of punishment via isolation with regards to trust-based verification. We wish to determine whether or not such methods are viable in large games with multiple players, and consider various scenarios that such logic may face. Success of such a method would prove useful in a variety of decentralized strategy considerations such as peer-to-peer networks and international politics.

3.10 The Challenge of Data Sharing

The scenario for our information exchange strategies is based on a loose alliance with no central authority to enforce behavior. Consider multiple countries in our original scenario which have learned of an impending terrorist attack and face the same challenges. They do not have conclusive data to suggest when or where the attack will occur, but each country has reason to believe it may occur on their own soil. Their objective is to attempt to thwart the current threat. However, given an indefinite time span in which the attack could occur and limited resources, they have each determined they must work together. After discovering each of them had a common goal, they have formed an alliance in which they exchange

information they have collected both at home and abroad. The nature of the game is one which occurs repeatedly for an indefinite amount of time.

Information is exchanged between members of the alliance individually at a regular interval. Each transaction occurs between two countries in such a way that the data is swapped simultaneously; both countries must decide on their strategy before the transaction is complete. This trading occurs between all possible pairs of countries simultaneously, assuming each pair agrees to do work with each other. The value of the information fluctuates within predictable boundaries, and no player has a considerable advantage over the rest.

Each player faces that challenge that they do not know of the kind of behavior the other members will engage in. While all countries involved are assumed to have a common goal, they may also see an opportunity to advance their own agenda. For example, one country may wish to keep what they know a secret from the rest, in the hopes of learning more at no real cost.

The strategies chosen by each country is determined by the overall behavior they have chosen. Each country wishes to find the optimal behavior to reduce the impact of defense on their national budget. We assume thus that countries are willing to adapt by altering the behavior to reflect the one they believe has performed the best. At the same time, as behaviors shift, the payouts of strategy choices may shift as well, leading to a dynamic balance of power. For example, a behavior to always lie may perform well when other countries are not verifying, but as others learn of the benefits of the behavior, others may follow suit. This would result in several liars always lying to each other and never gaining any information.

Determining whether or not the data is received is legitimate is the responsibility of the country itself. Since the data is primarily intelligence, verifying it has a substantial cost due to the resources, manpower, and time required. In our scenario, verification is always less than the value of the information itself, which means consistently doing so will still result in a net gain. However, note that our models remain valid even when verification is much higher than the information itself.

The strategies chosen by each country is determined by the overall behavior they have chosen. Each country wishes to find the optimal strategy to reduce the impact of defense on their national budget. We assume thus that countries are willing to adapt by altering the behavior to reflect the one they believe has performed the best. At the same time, as behaviors shift, the payouts of strategy choices may shift as well, leading to a dynamic balance of power. For example, a behavior to always lie may perform well when other countries are not verifying, but as others learn of the benefits of the behavior, others may follow suit. This would result in several liars always lying to each other and never gaining any information.

The use of adaption to improved behaviors within a game raises an interesting point about the duration of punishment. One option is to punish a deviating agent indefinitely. When this is done, any future benefit from that agent is simply not possible, potentially allowing more forgiving agents to flourish. Instead, punishment in our game is done in such a way that the other player simply loses a significant amount of their own potential earnings, reducing their net gain from the game. This indirectly makes the ideal behaviors much more likely to be chosen. Eventually, if this is practiced widely enough, overall behavioral choices yield an ideal environment where everyone can benefit. Likewise, when we have a fixed

interval when agents may take the opportunity to change behaviors, we would potentially discourage what may otherwise be an excellent source of profit; thus, forgiveness must be performed by all agents during this round. An example of this would be when a government agency has a new leader or a business comes to the end of a fiscal quarter.

Determining reputations within a distributed network can be a difficult endeavor. Since it is possible for a malicious participant to deal honestly with some players and dishonestly with others, a trust value must extend beyond a local perspective. This necessitates querying others for their opinions on opponents within the game, which introduces the possibility of the same malicious agents simply telling others they have an outstanding rating while their peers have just the opposite. This introduces the additional possibility that different players will come to separate conclusions, based on the 'noise' introduced by the subversion. Sepandar et al. at Stanford University devised the EigenTrust algorithm as an answer to these problems. [39]

The algorithm itself is relatively straightforward. Each player queries every other player for their opinion on the rest. This forms a matrix of relative trust, based on a score built from history among individual agents. From here, a normalized matrix is constructed, then evaluated with the Eigenvalue Decomposition technique. When all players perform this properly, they will all come up with the same left-principle eigenvector. This vector represents the Eigentrust rating of each player. The algorithm has been well received as a foundation for more robust distributed systems, though trust itself needs further refinement to be properly defined. [40] In real distributed system deployments, Eigentrust would be done in a distributed fashion.

The basics of each round of our game can be described with an immediate snapshot of the game matrix. There are essentially three choices every player can choose: lie, tell the truth, or stop playing with the other player. The potential benefit from the truth is an average of I_{min} and I_{max}, the upper and lower bounds of what the information is worth. A lie of course carries no value, but checking as to whether a piece is legitimate or not does carry a cost. This cost c_V is directly determined by the probability that verification will occur P_i coupled with the trust t_j assigned to the opponent j. The impact of trust is adjusted by the constant φ, in order to ensure it does not have a huge impact on existing values. However, certain behaviors never consider verification as a possibility, regardless of trust, and as such the payoff has no effect on the result. This is where we use the variable σ to remove these costs from consideration for each player as needed, setting it to 0 when no verification is done and 1 otherwise. Note that, for behaviors themselves, the game is not about how much is gained by the player but rather how much the player gains in relation to their opponent.

Table 3.1. The constructed strategy matrix for our game

		Play		Withdraw	
		Truth	Lie	Truth	Lie
Play	Truth	$\left(\frac{I_{max}+I_{min}}{2}\right)-c_V(\bar{P}_i+\varphi t_j)\sigma_i$ $\left(\frac{I_{max}+I_{min}}{2}\right)-c_V(\bar{P}_j+\varphi t_i)\sigma_j$	$\left(\frac{I_{max}+I_{min}}{2}\right)-c_V(\bar{P}_i+\varphi t_j)\sigma_i$ $-c_V(\bar{P}_j+\varphi t_i)\sigma_j$	0 0	0 0
	Lie	$-c_V(\bar{P}_i+\varphi t_j)\sigma_i$ $\left(\frac{I_{max}+I_{min}}{2}\right)-c_V(\bar{P}_j+\varphi t_i)\sigma_j$	$-c_V(\bar{P}_i+\varphi t_j)\sigma_i$ $-c_V(\bar{P}_j+\varphi t_i)\sigma_j$	0 0	0 0
Withdraw	Truth	0 0	0 0	0 0	0 0
	Lie	0 0	0 0	0 0	0 0

3.10.1 Equilibrium Analysis

We wish to consider the existence of a proof that demonstrates that an equilibrium exists in which *Truth, Truth* will always be selected by both players in an exchange. In earlier

iterations of this work, we utilized a simpler matrix construction that did not account for a more robust consideration of verification calculations. In this game, trust was not considered a part of the equation. Instead, verification rates were varied according to the evolutionary results due to randomly assigned values to those that used *Withdraw* as a strategy for punishment. Otherwise, the original game is identical to our current one. We prove that an ideal equilibrium exists with the original matrix, with greater fluctuations of information value and trust probabilities.

Table 3.2. A simplified version of the game matrix

		Player 1		
		$Truth$	Lie	$Withdraw$
Player 2	$Truth$	$\Delta_t^1 - p_V^2 C_V$ $\Delta_t^2 - p_V^1 C_V$	$\Delta_t^1 - p_V^2 C_V$ $-p_V^1 C_V$	0 0
	Lie	$-p_V^2 C_V$ $\Delta_t^2 - p_V^1 C_V$	$-p_V^2 C_V$ $-p_V^1 C_V$	0 0
	$Withdraw$	0 0	0 0	0 0

In a simplified version of the matrix, we consider the cost of choices in light of a much simpler situation. The value of information for either player is represented by Δ_t^i, where *i* is the player, and *t* is the time at which the game is considered. This is in contrast to the average value of *I* taken from I_{min} and I_{max}. The cost of verification remains at C_V, but the probability of verification is simply p_V^i, with *i* once again representing the player in question. Consider a traditional one-shot game. We must pick a strategy in which we can guarantee our success. Consider *Withdraw,Withdraw* as a natural Nash equilibrium. At first glance, this would appear to be a poor choice. Clearly, better payoffs are found in *Truth, Truth*. However, if we choose *Truth* as our strategy of choice in this setup, the other player can choose *Lie* as it increases their utility. If we choose *Lie* instead, we can take advantage of another player's

trust. Should they choose *Truth* and deviate from the equilibrium, their payoff will dramatically decrease while ours increases; at *Lie*, our payoff is as we expected. Withdraw of course neutralized both results. Thus, a Nash equilibrium exists at *Withdraw,Withdraw*.

In practice, not all games are classified as one-shot. Some involve players that play the same game multiple times. Such games enable players to use past data to both predict their opponent's behavior and even affect a particular outcome. In our model, the "data sharing" game will be played many many times by the participating agents. This scenario can be easily modeled by the "repeated game" ideas from game theory literature. [44] The main observation in repeated games is that the honest behavior in games like the "data sharing" game can be enforced if the game continues to be played with probability $\delta > 0$. In other words, if there are possible future gains, (i.e. if game continues with some probability) each agent can be motivated to be truthful. We can define the expected payoff for a player i participating in the repeated "data sharing" game as the

$$u_i = (1-\delta)\sum_{t=0}^{\infty} \delta^t \cdot g_i(\sigma_i^t, \sigma_{-i}^t) \tag{3.1}$$

where $\sigma^t = (\sigma_i^t, \sigma_{-i}^t)$ is the strategy employed at time *t,* δ is the halting probability of the game, and g_i is the gain achieved at each play of the "data sharing" game. Let $u = (v_1, v_2)$ be the payoff vector of the repeated game. Note that if every period $g_i(\sigma_i^t, \sigma_{-i}^t)$ is equal to some u then u_i will be equal to u.

To illustrate, consider an instance of the game between two intelligence agencies a_1 and a_2 at some point in time on round t. From the perspective of a_1, σ_{-i}^t is expected to be *Truth* for a_2 since $\sigma_{-i}^{t-1}, \sigma_{-i}^{t-2}, \ldots, \sigma_{-i}^1$ have all been *Truth* as well. According to this equation, we should expect the maximum utility of u for *Truth, Truth*. However, a_i could have a behavior that tries

to deviate at round t if a_2 has proven trustworthy. In this instance, σ_i will be *Lie*, and *v* will be greater than *Truth, Truth*.

Below we prove that our repeated "data sharing" game can be used to enforce truthful behavior by refusing the deal with dishonest agents that caught cheating. Our proof technique is very similar to the one used for proving "Nash Folk" theorem from the repeated game theory literature [6]. Our main difference as compared to the generic Nash Folk theorem is that in our case opponent's actions could not be observed unless a party to choose to verify the correctness of the data. Given the above "data sharing" game, we can prove that truth telling emerges as a Nash equilibrium as follows:

Theorem 4.1 *If telling the truth each round has a gain $g_i > 0$ for both parties then there exits $0 < \delta < 1$ such that telling the truth for both parties is a Nash Equilibrium for "data sharing" game.*

Proof

We will prove that utility of telling the truth given that the other party tells the truth is bigger than any other strategy that lies with some probability *p*. To see that let us calculate the expected gain of a given party who chooses to lie with probability $p > 0$ at each round. Note that in a given round with probability $(1-p)$ he will gain $g_{T,T}$ (i.e. the gain achieved when both party tells the truth) and with probability p he will gain $g_{L,T}$ (i.e. the gain achieved when he lies while the other party is telling the truth). If he cheats and is caught, he will earn zero for the rest of the game; otherwise, a new round starts. Under these observations, and using the fact that in our game $g_{T,T}$ is equal to $g_{L,T}$, we can write the total expected utility of lying

with probability p given that the other party verifies the correctness of the received data with probability q as

(3.2)

$$u_i = (1-p)\cdot g_{T,T} + p\cdot g_{L,T} + (1-p\cdot q)\cdot \delta\cdot u_i$$
$$= \frac{(1-p)\cdot g_{T,T} + p\cdot g_{L,T}}{1-(1-p\cdot q)\cdot\delta} = \frac{g_{T,T}}{1-(1-p\cdot q)\cdot\delta}$$

Similarly we can write the utility of always telling the truth (denoted as u_{Ti} below) if the other party tells the truth as

(3.3)

$$u_i^T = g_{T,T} + \delta\cdot u_i^T$$
$$= \frac{g_{T,T}}{1-\delta}$$

Note that if $q > 0$, then $u_i^T > u_i$, for $p > 0$ *and* $u_i^T = u_i$ if $p = 0$. Therefore, for any given 1>δ>0, telling the truth will be a Nash equilibrium because each party has no incentive to lie given that the other party is telling the truth.

End Proof

Although this proof was originally constructed with a simpler version of our current game theory matrix, it continues to hold true in light of the addition of trust-based verifications rates. This considered, we still believe there is a need to explore the outcome of simulated games. First, we assume that the value of information will fluctuate over time, due to the market nature of the value of data and what may already be known by either country.

Second, we are also interested in the benefits of behaviors as to their relative utility gains in the course of the simulation, especially as it applies to an evolutionary approach. Finally, we assume that not all players will be completely rational. Thus, they may still attempt to circumvent the ideal situation to further their own personal gains for several reasons, including deliberate sabotage, freeloading, or other malicious motivations.

3.10.2 EigenTrust

In order to calculate trust, we use the EigenTrust algorithm for calculating trust among participants. Originally designed to deal with reputation management in peer to peer networks, the algorithm allows each participant to calculate a global trust value for each player locally. Each player constructs this trust by querying every player for their opinion of every other player. These results are placed in a matrix and normalized to ensure all ratings are between 0 and 1, and then used as a reputation reference. The calculations can be done globally or in a distributed fashion. In both situations, all players still arrive at the same reputation ratings as long as a sufficient number of players are participating. This even holds when malicious players attempt to inflate their personal ratings and report poor scores for the rest of the players, as long as measures are taken according to the original author's work. [45] In our experiments, we base trust on how frequently players have been confirmed as liars in regards to the number of exchanges made. If no exchanges are being made, then the rating for the other player that has been cut-off is as low as possible. Each player provides the option to be queried for the rating at no cost to the asking player. We calculate the trust globally at the beginning of each round to simulate the distributed algorithm. For simplicity, we assume that some unspecified system is in place to ensure that any malicious rating

reports do not affect the calculation. The resulting reputation vector is then available to all players.

A player $\underline{i}$ has an opinion of another player j in the form of a numerical value s_{ij}. This essentially provides a snapshot consideration of a player's past dealings. For our scenario, we define s_{ij} as follows:

$$s_{ij} = \begin{cases} \sigma_t^{ij} \neq Withdraw & \dot{s}_{ij}(t) \\ \text{otherwise,} & 0.5 \end{cases} \tag{3.4}$$

where σ_t^{ij} represents the strategy player i has chosen for dealing with player j at time t. If the j is being punished by selected *Withdraw*, we automatically default the opinion to a neutral value. Otherwise, we use a calculation on the reputation of j at time t according to a separate function that considers the entire history up to the 'present':

$$\dot{s}_{ij}(n) = \frac{\sum_{t=0}^{n} \begin{cases} \sigma_t^{ji} = Lie \wedge \{t, j, i\} \in h_t, & 1 \\ \text{otherwise,} & 0 \end{cases}}{\sum_{t=0}^{n} \begin{cases} \sigma_t^{ij} \in \{Truth, Lie\}, & 1 \\ \text{otherwise,} & 0 \end{cases}} \tag{3.5}$$

Here, we take a look at all choices made by j as it has dealt with i up to the specified time period n. We sum the number of times j has told a lie according to the history that i has

recorded. In other words, as long as *i* has verified the information and recorded it in it's personal history h_i at that point in time, we can count it as a positive experience. Otherwise, we assume the worse and assign it a rating of 0. The sum here is then divided by the number of times a choice has been made to deal and not withdraw from *j*. Since none of the behaviors never start by withdrawal and will provide at least one opportunity to their fellow partners to exchange information, we will never have a dominator of 0. Note that since we are counting lies, the rating will actually be an inverse of the result that would normally be calculated in order to directly dictate that a less reliable individual will be more likely to be checked, and vice versa. See table 3.1 to understand the reason this decision was made.

We do not consider the history of the entire game in h_i. When the specified number of rounds has passed, and behaviors have been modified, we consider the history erased, the time 'reset', and continue the game with a fresh start for all players. This is done due to the lack of "meta behaviors" that motivate players our simulations beyond the desire to select the best behavior. In essence, we assume all players are naïve in the sense that their previously malicious strategy choices were due to an 'innocently' selected behavior.

The EigenTrust algorithm we used was relatively simple, and accomplished by using the COLT Library for Java to compute it. [46] The core algorithm is relatively simple when compared to its' distributed counterpart:

$\vec{t}^{(0)} = \vec{e}$;
repeat
  $\vec{t}^{(k+1)} = C^T \vec{t}^{(k)}$;
  $\delta = ||t^{(k+1)} - t^k||$;
until $\delta < \epsilon$;

Figure 3.1. The original author's algorithm to derive EigenTrust. [45]

Figure 4.1 is an excerpt from the original author's work and outlines the algorithm. Starting with an initial trust vector *t*, set to a uniform trust assignment vector *e* which has the same value for all players, the vector is continually modified using an aggregated matrix *C* of all individual opinions of reputation. It is computed as follows:

Here, c_{ij} represents the normalized value opinion player *i* has of *j* with respect to the opinions held by *i* of the rest of the players. Thus, it becomes a percentage of trust assigned to a particular player, and as such every entry in *C* is between 0 and 1. This is very important when opinions differ widely or disparity occurs between those that have been cutoff and those that have not. However, this is still insufficient for the EigenTrust metric, as we must now take into account the reputation other players have of each other. We compute an aggregated value of trust according to equation 3.7.

(3.6)

(3.7)

$$c_{ij} = \frac{\max(s_{ij}, 0)}{\sum_j \max(s_{ij}, 0)}$$

$$t_{ik} = \sum_j c_{ij} c_{jk}$$

The entry in matrix *T* yields a value t_{ik} that is aggregated from the rest of the players. In essence, every player asks every other player their opinions of each other in a recursive fashion. This forms a global perspective of trust amongst the players to form a more accurate

snapshot of player behavior. Factoring this back into the algorithm depicted in Figure 3.1, we repeatedly modify our initial vector until we arrive at the same conclusion for each player as equation 3.7 describes, as each row i in T will ultimately converge to trust vector t specified in the algorithm.

3.10.3 Behaviors

The Honest behavior takes a naïve approach to other players. Truth is the only strategy ever chosen, and it never verifies the strategies of other players. It has the advantage of never incurring the cost of verification, and it always maximizes the potential gains with other players by never severing the links. An example may be a country that wishes to set an example, or perhaps is simply under significant amounts of scrutiny. While this may prevail against behaviors that perform even the slightest verification, they will always lose in a competition with the Dishonest behavior. Essentially serving as the opposite of the Honest behavior, this behavior simply chooses the Lie strategy regardless of the outcome.

Not every player may believe that a predictable behavior is optimal. The Random behavior picks either Truth or Lie with equal probability. No punishment or verification is ever performed. Countries that wish to avoid being anticipated may choose this strategy. It carries the same benefits as the Dishonest behavior, but potentially only gains at most half the benefit.

During the course of our research, we encountered a unique yet simple approach to dealing with undesirable behavior known as the Tit-for-Tat behavior. Devised by Anatol Rapoport for a cooperative game theory contest, it follows a simple strategy selection process. Initially, it selects the desirable strategy, Truth. From that point on, it simply selects the same strategy as its' opponent within the game. [3] This was proven quite effective against

all but the most sophisticated collaborative opponents. [41] In our simulation, however, being able to mimic the opponent's actions requires constant verification.

Our own contribution to existing behaviors is the LivingAgent. Initially, like Tit-for-Tat, the Truth strategy is chosen. During each transaction, there is a probability P_V that the player will verify whether or not the other player told the truth. If a lie was told, the other player is punished by severing the link for R_S rounds. This is a sacrifice in the sense that, if the other player is telling the truth for at least part of the time, further opportunity will be lost. The goal with this behavior is to place a high price on deviation. A country which behaves in this manner may be attempting to send a message to the rest of the participants, or may simply be unwilling to waste time with participants whom are equally unwilling to share valid information.

A variant on this behavior is the SubtleLiar, which obeys the same principle but has a threshold in which it will automatically choose the Lie strategy. The net effect is a behavior which can take advantage of a low P_V and net a slightly larger gain in information. An example of this behavior may be countries that believe their fellow members trust them enough to be taken advantage of.

Finally, we have the Liar. This behavior is almost identical to that of LivingAgent, with one notable exception. Is called Liar simply because it essentially tries to pass itself off as a honest participant while consistently trying to take advantage of the right situations. Assuming that the value of the information about to be received is known in advance by both, the Liar will always lie if the received value is within a certain threshold of I_{max}. Thus, this agent appears to capitalize on advanced information by attempting to only take a risk when the gain appears to be sufficient. This threshold is determined by the constant $I_{valuable}$.

3.11 Setting up the Scenario

Our experimental setup involves creating an alliance of 100 virtual countries. They begin with equal levels of trust, and hold all of their peers in the same regard. Each experiment begins by distributing initial behaviors based on a configuration file. The behaviors are assigned to each player based on the distribution specified within the file. During each round of the simulation, a transaction is executed between all possible combinations of players through a virtual link. After this round, the trust metrics are updated, history files are recorded, and agents receive their payoff based on the results. This payoff is used to directly determine the performance of the agent itself. The value of the information varies between constants I_{min} and I_{max} , set to 3 and 7 respectively. If the information provided is false, it has no value. In the event verification is performed, it comes at a cost C_V, set to a value of 2. Thus, even if all information is verified, a net gain is still possible. When verification occurs, and no lie was told, it is noted as waste. The overall score a player receives is simply the total value of all information sans any cost incurred. The threshold for Liar to lie is at 6.9.

All players are assumed to be willing to change their behavior to a more effective one, based on the performance of their neighbors. To simulate this, every 5,000 rounds, each agent is assigned a new behavior based on a weighted probability assigned based on the total gain achieved. For example, if the Honest agent has an total net payoff of 10,000, while Dishonest has a total net payoff of 20,000, each agent has a 33% chance of choosing Honest and a 67% chance of choosing Dishonest. Thus, the new distribution reflects the relative performance of all agents. Note that while this method does not necessarily guarantee an ineffective behavior will be eliminated, it will ensure that any effective choice will be much more likely to emerge victorious. The simulation ends when either 100,000 rounds have

passed or all players have chosen the same behavior, the latter considered a ‘win’ by the behavior adopted by the rest.

There are several verification rates possible for a LivingAgent-derived behavior. To ensure a larger search space is explored during the experiment, a small mutation rate is introduced on P_V for each player behavior that is copied. This allows players to adapt over time and consider reducing the potential waste as the system approaches an equilibrium in which all participants tell the truth.

3.12 Results

3.12.1 Liar War

The LivingAgent performed admirably against Tit-for-Tat, even when the latter behavior began in the experiment with twice as many players adopting it. Out of all experiments, neither the Dishonest nor the Tit-for-Tat behavior ever successfully became the dominant behavior. However, the LivingAgent behavior only won the game 87% of the time, while Liar won the rest. The average verification rate was 14.8% in the final behavior tally, while the average standard deviation was only 2%, taking an average of 15 generations to declare a winner.

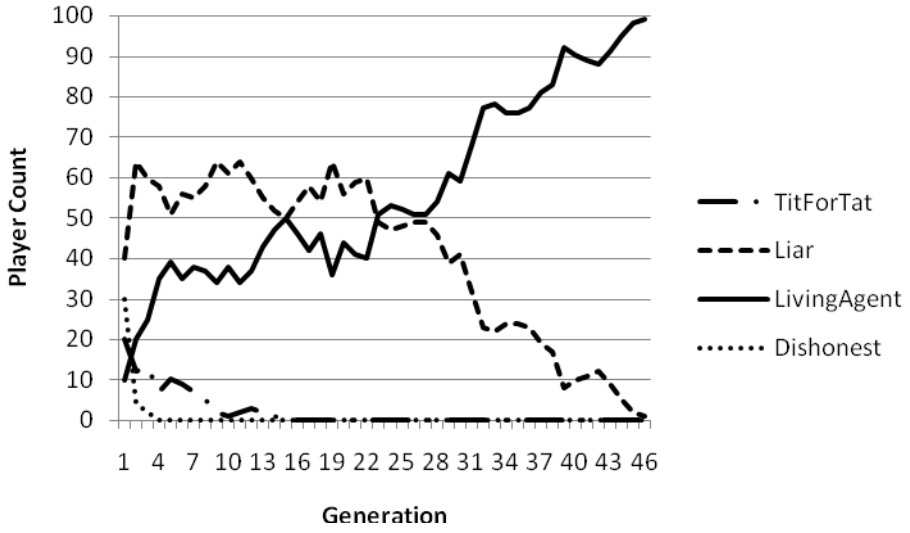

Figure 3.2. Graph of LivingAgent performance against malicious agents and TitForTat

In figure 4.2, we see that LivingAgent successfully achieved the majority behavior despite beginning as a much smaller population. At first, Liar benefits from the fact that it and the rest of the behaviors engage in punishment. Those that leave the Dishonest behavior go towards both LivingAgent and Liar, with more towards Liar. However, Dishonest is no longer being used, Tit-for-Tat begins to help LivingAgent by working slightly against Liar. A trend in the results is that once LivingAgent achieves half of the population, the Liar behavior rapidly loses ground to the point of complete loss. This appears to be the critical mass for the behavior in such a situation.

When Tit-for-Tat was able to sustain itself for at least 10 generations, LivingAgent often benefited from this indirectly. The Liar players, even when they initially surged ahead, would usually observe that Tit-for-Tat was a better choice. As Tit-for-Tat increased, LivingAgent took some losses, but the efforts ended up working in concert to reduce the threat of a lying behavior. Ultimately, once Tit-for-Tat was no longer in play, LivingAgent needed only to compete with a much smaller pool of malicious players.

Another trend that made itself apparent is that the verification rates of the LivingAgent did not correlate directly with the number of malicious agents present. In many instances, as the number of Liars increased, the average rate continued to drop. In these cases, it appears that players simply cannot afford to refused business with other players that ran the risk of lying. Essentially, the punishment method ended up only punishing the enforcer.

3.12.2 No Living Agents

In order to demonstrate the effectiveness of the LivingAgent approach, we ran experiments involving all behaviors not derived from it. The behaviors here were only Dishonest, Honest, Tit-for-Tat, and Random. The first thing we noticed is that there was automatically a large increase in the number of iterations necessary. The winner was not always clear, and it appeared that the fluctuation in the payoffs alone caused some agents to benefit more than others. The biggest competitor to Honest was strangely Dishonest, even though both used the polar extremes of selecting a strategy, and neither made any efforts to check information validity.

Another issue that arose was how quickly Tit-For-Tat was eliminated at times within the first generations. This, however, was no surprise; the verification costs no doubt came at a high toll to the payouts. It contributed to the game by removing malicious behaviors consistently, but this often only resulted in a surge of Honest behavior adoptions. Once Tit-for-Tat was eliminated, malicious behaviors again rose substantially in numbers.

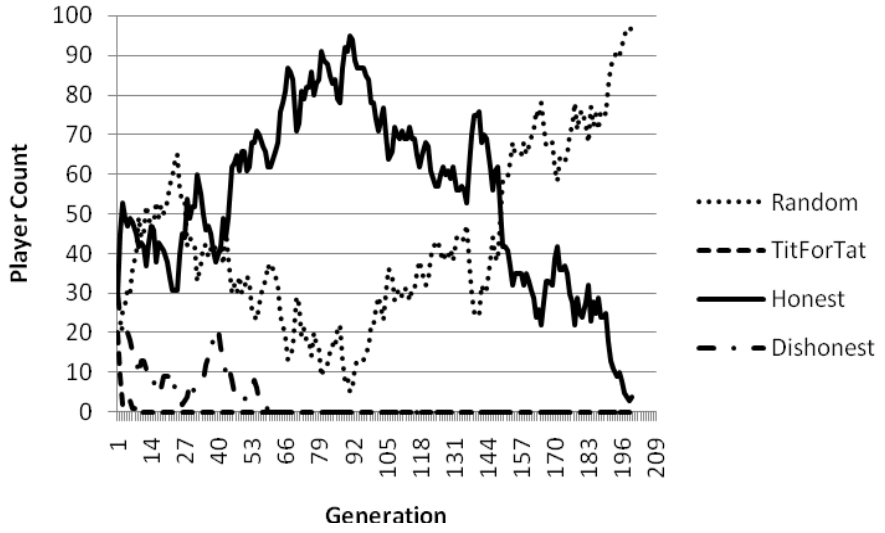

Figure 3.3. Performance of Honest and Random behaviors, Experiment 2

In experiment 77 for this collection, we find that the Random behavior has won. Again, Tit-for-Tat helps Honest surge ahead briefly, but the verification costs cause it to fail to function after only 7 generations. From that point on, Honest appears to be the certain victor, eliminating Dishonest. At generation 92, Random begins to succeed. Essentially, because Random does not discriminate against which players it lies to, it runs the risk of dropping off at just shy of 50% of the player market. This happens twice during the simulation, but due to fluctuations in information value, it eventually achieves victory. Note that this particular equilibrium took 197 generations to achieve, and it was primarily based on the delta in the value gained.

There were only three winning behaviors out of all the experiments. The Honest behavior only achieved the majority 26.1% of the time. Out of those instances, only 84% of them actually resulted in an equilibrium behavior. The Random behavior faired equally well, with only a slightly smaller number of wins at 23.3%. However, the Dishonest behavior beat both of them twice as often at 53.3% as either of them.

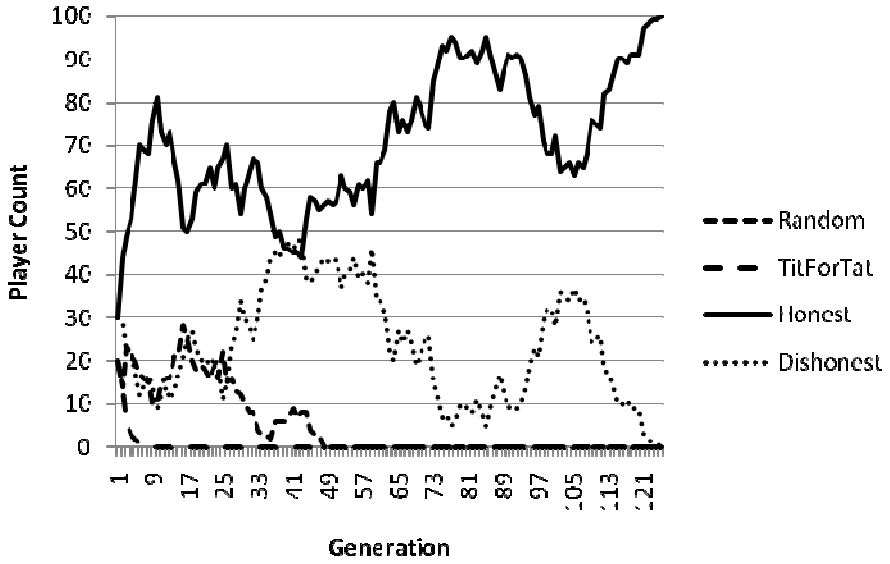

Figure 3.4. Performance of Honest and Random behaviors, Experiment 1

3.12.3 Mixed Environment

In reality, players within these games can vary widely in their ulterior motives, beliefs, and decisions. To observe this, we wanted to observe all of the devised behaviors in action. Unsurprisingly, Honest won 99% of all games played. The abundance of malicious agents, coupled with a roughly 3:1 starting ratio to the Honest behavior, allows LivingAgent to flourish briefly. However, as malicious agents begin to disappear due to lack of relative performance, the appeal of the Honest approach eventually coerces a majority of agents away from it. The loss of Dishonest and Random predictably caused a 25% drop in LivingAgent's presence.

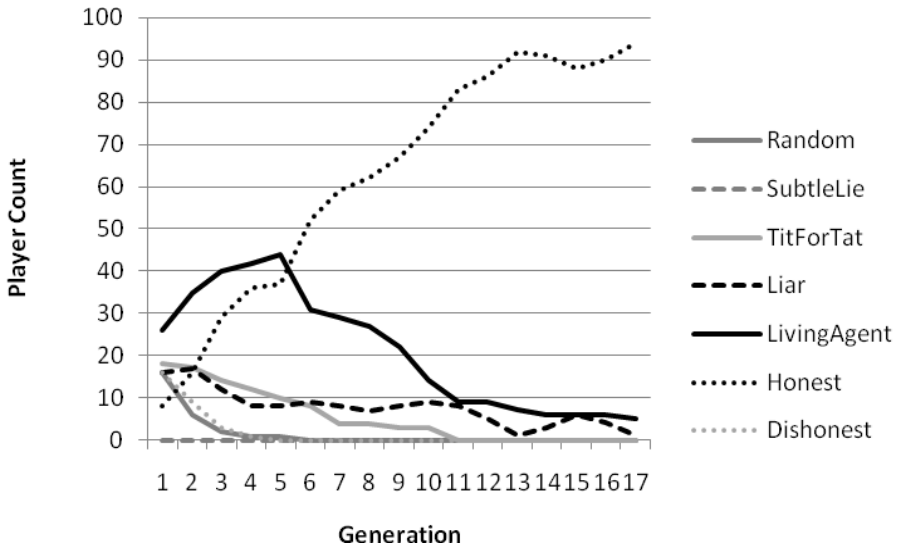

Figure 3.5. Simulation of all behavioral models

On the surface, it may appear that our own LivingAgent is a failure under these circumstances. If our goal was simply to find the 'perfect' behavior, this would indeed be true. However, our constructed behavior helps to create an environment in which the Honest behavior can flourish. Thus, the end result is still the ideal, truth-telling environment. Since previous results demonstrated that Honest would normally fail against the malicious competitors, the introduction of our behavior has acted as an indirect policing force within the system. The Honest behavior achieved the majority roughly 97% of the time within our initial experiments, over three times what it achieved under similar conditions on its' own.

Additional experiments were performed to observe the minimum behavioral mix necessary to ensure Honest would succeed, performed in the form of ratios between LivingAgent and Honest. When equal parts of both behaviors were present, Honest won 86% of the time. Increasing the ratio of LivingAgent to Honest to 2:1, it increased to 92% of the time. At a ratio of 3:1, an effectiveness of nearly 100% was achieved. Thus, although LivingAgent

benefited the rest of the group in achieving a Truth-telling majority, a significant number of the agents needed to have this behavior to guarantee it.

3.13 Conclusions

The overall experiment was a relative success. When enough players choose a behavior that reflects our approach to punishment, the malicious behaviors were successfully eliminated from consideration. The underlying nature of LivingAgent allowed it to defeat even variants of its' own behavior involving light amounts of deviation. However, the same nature of the persistent verification meant that the behavior did not succeed against the Honest behavior, which performed no verification whatsoever despite the circumstances. Even in this scenario, the ideal situation still arose, allowing players to conclude that honesty is indeed the best choice.

However, there remains an undesirable scenario in which our equilibrium does not hold. Consider several players which have decided to adopt behaviors which are truthful up to time n. At time $n+1$, they revert to telling nothing but lies. If the rest of the players have reached a point at which no policing behaviors such as the LivingAgent are left, then there exists the possibility that they will subvert the ideal situation by deviating. Thus, the ideal situation requires several players to make a continual sacrifice for the group by adopting the LivingAgent strategy with regards to the losses which would otherwise be incurred after n.

We hope to increase the robustness of our punishment method in scenarios closer to reality. In real life, information cannot always be verified with 100% accuracy, nor do even the best intelligence agencies guarantee that information provided will be completely true. Such inadvertent mistakes would result in a system which potentially punish otherwise trustworthy

players. This can be addressed with a mixture of a higher tolerance for lies and a slightly more relaxed punishment.

The growing size of networks such as the internet and the increasing use of distributed systems suggest that centralized authority approaches will be insufficient. Insuring behavioral choices by members of peer-to-peer networks requires an approach which can scale as much as the system itself. We believe our work offers a solution to the problem of encouraging behavior when players become responsible for their own outcome.

The nature of international politics and the variety of societal mindsets across the nations of this era serves as a subtle reminder that game theory must always account for varied mindsets. The agendas which emerge from centuries of historical affiliations and conflict can often result in situations that seem to escape logical reasoning without the appropriate context. However, when the security of a nation is in peril, these factors must be accommodated in such a way that permits the flow of information in a timely manner.

CHAPTER 4

SIMULATING BIOTERRORISM THROUGH EPIDEMIOLOGY APPROXIMATION

4.14 Biological vs. Traditional Attacks

Much of the motivation in cooperation between countries at this point in time is due to a mutual desire to deal with terrorism. Although some organizations pose only a domestic threat, many terrorist groups have operations which span the world. Intelligence is shared in these circumstances to avoid deadly attacks which threaten the safety of their citizens. However, the majority of the attacks carried out up to this point of history have been limited to the original victims. Bioterrorism potentially affects far more than just the initial targets, and poses an even greater threat than 'traditional' terrorism acts. Even when the exchange of information is successful, even the groups themselves which plan to be responsible for such an act may be unaware of the implications their attack could have. Understanding the impact can provide a significant advantage for a defending country. [42] To date, no known terrorist acts involving biological agents has spread beyond the initial attack itself.

However, the lack of full-scale incidents of biological warfare in recent world history has created a difficult situation for those attempting to prepare for an epidemic. Several isolated incidents involving potential weapons, such as the SARS outbreak in Canada [43], provide glimpses into how a situation might unfold. Rigorous study of diseases and agents that could be used is undeniably helpful, but the need for data remains. The only safe means by which this data can be provided is through the use of epidemiological simulations.

A traditional but popular approach for simulating epidemics is the SIR model. This compartmental mathematical approach is based on probabilistic transitions over the passage of time. It is considered one of the most widely used models in epidemiology [21]. Individuals are modeled by assigning them to one of three states: susceptible, infected, and recovered. A susceptible individual is considered one whom can be infected. They have no natural or artificial immunity to the contagion. An infected individual is considered stricken with a disease and contagious to those that are susceptible. Eventually, an infected person will no longer be contagious by transitioning to 'recovered', either by successfully overcoming the disease and acquiring immunity or dying. The transitions are based on three factors: the probability of contact between susceptible and infected groups, the rate of infection, and the rate of recovery. Any considerations of intervention or exception must be captured through these rates.

Although popular, the SIR model falls short in several key areas. First, all participants are considered identical in terms of susceptibility. Someone whom interacts with a variety of individuals on a daily basis is considered just as susceptible as someone whom does not interact at all. Second, the area being simulated is completely homogenous. There are no concepts in the model of locations, transportation, or variance in rates. Finally, the model requires a broad generalization of infection rates and the percentage of a population in any state. Complex dynamic or dependent factors simply cannot be accounted for without collapsing them into a single dimension.

4.15 Our Work

Our own model is a hybridization of social interactions on a household scale, situation intervention, and the simplicity of the SIR approach. The system arose out of a need for a

deterministic model that can balance a desire for accuracy in representing a potential scenario with computational resources and time. Recent work has suggested that more detailed models of social networks have a diminished role over the results in the spread of an epidemic. [14] We believe we can generalize complex interactions into a much more concise simulation without adversely affecting accuracy. The ultimate goal of our research is to integrate a model for biological warfare with a system that can evaluate multiple attacks with respect to passive and active defenses. As a result, we have created a simulation that serves as an approximation of the impact of a biological attack with speed in mind, allowing us to explore a large search space in a relatively shorter amount of time as compared to existing detailed models.

The base component of the simulation is the home unit. A home can range in size from a single individual to a large household. Within this unit, the probable states of the individuals are tracked via a single vector of susceptibility, infection, and recovery. Given a population distribution of a region and basic statistical data, we can easily create a series of family units that represent the basic social components from a rural community to a major metropolitan area. A single home unit with no interaction is essentially a basic representation of the SIR model.

Interaction occurs within what we call social network theaters. A theater is essentially any gathering area at which two or more members of a home unit meet. The probability of interaction depends on the type of location and the social interaction possible at it. To capture this, separate infection rates are assignable to each theater.

In the event of a life-threatening scenario such as a bioterrorist attack, we assume a civil authority will act at some point to prevent a full-scale epidemic. We model such an entity by

providing means in our models to affect social theaters and the probabilities associated with state transitions. For simplicity at this point, we will not consider resource constraints, nor will we model how an event is detected. The recognition of an attack will be simulated using a variable delay. After this delay has passed, the infection is officially recognized.

Several known types of options are available deal with an epidemic. [19] [44] [45] The most basic form of prevention is by inoculating the population against an expected contagion. Several options exist at this level, ranging from key personnel to entire cities. Anyone inoculated is automatically considered recovered. Second, a quarantine strategy can be used to isolate the infected population from the susceptible population. This requires the explicit removal of individuals from home units to appropriate facilities, and can be simulated on a fractional basis representing probability of removal with varying levels of accuracy. Third, the infection and recovery rates can be altered, through such means as allocating more resources to medical personnel and educating the general public on means to avoid infection. Finally, a potentially controversial but interesting option is the isolation of communities by temporarily eliminating social gathering areas. For example, public schools could be closed, or martial law could be declared. The motivating factor is finding ways to force the population at risk to remain at home. Such methods could reduce the number of vectors over which an infection could spread.

A day is represented by multiple distinct time periods. We use a modified form of the SIR model to represent the various contributions by the involved members of home units, as seen in equation set 4.1. We have broken down the contributions by both those participating in a theater and those at home during each segment as α and β, respectively.

$$
\begin{aligned}
&\alpha_i(t+1) = \sum_{k=1}^{m_i} \begin{cases} W(t,k) = 1, & \Lambda(t,\lambda_k)\left(\sum_{j=1}^{n_k} \frac{S_j(t)}{h_j}\right)\left(\sum_{j=1}^{n_k} \frac{I_j(t)}{h_j}\right)\left(\frac{1}{n_k}\right)^2\left(\frac{S_i(t)}{h_i}\right) \\ \text{otherwise}, & 0 \end{cases} \\
&\beta_i(t+1) = \Lambda(t,\lambda_k)\left(\frac{S_i}{h_i}\right)\left(\frac{I_i}{h_i}\right)\left(1 - \frac{\sum_{k=1}^{m_i} W(t,i,k)}{h_i}\right) S_i \\
&S_i(t+1) = S_i(t) - \alpha(t+1) - \beta(t+1) \\
&I_i(t+1) = I_i(t) + a_i(t+1) + \beta(t+1) - \Gamma(t,\gamma) I_i(t) \\
&R_i(t+1) = R_i(t) + \Gamma(t,\gamma) I_i(t)
\end{aligned}
\tag{4.1}
$$

The current time is represented as *t*. Individually, each home unit as previously mentioned is actually a self-contained SIR model representation. However, since members of the home unit can participate in social theaters, the calculations required to update the model are influenced directly and indirectly by the rest of the units in the simulation.

The first function, a_i, determines how many susceptible individuals have become infectious in home unit *i* due to social theaters. Every one of the m_i theaters which *i* participates in must be considered. For each theater *k* in the set, there are n_k participants. The *W* function, given time *t*, home unit *i*, and work theater *k* of *i*, returns 1 if the home unit participates and 0 if it does not. This allows us to distinguish types of theaters across time. No participation means that the specified theater does not impact any result during *t*.

Next, we must consider the individual infection rate for theater *k*, as specified by λ_k. We assume that the infection rate starts out at some level of 'normal', uninhibited probability. However, at some point in time, there exists the possibility that a civil authority will step in and decrease this factor by bolstering public awareness, providing necessary information to hospitals to enhance treatment, etc. We represent this by adding a new function $\Lambda(t,\lambda_k)$ which can factor in both the custom rate and any modifiers we need, applying them at a predetermined time to represent detection and subsequent intervention.

Now, we must consider the probability that we will have a susceptible individual interact with an infected individual. The size of each household is h_j of the n_k households present in

the theater. We take the susceptible population $S_j(t)$ of j, divide it by the total size of j, and sum the results. This is repeated for $I_j(t)$. Note that the sum for both populations is an actual head count; hence the need to convert the total into a probability for each. We do this by dividing each summation result by n_k, the size of theater k. The product the results is the probability a susceptible individual will encounter an infected individual at the site.

The next step is to determine the part of the household being affected. This is obtained by taking the susceptible population of the home unit $S_i(t)$ and dividing it by the size of the home h_i. Note that this does not entirely compartmentalize participants from each home unit and suggests that a single individual from the household is selected with equal probability to participate. We allow this to simplify the calculations necessary to avoid tracking each individual separately in a pure agent-based format.

Finally, we consider whether or not social theater k is going to impact household i. We base this on three factors. The time of day determines which time segment the simulation is in and which theaters the home units are involved in. The household i determines which theater participation set is being referenced by k. The result of this function is 1 if the household will be involved, and 0 if it is not.

The second calculation β_i determines how many people, during the time segment at t, stayed home and participated in the home unit update model. Most of our calculations here are identical to those of the original SIR model. We calculate the probability that a susceptible person at home will become infected by those already carrying the contagion. However, we also must calculate the portion of the individual updated by considering the fraction not present. We thus determine the number of people not present, divide it by the total number of people in the home unit, and invert the percentage by subtracting it from 1. We implement

a constraint on the W function, requiring that a home unit can never participate in more theaters than there are members h_i. This allows us to sum up this participation by type assigned to a given time period.

The calculations for recovery of infected individuals are much simpler. Regardless of theaters that they participate in, individuals recover at a uniform rate as determined by $\Gamma(t, \gamma)$ during the course of the simulation, in a similar fashion to Λ. We assume that it is possible for a civil authority to affect the base recovery rate γ. For example, a state can bolster the level of hospital care by bringing in doctors from other locations, providing emergency resources to the existing facilities. Other possibilities include direct emergency funding and simply making a medical examiner aware of what symptoms to look for.

When no social theaters are considered in use, the simulation enters a 'home' period. During this portion of the day, the members of the home units only interact internally. The W function returns 0 for any work entry at this time. Essentially, each home unit is calculated as if it were an entirely self-contained SIR model.

It is important to note the use of 'fuzzy' states in the home units. The three population states are considered probabilities of the household's overall condition. For example, a susceptible value of 80% and an infection value of 20% mean that there is a 1 out of 5 chance for each member of the household to be sick. Note that a small attack can potentially infect the same size of a population as a larger attack when no intervention is present; there may simply be a smaller chance that each member of the population is infected. Note that within a unit there is not necessarily a distinction in how many members are infected.

4.16 Experimental Setup

4.16.1 Population Generation

For our experiments, we wanted to have a variety of home units with a predictable set of sizes based on household statistics. Starting with a population of 10,000 we divided them into groups ranging from a single individual to a family of size 6. The biggest set was the three person household. Exactly 5,165 total households were generated from this pool of individuals and used consistently throughout all of our experiments.

4.16.2 Theater Generation

The social theaters are generated by the population size that they will ultimately contain. The assignments themselves are divided based on power law averages, used to generate random sizes. To keep these assignments consistent, we use a pre-determined seed in an isolated random number generator for each distribution session. For work theaters, we have a maximum population of 5,000 spread of theaters randomly generated by power law from size 1 to 15^2. Education and recreation theaters are allocated 2,000 people each, with maximum sizes of 900 and 25, respectively. These assignments give us 57 theaters for work, 144 theaters for recreation, and 3 theaters for education. We use this assignment throughout all of our experiments. Note that we can use specific assignments if needed for more accurate representations of populated theaters. We do not consider high-traffic transportation hubs such as airports; not all cities have them, and our simulated city is comprised of only 10,000 people. Adding in these locations in future work can be done by simply adding a large theater.

4.16.3 Time

We divide the day into three 8-hour segments. The first segment represents participation in both the job market and education. Bioterrorist attacks would be most effective here due to a high transitory population. Those that have a job go to their workspace and interact with coworkers. The infection rate at a business is only 80% of the baseline, due to the formal interaction and greater possible isolation during the time period. However, other household members may instead attend an educational institution. Here, the levels vary based on age groups, but the general proximity of people to each other is far greater due to enclosed classroom environments. To represent this, the infection rate is 10% greater than the baseline.

Next, in the second segment we have recreational pursuits. Fewer people are involved while interaction is less formal. We assert that such groups would typically be much smaller than those found during the previous segment. The resulting infection rate is estimated at 90% of the baseline.

Finally, we end with a simulation of the household, representing a normal sleep cycle and estimated home visitation pattern. During this period we assume that the family is most susceptible to the spread of a contagion, due to both close physical proximity and deeper levels of interaction. The infected rate used during this period is the strict baseline rate established throughout the simulation.

At any point, if a person is not participating in one of the theaters in the first two segments, they are assumed to be at home. Those remaining in the home unit are calculated the same way as during the first segment, although the effects are reduced based on the present number

of people. This reflects parts of the population that either does not work and/or prefer to remain at home during the evening.

When assigning parts of the population to social theaters we enforce a few rules of assignment. We limit membership to one theater per individual per segment. Multiple members of a household can participate in the same theater, and one individual can participate in multiple theaters on different segments. However, we do not model those that hold multiple jobs. We also do not model those with jobs at night or other parts of the day; jobs and educational participation is limited to the second eight-hour segment.

4.17 Fighting Virtual Epidemics

We explored several factors in our models across a spectrum of infection and recovery rates to determine some of the most effective ways of dealing with outbreaks of infectious diseases. To consider this, we experimented with the shutting down of social network theaters with varying degrees of severity. In theory, if any given theater is no longer in play, we essentially remove its' node and corresponding links from the social network. When this shutdown occurs is dependent on when the attack is either announced by the responsible party or the civil authorities are aware of the situation. Detection is outside of the scope of this research. To simplify this, we experiment with specific dates on which the shutdown occurs.

In all of our experiments, we simulated a hypothetical contagion over the course of 40 days. We set our base infection rate λ to 60% and our recovery rate γ to 25%. Our goal is to represent a relatively aggressive infection that takes a minimum of several days to recover from. The initial infection is of a single person in the same household each time.

We first wanted to establish the relevancy of our work to the existing body of research in the SIR model. To establish this, we compared the original model to our own and attempted to find correlation between them. We found that the characteristics of the infection rate were virtually identical with some adjustments to reflect the average infection rate across different theaters. For example, a traditional model with an infection rate of 72.5% and a recovery rate of 33% acted similarly to our own model under our default setup. However, the matches were not perfect. Although the total number of infections and the nature of the peak infection count were within 1% of each other, our own model did not reach the peak until a few days later. This difference was traced back to how our model requires a lead time for the infection to spread across the social network from the initial point of contact, versus the original model which stated the infection could potentially reach anyone in the city on day one.

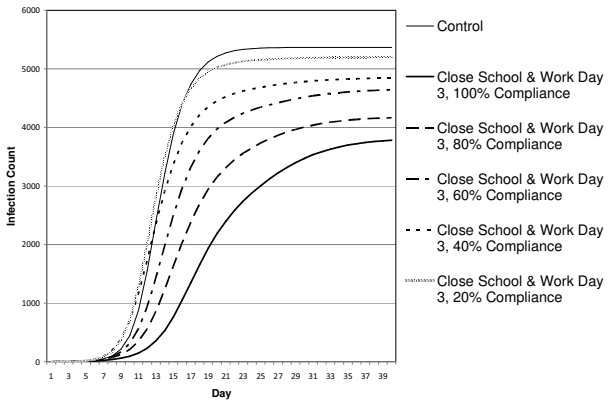

Figure 4.1. Recovery totals over a 40-day period with variance in the

The results showed the impact of the epidemic can be substantially reduced through the closing down of social theaters even with a significant delay. Analysis shows that shutting down any kind of social theater yielded a net loss to the total number of individuals infected over time. Shutting down all public education buildings reduced the tally by a minimum of

8% as late as the 7th day after the attack. When businesses were asked to close, the results showed a 19.5% reduction of infection totals under total cooperation.

In reality, however, there is no guarantee of cooperation when shutting down social gathering areas of any kind beyond public institutions. We next experimented with requests to shut down of varying efficiency, ranging from one out of every five organizations complying with the request to complete cooperation. Our results suggest that the degree of compliance is directly proportional to the effectiveness of the request. For example, a request to close all educational and work theatres on the third day has a 60% compliance rate. In this scenario 4,640 are infected; without any intervention, the normal result would have been 5,367. This translates into a 13.5% improvement. However, at 100% compliance, only 3,769 people would become sick, a more efficient 29.8% reduction. Altercation of the recovery rates in our model unsurprisingly assisted in reducing the total number of infections. However, the effect was relegated below a particular threshold. When the recovery rate was increased by 10% and the infection rate reduced by the same amount, the modification reduced the number of infected individuals to 4,549, a 20% decrease for any time period between initial release and day 9. However, beyond that point, there is a dramatic increase in infection counts up to day 15, after which the total is identical to no intervention at all. This essentially translates into a two-week window for any altercations to existing rates. Looking at these rates individually in our experiments, it is clear that prohibiting the spread of infection will benefit more than enhancing recovery; however, in practical applications, doing both would be ideal.

For an overall analysis of the techniques we have considered, we ran a comprehensive battery of experiments across several days (see Fig. 4.2). If the epidemic can be detected

early, the single most effective technique is closing work with 80% compliance in terms of the peak severity of the epidemic as well as a 15.1% reduction in the total number of infections. However, adjustment of the infection and recovery rates offered the greatest reduction in infections, reducing the total by 16.4%.

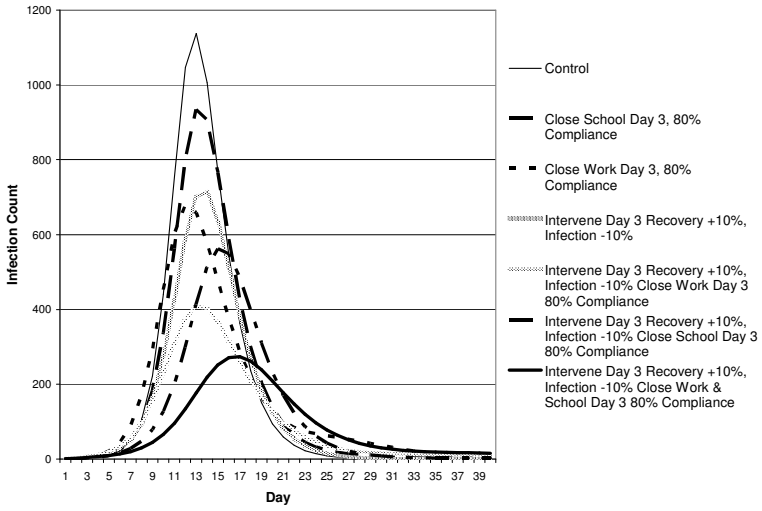

Figure 4.2. The infection peaks for various civil intervention methods.

Across a range of days on which these combinations of interventions occur, we find that the benefits are fairly consistent. However, as the delay between the introduction of the contagion and its' detection increases, the differences among these options begins to shrink. Intervention on the third day can be up to a 48.6% (see Fig. 4.3) decrease in the number of infections when using all of the methods available. However, beyond day 13, we found that the overall benefits were only a marginal variance amongst all combinations of options.

We analyzed a worse-case scenario epidemic with the same infection rate but a recovery rate of only 5%, representing a contagion that was much more difficult to treat. Under these conditions, even a combination of 80% compliance in the closing of both work and school theaters, a 10% boost to recovery rates, and a 10% reduction of infection rates, doing this as

early as the 3rd day only reduced the total number of infections by 7.5%. When dealing with these scenarios, even aggressive policies on civil intervention would make very little difference in the outcome.

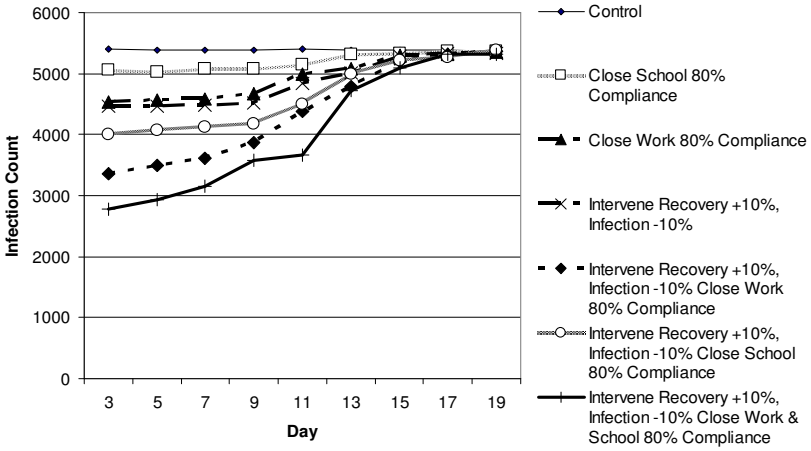

Figure 4.3. Intervention method vs. the day in which it was implemented

In terms of performance, the time taken to run each experiment was tied closely to the number of social theatres involved. The code was written in Java and carried out on a 2.0 GHz dual-core machine with 2GB of RAM. Only one core is used by the application. For a population of 10,000 interacting across 220 social theatres, each day took approximately half a minute. Running non-stop for 24 hours, we can simulate roughly 2,800 days. Since we evaluated many of our experiments for 40 virtual days, this translates into approximately 72 simulations a real day. Population size by design has no impact on the performance.

4.18 Limitations

Although these quarantine methods could be considered controversial under certain political ideologies, our results demonstrate that they are potentially highly effective. In a democratic country such as the United States, the protection of civil liberties has often been at odds with

the need for greater security. However, regardless of public opinion, these options should not be eliminated. In fact, in instances where inoculation is not a viable option, quarantine efforts may be the only way to ensure that the epidemic does not spread any farther.

However, as these studies also suggest, all of the methods will only be effective if used within a reasonable amount of time from the original infection. Once the infection has spread beyond a particular threshold, the effort taken to act may be wasted. We conclude then that there must be particular emphasis in the field of bioterrorism research to analyze and improve detection methods. Likewise, there must also be a rudimentary communication structure in place that brings any possibility of an epidemic to the attention of the appropriate civil authorities as quickly as possible.

CHAPTER 5

THE INFLUENCE OF PERSONAL RELATIONSHIPS AND ATTACK VECTORS ON DISEASE TRANSMISSION WITHIN SOCIAL THEATRES

5.19 The Role of Associations in Biological Destruction

Social networks play a particularly important role in the study of epidemiology as direct or indirect contact between individuals represents the most crucial vector by which a contagion can spread. When performing a case study of the spread of diseases through personal contact, a vast majority of researchers with access to the patients will often perform surveys on personal connections, particularly in the studies of sexually transmitted diseases. [46] [47] [48] [13] Although it may be difficult or otherwise impractical to do the same in light of an airborne contagion, using approximate representations of these networks can yield more accurate results. This, in turn, allows us to gather a better understanding of a potential biological attack and generate more accurate results.

The nature of propagation within graphs yields the trivial observation that starting an infection at one node may yield different results than beginning at another. Biological attacks present the responsible party the opportunity of determining who will be initially afflicted. The possible targets simply referred to as attack vectors from this point forward, may introduce serious implications for a vulnerable population. For example, an attack on a few individuals in a crowded public gathering could yield a more effective attack than targeting a set of likeminded individuals who may share the same acquaintances within a social network. The latter would likely yield an infection that would take a longer time to

spread than the former due to transmission vector overlap, while the former holds promise of reaching more of the network quickly due to diversity of associations.

5.19.1 Refining the Growth of an Epidemic

We built our prior epidemiological model with home units forming a cohesive, single unit. While this is a highly effective use of computational resources, it reduces the details of transmission vectors in such a way that if an individual arrives from a social theatre at home with an infected state of 100% and the three other members leave for recreational theatres in the next time period, they automatically 'bring' 25% of the infection with them. This of course is due to a lack of distinction in what portion of the household is what state.

The added detail of additional nodes also provides a way to simplify the mathematics of the problem. Instead of treating home units as a special case, homes can simply be viewed as the default social theatre when individuals do not go anywhere else. This makes our equations more elegant in their functionality through the use of similar mechanics. We can also remove the original calculations that offset the size of a home unit to ensure only individual contributions are made to the infection model.

(5.1)

$$S_i(t) = S_i(t-1) - S_i(t-1)b_i(t)$$

$$I_i(t) = I_i(t-1) + S_i(t-1)b_i(t) - I_i(t-1)\gamma(t)$$

$$R_i(t) = R_i(t-1) + I_i(t-1)\gamma(t)$$

$$1 = S_i(t) + I_i(t) + R_i(t)$$

The core of the math in this simulation is still virtually identical to that of the SIR model. Time passes discretely via variable *t*. For a given individual *i*, we want to track the percentage that they are in one of the three possible states: susceptible, infected, and recovered. The chance of becoming sick is tracked by b_i based on time. This delta is subtracted from the susceptible percentage and added to the infected percentage. The chance of recovery determines whether the individual will remain ill or no longer be contagious.

(5.2)

$$a_i(t,j) = N_i(j)\lambda(t,j)\left(\frac{S_j(t-1)}{T_j}\right)\left(\frac{I_j(t-1)}{T_j}\right)\left(\frac{1}{T_j}\right)$$

$$b_i(t) = \sum_{j \in P_i} W_i(t,j)a_i(t,j) + \left(1 - \sum_{j \in P_i} W_i(t,j)\right)a_i(t,h_i)$$

We track the social theatres by using the b_i function, which represents how the individual has been affected by their social interactions. This is essentially the effect of all theatre interactions, including the home. The group P_i represents a list of all theatres excluding the home that the individual belongs to. The home itself is tracked as h_i. W_i represents the participation of the individual in each theatre given a particular time. This can be thought of

as a matrix for each individual in which all rows and columns add up to 1; only one theatre can be visited at any given point in time.

$$
\begin{aligned}
W_i(t,j) &= \{0,1\} \\
T_j &= S_j(t) + I_j(t) + R_j(t) \\
S_j(t) &= \sum_{i \in P_j} S_i(t) \\
I_j(t) &= \sum_{i \in P_j} I_i(t) \\
R_j(t) &= \sum_{i \in P_j} R_i(t)
\end{aligned}
\tag{5.3}
$$

At social theatres, the population tallies are formed from a composite of each participating individual's state. As an additional improvement to the simulation in the previous chapter, we model the social network of an individual at each theatre. Two people within the theatre can have a link between them of four discrete strengths. First, there are close friends. These are the people whom share an association based on years of interaction, and are the most likely vector by which a contagion is passed. Next, there are the regular friends based on people who know each other through either their work or mutual interest. These people have less interaction than close friends but still associate with each other at least once a week. After this classification are the associates. Typically, two associates will only have contact roughly once a month. Finally, we classify the rest of the links as strangers. These are people who do not know each other personally but still have a small chance of interacting at the theatre. [49]

At a theatre, the probability that an individual will be infected by another individual is based on three factors. First, we calculate the probability that the susceptible individual will

encounter an infected individual. This is based on the number of individuals in either state at the theatre itself. Second, we account for the probability that this particular individual will be infected out of all theatre participants. Finally, we consider the infection rate in relation to this individual's social network and the theatre's infection rate. The social network impact is determined by the average of all associations. The probability of an individual will recover is simply the chance that they will either survive the affliction or expire.

We keep all of the same options that were available in our prior simulation for dealing with an outbreak. Inoculations are available, but only half of the population can be vaccinated due to limited supplies in our motivating scenario to consider more difficult circumstances. Intervention in the form of providing additional funding is available to alter the infection and recovery rates, but in such a way that the maximum change must be within 10% of the original rates. Closings remain an option, but we do not consider partial compliance as we have already explored it in the previous chapter.

5.20 Improving the Virtual Incubation Process

For our tests, we created a city of 400,000 citizens. Within this population, we assigned membership consistently across all experiments in three kinds of social theatres: businesses, educational institutions, and recreational gatherings. These theatres are limited in this set of experiments to an average of 100 members; although several larger businesses may exist, we surmised that the typical commercial office space would be chaotic if setup in such a way that permitted a larger group to coexist in the same area. We assume they have been grouped either by floor or building. While migration between parts of facilities is certainly possible, we disregard the effects and allow the network to naturally reflect weaker linking that results from it.

Educational institutions involve closer contact with larger groups of individuals in a smaller amount of space, resulting in a higher relative transmission rate. Such institutions include a full range of possibilities, ranging from public elementary schools to universities. While they present a more dangerous opportunity for epidemics to spread, in many societies the majority of these institutions are publicly funded and controlled by the government. Thus, in the event of an emergency, we assume the defending country will have enough control to shut all of them down indefinitely in the interest of student safety with little repercussion.

During the later parts of the day, some individuals may still choose to participate in a smaller group activity. Contact here is among much smaller groups, but the level of interaction tends to be much higher, resulting in a more aggressive infection rate balanced against the presumably relaxing nature of the activity. These groups are also harder to control; social functions can be difficult to regulate due to societal views, personal freedoms, and no guarantee of a single location at which the activities take place.

When any individual is not attending any of the theatres during the day and evening periods, they are assumed to be at their place of residence. This has been transformed into a social theatre which any individual not attending another must return to. However, relationships at the home are assumed to be homogenous for the purposes of simplifying our simulation overhead.

By default, our theatre setup for relationships uses a bias representative of work within social networking on the levels and frequency of friendship. We assume that close friends are rare, composing 5% of a person's social circle and are likely to meet up with the individual at least once a day. A regular friend is someone who one is often likely to have twice as many of at 10% of their immediate network but will only be seen on average once a week. Associates

are those we interact with due to professions, casual acquaintances, and other people we are likely to see at most once a month. They compose an average of 20% of an immediate network. Finally, there are strangers, people whom are often seen no more than once a year, compromising the rest of the people known for most individuals. These figures are based on information derived through social study and represent an approximation of the results. [50]

We now consider a number of possible places in which an infection can begin, known as attack vectors. These possibilities were isolated to 3 targets in 3 degrees of intensity. We named them based on the target *t* and size *s*, using the simple designation of *t-s* in our experiments to indicate the attack vector. This is done in part to simplify the analysis of all possible scenarios; however, in a practical light, this makes sense. Consider the motivations of many organizations willing to engage in biological warfare on a civilian population. Such an attack would take a considerable investment in time, resources, and money. We assume that they in turn would want to make a considerable impact on the society or rival organization in question, and as such would prefer to introduce the contagion to a select list of highly desirable targets. In these situations, we assume that the protecting agency has isolated the most likely people that the attacks will be directed at due to existing research.

5.20.1 Improving Experiment Throughput

In the previous chapter, experiments were run on a single processor with little need to expand resource consumption. As we increased the complexity of our model and the scope of our experiments, we realized that we needed to maximize the resources available. To this end, we have created a new generic platform for running stand-alone experiments, which we have dubbed Project Wildfire.

Each experiment is setup according to an initialization string that carries information such as infection rates and where to start the contagion. These setup strings are stored at a central server publicly available on the internet. The actual execution of the experiments are carried out by agents we call contractors that are given the location and port to reach the server at. Each contractor begins by registering with the server using a specified nickname for a unique identification number. Then, the server is queried for the next available job. This job is retrieved, setup, and executed on the contractor's local system. Updates are sent back to the server periodically, yielding information such as the current progress and status of the system. After a job is complete, an informational packet containing the resulting data is sent back by the client, which the server stores and records according to the original order of the list of experiments. The contractor then asks for another job and repeats the process until no more experiments are available. This setup allowed us to leverage computers anywhere on the internet, enabling us to pool remote resources and increase our overall throughput. Once all experiments have been completed, a results file is created as if the experiments were run on a single system.

Although we enjoyed a remarkable increase in efficiency at first, problems arose in long-term experiments that required predictable uptime of all participants. Several levels of redundancy were created to harden the system against data loss. First, all jobs must be updated by their respective owners within a time period of a few minutes. Should the contractor freeze, crash, or simply fail to report, their assigned job is released and allocated to another. Second, a parallel thread runs with the job manager and periodically writes a checkpoint of complete and incomplete jobs to the local storage. If the manager itself shuts down or must be turned off, the checkpoint can be processed and given to the next instance of the server, allowing it

to pick up where the other had left off. Finally, simple mathematical gauges were added that allowed the user to know how many jobs had actually been completed, how many were left, the speed at which the system was processing a single job, and an estimate of the time necessary to complete the work. Our goal is to make the system and code available to the academic community by December of 2008, in the hopes that others could take advantage of this simple approach.

5.21 Simulating Epidemics across Multiple Dimensions of Choice

In light of the choices available to both the attacker and defender, we simulated several scenarios to explore the space of strategies available. The multitude of options to select from allowed us to create several matrices of choices to run experiments on. The goal in our work here is to understand how our refined experimental model predicts the outcome of an attack with the specified origin and defensive choices made.

5.21.1 Random Inoculation with Isolation Methods

Comparing the closings of several theatres with respect to varying levels of random inoculation, we find that the two methods continue to scale well together in our revised model. The number of inoculations performed in advance linearly scaled back the total number of infections. As more of the population was inoculated, the effect of the closings carried a similar effect. However, the impact of the closings themselves diminished as a later date was chosen. After day 3, the impact on the total number of infections became negligible. The peak infection rates suffered from the same issue.

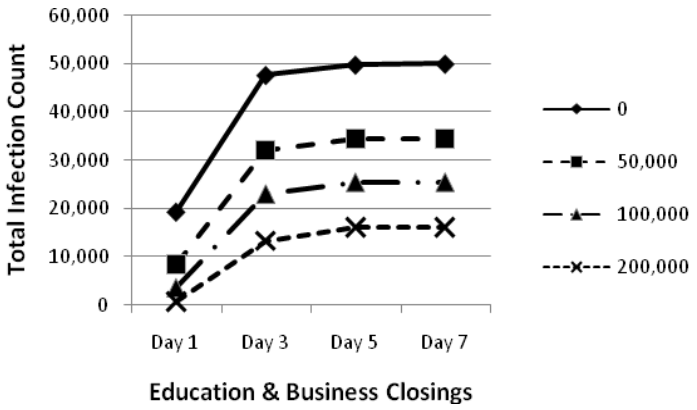

Figure 5.1. Inoculation sizes vs. the closing down of business and education

The point at which the peak number of infections occurred was pushed back subtly over time. As higher levels of inoculations were considered, the nature of how the infection progressed changed considerably. When half of the population was inoculated, the infection essentially arrived in three 'waves' that became more distinct at higher levels of inoculations. For example, when 25% of the population was inoculated and both work and school theatres were closed on day 3, the infection stalled or peaked on days 41, 59, and 67. However, when the inoculation was increased to 50%, the peaks shifted to later dates on days 46, 60, and 69. The importance of peaks and when they occur is paramount in dealing with an epidemic. The peak of an infection determines the maximum amount of resources necessary to attend to the needs of the ill. For example, consider a hospital dealing with infected patients. If the number of patients at any given time does not exceed 100, then the 120 beds reserved for those afflicted is never exceeded. Even if the infected population holds steady for weeks at a time, patients can still be given the level of care necessary. However, if a shorter period of infection ramps up to a peak of 200 patients, then the hospital must either invest heavier resources in meeting the capacity or turn patients away to other facilities which hopefully are

not suffering from the same problem. The complexity of our social network overlaid with the SIR model has resulted in delays as the contagion propagates into members of a household, develops, and eventually is passed on to other individuals via theatres. While we saw this to an extent in chapter 4, this has become much more pronounced in our revised simulation.

We assume that individuals remaining at home will have a lower probability of sickness than those at most theatres. However, we have considered other experiments where the opposite situation was assumed. One of the more disturbing finds in this line of experimentation is the effectiveness of a contagion incubating within members remaining within a home versus the attendance of social theatres. We assume that social networks within the home are irrelevant for most contagions in that the proximity and standard level of contact is the same. If the infection rate at the home is higher as a result of this factor, the potential spread of the contagion becomes extremely important. Once the infection has spread across all transmission vectors, closing social theatres will not benefit the population in this situation. Additionally, if the infection rate becomes higher due to social networking within the home theatre than other theatres, closings will actually do more harm than good. Homes would become incubation chambers for the contagion, intensified as fewer locations are available.

5.21.2 Inoculation vs. Size of Attack

The options available to the attacker can vary considerably. They essentially have the option to attack any members of the population as they see fit, resulting in essentially up to 6.4 x 10^{16} choices if only three people are infected. To avoid the temptation to explore every possible scenario, we assume that the attacker will only consider a handful of potential targets. Experimenting with the possible targets, we discovered that the attack vector impact

does not always scale directly. Consider vectors 3-3, 3-6, and 3-9. Each attack progressively includes the smaller targets plus three additional people. A traditional consideration of the scenario suggests that an increase in the number of victims would directly result in a much larger infection.

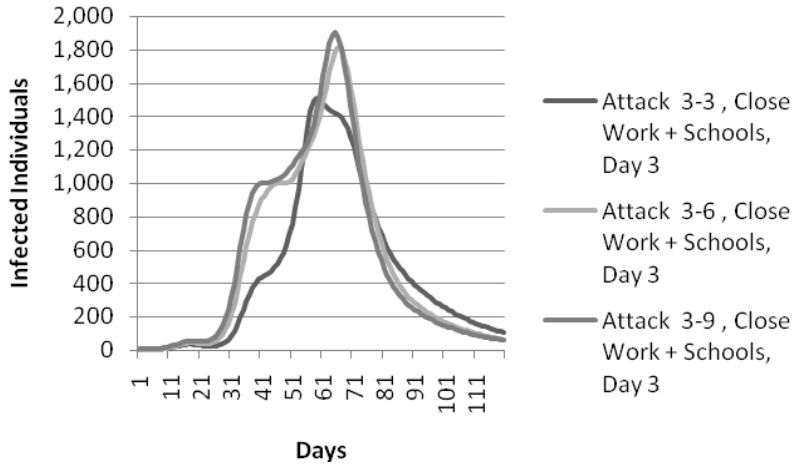

Figure 5.2. The infection graph of 3 different attacks of varying size on location 3.

While attack vector 3-6 was a significant step forward from 3-3, the distinction between 3-6 and 3-9 was much smaller. According to the original SIR model, the number of infected individuals essentially results in a proportional probability that any susceptible node in the network will be infected. On Day 15, the first peak in each simulation is indeed higher based on how large the initial attack was. However, as three more weeks pass, the influence that the initial targets had over their peers begins to overlap significantly. Attacks 3-6 and 3-9 reach their first peak at nearly 1,000 infected people, while 3-3 does not touch that value until 3 weeks later. Since the latter attack has a smaller original impact, the recovery rate insures that the total number of infected individuals never rises above 1,531. The day on which the peaks occurred also demonstrated the complexity of the underlying network. Even as the closing dates are varied instead, the characteristics between the vectors remain similar.

5.21.3 Infection Rate Variance

Another gap in our understanding of biological warfare is the estimation of the baseline infection rate. This rate essentially yields a simple probability representing the complexities of the human immune system and the impact of external influences. Typically, this varies based on a number of factors such as health, age, and personal hygiene. We did not account for potentially variant infection rates in chapter 4, as we used a constant instead. We varied the infection rate over 70%, 80%, and 90% in order to analyze whether or not the methods considered scale well.

The infection rate of 70% serves as our standard baseline for our work. Looking at an inoculation percentage of 12.5% and attack vector 1-9, we saw a remarkable difference in the results as it was varied. At 80%, the number of afflicted individuals rose by 5,005, while 90% saw over 4 times that amount at a delta of 22,754. In both cases, there was an additional 'aftershock' of infections. While a 70% infection rate saw a subsidence around the 3 month mark, the 80% rate saw an additional swell on day 65 that peaked on day 100. This resulted in a significantly higher secondary 'swell' of sick individuals.

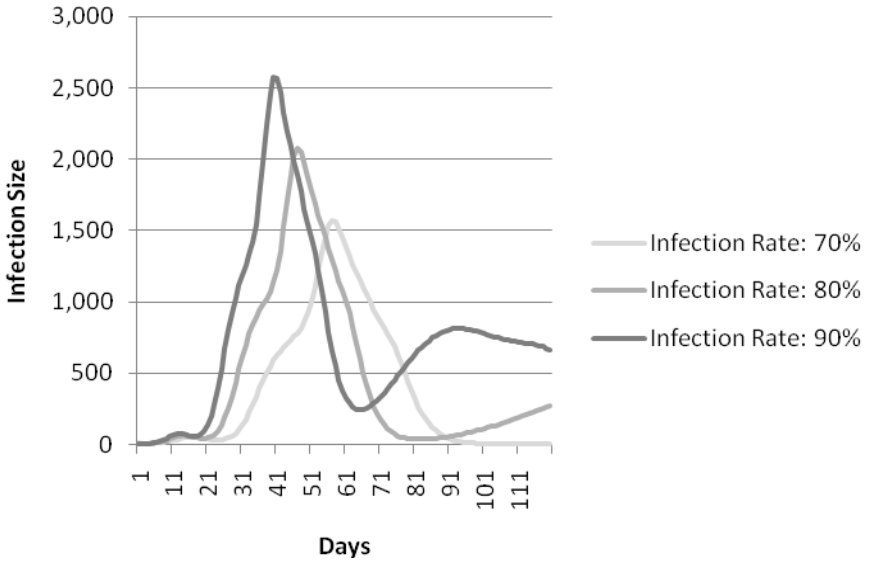

Figure 5.3. Attack Vector 1-9 on a variety of infection rates.

Analysis suggests that a higher infection rate allows a contagion to have a more pronounced movement through a system, creating the 'aftershock' when prior individuals that only had moderate chance of infection before the recovery rate eliminated the growth instead saw a reemergence of the threat as other parts of the network began to reach individual infection rate peaks. In other words, the disease is simply not eliminated from the population fast enough, allowing it to leverage more distant parts of a social network before returning to infect the remaining susceptible 'core' majority.

5.21.4 Social Distribution Impact

The impact of the complexity of a social network becomes particularly apparent in the midst of random inoculations. Looking at the 1-9 attack vector over the course of inoculations, an initial glance suggests only subtle variations over time. Figure 5.4 demonstrates that the inoculation method maintains an increasing effectiveness when all facilities are closed on day 3.

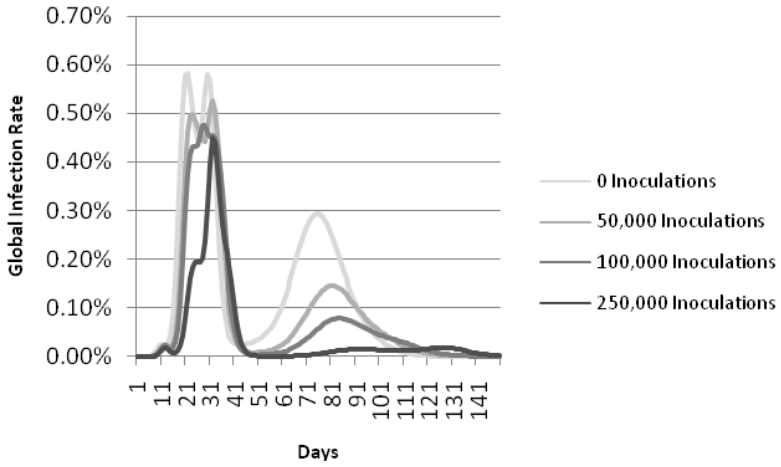

Figure 5.4. The influence of inoculation sizes on the results of attack vector 1-9.

Conversely, if the relative delta of each change is looked at instead, we see a very different set of patterns emerge. When no inoculation occurs, there are 4 distinct epochs as which the spread of the contagion peaks. This occurs at days 12, 22, 30, and 72, the maximum of all on day 30 at a delta of 5.81%. The first epoch is roughly the same for all inoculations, with predictable variance in the time and intensity as a function of the inoculation size. However, the intensities of the 2nd and 3rd epochs share almost identical values.

When a quarter of the population is inoculated, a different trend emerges. The infection in this situation appears to go through at least 5 discernable peaks, the maximum of which is 4.75% on day 28. This is a 27.8% reduction in the delta, but it occurs on the same day.

When we compare this with 2-9 attack vector, show in figure 5.5, a similar phenomenon occurs but with varied characteristics. The two initial peaks which emerged under no inoculations remain the same distance apart but now vary in terms of the maximum in a difference of 0.034%. However, the overall intensity of the same infection rate appears to have been lessened, with a final peak at 0.249% rather than 0.288%, suggesting that the 2-9

attack does not reach as many nodes as quickly as the 3-9. The results for higher levels of inoculation appear to follow suit, though the time period has shifted for most peaks.

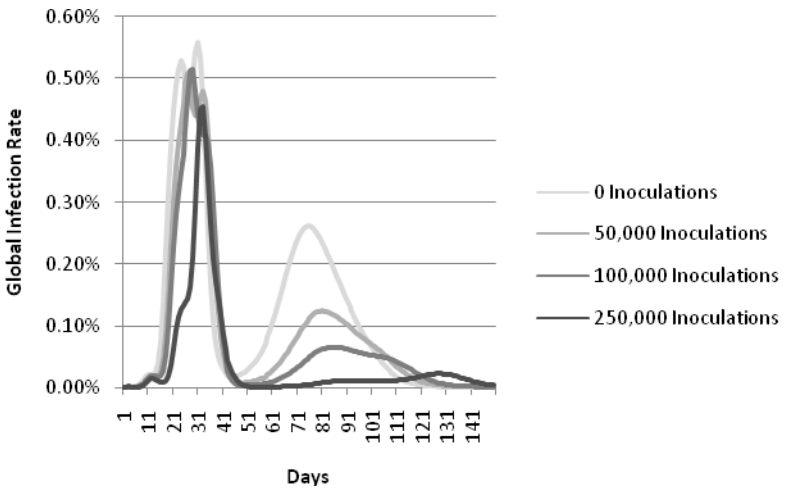

Figure 5.5. The influence of inoculation sizes on the results of attack vector 2-9.

5.22 Simplicity within the Outcome

As the complexity of the associations of our social network increased and the degree of separation between any two nodes decreased, we found occasions in which our model began to resemble the simpler SIR model more closely. We believe that locations in which either the social connections held by a single individual are high enough or the amount of contact between individuals is high, the need for complex models begins to decrease. Consider for example a dense population area such as New York City. The vast majority of the residents are unable to avoid the need to walk at least some distance through the city to reach a desired destination. During this walk, the number of people that are contacted directly or indirectly can range between ten to hundreds. Given this fact, the infection rate between individuals begins to become much more uniform. Coupled with a lifestyle that favors crowds, the SIR model may actually be a better choice for some scenarios.

The impact of closing all facilities, while moderately effective, did not fare as well as closing all but recreational theatres. This appeared to be due to the choice of a lower infection rate for recreational theatres, which in turn is due in part to the nature of what can contribute to the susceptibility of an individual based on their mental state of health. Studies have shown that stress can increase the susceptibility of a person to viral infections by as much as 70% to 95%. [51] Other reports indicate a positive mood can actually boost the body's natural defenses against maladies such as the common cold. [52] [20] [53] Essentially, positive social activities have been proven to reinforce the immune system, which actually suggests that our recreational areas are accurately depicted in our models. However, some social activities such as those that involve alcohol consumption may actually have a negative impact. [49] Additionally, the societal norms that dictate what is acceptable for a government's involvement may indicate that a government mandated closing of social theatres is a violation of human rights, resulting in unhappy citizens that may pose other threats to society as a whole. In short, the closing of recreational areas is most likely going to have a negative rather than positive impact on the safety of a population.

In the previous chapter, we discussed that shutting down businesses, even in the interest of protection, could have a largely detrimental effect on the local economy. Instead, a more effective approach would instead be to concentrate the remaining methods on different social theatre participants. In several countries, schools are publicly funded and controlled in part or wholly by the state; they could easily be closed with little immediate effort. Meanwhile, inoculations could be carried out at all places of work. However, as effective as this combination may be, mixing the methods raises an interesting question in terms of public safety and moral perceptions. There is a distinct possibility that, despite the best intentions

of the state, the public may interpret the inoculations of only those who attend business theatres as a form of elitism, as it would exclude those unemployed or whom work for themselves. Any benefit from enacting such a policy could be substantially offset by a population that may already be afraid due to the nature of the threat; the consequences could range from mild civil dissatisfaction to riots and non-compliance with future orders. This depends primarily on the nature of the state, the general status quo, and the shared beliefs of the population.

After our modifications to the model, we still consider the performance of our experiments an important factor in our success. The speed of the simulation varied based on how many social theatres were currently still available. Our test system was an AMD Phenom X4 running each core at 2.4GHz with 2GB of RAM. Each simulation ran on a single core and occupied approximately 420MB of RAM. On average, a complete day took approximately 1.1 seconds to run on a population of 400,000. Within a single real day, we can simulate over 7 years on just four cores. Given that the average scenario takes 120 days, we can run through over 650 scenarios in the same time period. This provides room for a governing body to consider a myriad of possibilities or consider scenarios in which all of the data is not known.

The experiments suggest that the spread of a contagion through a varied social network can offer misleading results when analyzed with regard to the trends of information. A traditional SIR model has a predictable peak, emerging gradually. Introducing noise into this trend does not significantly affect the characteristics of the observed trend if an average is taken. However, our model demonstrates that even a low probability of the spread of infection can lead to significantly different and potentially misleading information. Consider

the first epoch in the comparison between attack vectors with respect to recovery. It at first appears that, within the first two weeks, the infection has subsided and resources can be withdrawn. However, as the following week unfolds, the number of infections rose dramatically as latent infections arose in the individuals they had contact with. The results of our model suggest that the small world phenomenon may be able to predict the number of 'peaks' that are observed in the data. For example, assume that there is an average of 4 hops between any two members of a social network. Assuming 100% probability of passing an infection, a biological attack would likely spread in 4 days or less to virtually every member. This is a trivial observation. However, if we reduce the probability dramatically and introduce multiple possible vectors over which the infection can spread, the overall system will likely see swells in infections spread out over a period of days.

Multiple peaks within infection rates are not new to epidemiology [47], but it has so far been apparently overlooked in the realm of biological attacks. Traditionally, multiple peaks occur due to the nature of smaller populations and the latency between infection and discovery. However, we have observed that even an epidemic that unfolds rapidly may encounter sufficient resistance within the traversal of large-scale social networks, resulting in infections swelling at different points in time. This poses a grave threat to a country that wishes to withdraw intervention methods in order to resume normal operations, as they may actually encounter an additional serious climb in infection rates as much as a month after the epidemic has subsided. This dictates that proper understanding of an epidemic requires at least a rudimentary sketch of a social network to allow for delays in the propagation of infection across multiple vectors. The only exception to this would be an epidemic that spreads via an airborne delivery system rapidly without the need for close proximity.

In short, one of the most common features of our results is that the nature of social networking, the ways in which individuals associate, and the factors involved can all lead to a variety of scenarios with a myriad of challenges. This further reinforces the need for accurate models that can accommodate these factors properly with the most accurate data available. Otherwise, simulation results from simpler models may yield inaccurate information that could lead to deadly results in light of an actual attack.

CHAPTER 6

APPLYING GAME THEORY TO EPIDEMIOLOGY MODELS

6.23 The Motivation of Tragedy

In the wake of the frequently tragic aftermath of an act terrorism, one of the first questions much of the afflicted population will ask is, "Why". Namely, it is puzzling for many as to what would motivate an individual or group to resort to acts of terrorism carried out on a civilian. [54] This is clearly a difficult question to answer, one that is rife with considerations of ideologies, the value of human life, and what constitutes acceptable behavior in light of what are often radical beliefs. Such an answer is vital to understand not only the cause of an existing attack, but what would motivate attempts in the future in the hopes of preventing other attacks.

Securing a population against a bioterrorism attack is clearly an important goal. However, realistically, an interest in protecting the population and what is actually feasible with available resources often do not coincide. We assume that our hypothetical country wishes to minimize, if not eliminate, the potential casualty rate due to an attack. As precious as human life is in the eyes of many, the challenge of limited resources must be weighed considerably in light of future obstacles that may or may not be terrorism related. The goal of our work is to answer a considerably difficult question which is naturally derived from these constraints: Is it possible to maximize the use of existing resources to minimize the impact of a biological attack in the worst-case scenario?

Assuming that the goal of a terrorist organization is to maximize casualties, and the goal of the local government of the targets is to minimize them, it becomes possible to realize the scenario as a perilous challenge between two opponents. Game theory is an approach to strategy and mathematics that is rooted on the assumption that two or more rational players wish to maximize their overall utility in light of their opponents' choices. [55] Such an approached served well in the information sharing phase. Thus, we continue the use of this model by realizing this scenario as a game played between the attacker and the defender.

Consider the nature of public governments. In many countries, one of the responsibilities of a governing body is to be accountable for its' actions to the people it oversees. In this situation, any move made to protect the population will be highly visible to the public. For example, in December of 2001 the state of Texas held a forum on biological preparedness that outlines local government responses to a potential attack. The information is available for public viewing due to the transparent nature of the state government. [56] Mass inoculations would be difficult to hide either as a mandatory or optional course of action for individuals. Any training for preparedness would be hard to keep secret if local governmental agencies such as regional police stations and fire departments are involved. In short, the actions taken by a governing body in defense of the population are difficult to keep a secret; thus, we make the assumption that any defensive measures are going to be known well in advance by malicious organizations.

The actions taken by the same organizations are often much more difficult to observe. Since several terrorist groups operate in loosely coupled cells [57], ascertaining their moves requires serious investments in research and intelligence. How far in advance an action can be anticipated cannot be relied upon. In a conference held by the OECD on the 22nd of

November in 2004, analysis of the nature of risk assessment and insurance with regards to terrorism revealed that insurers were concerned due to the lack of recent history of terrorism and the unpredictable nature of how, when, and where an attack would be carried out. [58] Indeed, the concern is so high that the Terrorism Risk and Insurance Analysis act of 2002, specifically designed to protect insurers in light of catastrophic terrorism events, was originally set to expire in December of 2005. Due to continued concerns regarding trends in terrorism and the lack of data available to insurance companies, the act was extended twice, and now remains valid through December of 2014. [59] We thus must assume that, as a player, the attacker's actions will not be known until after they have been committed.

A Stackelberg game is one in which two players, a leader and a follower, attempt to maximize their own utility in light of disparate knowledge. The leader goes first, choosing a strategy first from available options. The second player, the follower, then picks their own strategy with perfect knowledge of what the leader has chosen. The leader, however, does not know any more than the available strategies to the follower, and as such, must attempt to make a decision that maximizes their own utility in light of what the follower wants.

$$\max_{(a_1,a_2)\in A_1\times A_2} u_1(a_1, a_2) \text{ subject to } a_2 \in \arg\max_{\acute{a}_2\in A_2} u_2\left(a_1, \acute{a}_2\right) \tag{6.1}$$

$$A_1 = C(\text{inoculation}) \times C(\text{intervention}) \times C(\text{closing})$$

$$A_2 = C(\text{target}) \times C(\text{size})$$

Thus, the leader must choose the most profitable strategy a_1 while understanding the follower will choose their own a_2. The strategies available to either player are determined by our experiments via the choice function C which, given a particular parameter, generates the strategies available. The defender must choose from a set of strategies A_1 which contain the number of random inoculations, what type of intervention will be administered, and what type of closings will be performed when. The attacker's strategy set A_2 is composed of what to attack and how large of an attack to carry out.

The utility function for the defender u_1 determines how the outcome of the simulated epidemic will be perceived. The simplest approach to this would be counting the number of casualties that result. However, the function can also be based on real world data, providing an estimated currency-driven value on which to consider both the investment needed for the defensive choices made as well as the casualty count. We describe this in more detail later in the chapter. An attacker's utility u_2 was difficult to determine, as our research did not uncover reliable statistics on the costs to a terrorist organization. We will instead use the number of people infected during the attack.

(6.2)

$$u_k = \sum^{i \in G} R_i(t) + I_i(t) - R_i(0)$$

(6.3)

$$\dot{u}_k = -\left(\sum_{t=0}^{n} \sum^{i \in G} I_i(t) C_h + R_i(0) C_i + c_f(a_1) + c_v(a_1) + c_i(a_1) \right)$$

Gauging the results of choices in a game remains one of the most crucial tenants of a realistic scenario. We use the utility functions u_1 and u_2 to determine what that outcome may entail. If the attacker wishes measure results based on the total number of infected individuals, then the function is simply defined according to (6.2), where G represents all individuals in the game, k represents the player using it, and t represents the time at which the game has ended. R_i at $t=0$ represents people who were originally inoculated. Since this area is subject to speculation and often strays to philosophy during debates on the topic, we will focus our concerns on what can be measured with available data. Note that the defender would use the negated function $u_1 = -u_k$, as they wish to minimize the number of casualties.

When the defender wishes to consider numerical cost, we use the $\dot{u}_k$ function instead. First we assign a daily cost C_h that determines the perceived impact of each sick individual. This is multiplied by the number of infected people per day and added up to the end of the simulation on day n. Next, we multiply the cost of each vaccine C_i by the number of people vaccinated. This is followed by the cost of the facilities function c_f which takes the choice made by player 1 to add the initial inoculation overhead based on whether or not any

inoculations were made. Following this is the intervention method cost function c_v that yields the impact of the choice made as well on where to put funds for hospitals, public service announcements, etc. Finally, we have the cost of closings determined by the function c_l, using the same data to determine the economic impact of the social theatres. Note that this function must calculate how many business employees are affected by the closings, if any. Should the attacker wish to determine their benefit based on the cost incurred to the defender, they would use $u_2 = -u_k$ for their own utilization function. The detail of these costs and cost functions are described later in section 6.2.6.

Gauging the results of choices in a game remains one of the most crucial tenants of a realistic scenario. Considerations in this game include a number of options and consequences, ranging from whom to inoculate against a suspected biological agent and determining the loss of human life in light of who is saved. Since this area is subject to speculation and often strays to philosophy during debates on the topic, we will focus our concerns primarily on the financial impact of an attack amidst varying considerations of cost to both players.

Ideally, we wish to find a defense strategy that provides a reasonable and theoretically guaranteed upper bound to the impact of an actual attack. This suggests that we should seek a dominant strategy within our game, which in this instance is simply a strategy choice made by the leader that minimizes the maximum damage done by the follower's moves. [60] Since there are so many factors involved, it is possible that we may find several equilibrium spanning different scenarios, or perhaps no equilibrium exists at all. Regardless, there is a clear benefit to these conclusions in light of a tangible biological threat.

6.24 The Culmination of Ideas

6.24.1 Creating the Defending Population

We varied slightly from our previous experimental hyper-matrix of options with a different setup. Our hypothetical target is a city with a population of 400,000. Within this population, 250,000 individuals are employed. The average family has 4 members. 80,000 members of the community attend some form of educational institution. For all theatres except homes, a detailed social network is overlaid using the parameters provided in our prior work creating the effect of relationships and the impact they have to the probability of transmission.

The virtual day is divided into three 8-hour time periods. The first period is considered the work period; the population is either at work, attending an educational institution, or residing at home. The second period of time is the recreational time period, where people are relaxing at home or at a recreational theatre. Finally, there is the rest period, where all individuals are assumed to be at home. A total of 120 days, representing roughly 4 months, are simulated.

Infection rates at each location are assumed to vary due to the nature of activity. Educational theatres carry the highest infection rate, as they frequently require several individuals to be confined in the same room to attend class, lectures, etc. for an extended duration of time. Work theatres are second, as they have less personal contact and relative proximity but often see an increase in infection rates due to stress, as discussed in the previous chapter. Homes are more confined spaces but have the potential to be a relaxing environment. Finally, recreational areas have the lowest infection rate, as they are often less confining and are assumed to bolster immunity due to the recreational nature of activities.

6.24.2 The Nature of Biological Attacks

Any citizen of this city represents a possible target. However, taking into consideration all possible targets of a hypothetical attack on just three people means there are a considerably high number of possibilities. Instead we will assume the attacker is politically motivated and 3 possible targets. Within these targets, the attacker can commit resources to infect 3, 6, or 9 people, yielding a total number of 9 attack vectors. Obviously, if we knew which targets were in question, we could simply discretely inoculate them. Instead, we will assume for the sake of relevant data that this cannot be done.

The base probability of infection is set at a probability of 80% based on a study of existing data. [64] Recovery will be set to 10%, based on the same data set which suggests most people will have infected others within a lower bound of 10 days after the onset of the first symptomatic period. After this point, we assume they have either passed away or have been quarantined because of visible symptoms.

6.24.3 Intervention Options

We will assume that the city governing entity wishes to consider options of intervention. There are three levels at which they can alter the infection and recovery rates: 5% and 10%. In the case of infection, this would be a reduction, while recovery would be bolstered. This is done through means such as promoting public awareness, civil servant preparation, and allocating funds to medical facilities. It is assume that a governing body will choose at least one of these rates to be altered in order to stem the tide of the infection.

6.24.4 Inoculation Considerations

An obvious course of action is to use a vaccine to inoculate members of the population. We will assume that this is carried out by some branch of the government at a cost both for initial investment and the vaccine per person. Possible values considered are no inoculations, 12.5% of the population, 25% of the population, and 50% of the population. Clearly, 100% would be the most effective, but we will assume that for reasons beyond control only 200,000 doses of the vaccine in question are available and the city cannot control who will receive it. Note that any individual who is inoculated is considered completely immune to the contagion.

6.24.5 Closing Social Theatres

We introduced the concept of closings and their impact on stopping infections in chapter 4. Several options are available to the governing entity. Assuming that the infection was known of either through detection or announcement, the city can close theatres within one to four days after the attack. Since there is no real data available on this method, the cost of a closing will be based instead on the indirect effects on the local economy, such as the loss of income taxes when employees do not earn a paycheck. Facility closing options include just schools, schools and workplaces, all theatres, and none.

6.24.6 Evaluating Cost of Life

Evaluating the results of an epidemic is a considerably difficult task. One must weigh the value of human life in relation to the cost of preventing the loss of it as well as ensuring the quality thereof. Dealing with an epidemic is no small task for the authority responsible for the safety of the victims, and there must also be considerations that span the whole of the

body at risk. An epidemic in Houston for example can threaten the population of the entire United States. Obviously, this question is subject to a number of schools of science, philosophy, and political ideologies. However, for the sake of evaluating the results of our simulation, we will assign costs to the actions to the best of our ability within the realm of practical science and existing techniques. Since the factors that determine these costs can vary widely across the globe, we will confine our hypothetical simulation to biological attacks within the borders of the United States and consider loss in terms of US Dollars.

Additionally, since the results of an epidemic are dependent upon the biological agent used, we must assume the use of a single infectious disease. Despite the influence that concerns about Anthrax have had on the increased research in bioterrorism, understandings of the bacteria have shown that it is difficult to spread quickly beyond the initial attack. This is due to the fact that it requires the direct inhalation of the spores, the distinct symptoms manifest quickly, and it is only transmissible through direct intentional contact or zoological vectors. The severe acute respiratory syndrome coronavirus (SARS) has been suggested during an outbreak in China during 2003 as a possible agent that was the intentional result of biological warfare research, but the highest observed infection rate was less than 7% and the mortality rate was relatively low. [65]

We instead chose Smallpox. Efforts in the 1970's worldwide resulted in the near elimination of the virus. In light of recent concerns, the Center for Disease Control considers the virus as a class A threat. There is a relatively long incubation period for the virus, during which time the affected individual is neither contagious nor tends to feel any symptoms. After that time, the person usually becomes contagious and begins to experience the early symptoms. Depending on which type of smallpox is involved, death rates can be as high as 50%. [66]

Additionally, the appropriate strain can be distinct enough that prior inoculations during the 1970's would not be effective. We will however assume that a vaccine is available, for the sake of analysis.

The first and most pressing issue is the cost of fighting infections. Although one could argue that it is invaluable, several governmental agencies within the US have assigned a specific dollar value to a single human life based on a number of factors which primarily revolve around not on earning capacity, but rather willingness by an individual in peril to expend resources to avoid death. We will be using the Environmental Protection Agency's rating of $6.1 million US Dollars adjusted via GDP deflation from 1999 to 2008. [67] This cost will be applied to each projected death as a result of an infection. For those that are rendered unable to work due to infection, we will apply that amount averaged over the 2007 life expectancy of 77.8 years. [68] This comes out to roughly $214.81 as the daily 'willingness to pay' to extend an individual's life. Although it seems logical, we will assume that human life is invaluable enough to avoid consideration the EPA's number as a cost in the event of an actual death.

The next cost to consider is the impact of inoculations. First, the handing out of inoculations will require an initial amount of resources in terms of staff. We assume for the sake of simplicity that a staff of 20 registered nurses has been assigned and that they can each inoculate a single individual in a minimum of 2.4 minutes. This is a preventative measure taking place well in advance of the attack itself, so normal hours can be assumed, meaning that they are working 8 hour shifts and inoculating at most 200 people apiece per day. Assuming the average accounts for travel time, the entire city could be inoculated at this rate in approximately 120 days. As of the writing of this document, the average registered nurse

receives $25.80 US Dollars per hour [69] which puts the total cost of staff alone at $24,768 if we employ them on a simple four-month contract regardless of the number of inoculations during that time. The actual cost of each inoculation on top of this is difficult to calculate, as while the vaccine itself can be relatively cheap, the infrastructure that goes into its' production can be attributed to the final price. In 2002, the United States Department of Health and Human services awarded a $428 million dollar contract to produce 155 million doses of a smallpox vaccine by the end of 2002. [70] Assuming that deal went through and no additional costs were incurred, each dose is worth $2.76. The final cost of inoculations then comes out to the base cost of the nurses plus the expense of each inoculation.

Dealing with closings presents an additional challenge. If schools are publicly owned, closing them will have no immediate perceivable financial impact. We recognize this is a simplification, as the continuing education of the populace is certainly a high priority. However, the concerns of many governing bodies will likely be focused on immediate tangible economic drawbacks. In 2007, the population of the United States was 302 million [71] with an unemployment rate of 4.6%, resulting in a total of 288 million employees nationwide according to the CIA World Factbook. [72] The IRS reported gross tax income of $1,245 billion due to corporation and employment taxes in 2007. [73] In the same year, $319 billion in state taxes were collected via income taxes. [74] Assuming that businesses would suspend operations in light of being shutdown and no longer contributed during that time to any of those situations, this equates to a cost via loss of $14.88 US Dollars per day per person. Although it would also likely have impact, we will assume that recreational theatres have only negligible impacts and will not be considered at cost.

The final cost to consider is altering the course of the epidemic via public service announcements and funding to medical facilities. We will assume that a public service announcement costs nothing in the time of an emergency but it must be coupled with hospital funding to be effective. There are two ratios that are being modified: infection and recovery. We will assume that initial 5% of modification in either category individually will be half the cost necessary to reduce or increase it an additional 5%. In a study of hospital budgets, it was found that the average cost of a patient in normal levels of care is $1,121.59. We will assume that some fraction of this is necessary to increase in order to allow the hospital to bring in additional resources such as doctors and medicine, and as such 5% of this amount will be necessary for the first 5% of each rate. Thus, increasing the effectiveness of treatments per patient would be $56.08 to either increase recovery rates by 5% or decrease infection rates by 5%. To bring either to 10%, an additional $112.16 per patient would be necessary. This dictates how much each infected individual will cost the governing entity per day. For example, if a 10% reduction in the infection rate was desired with a 5% increase in recovery, it would cost $224.32 per person. This will be applied to all infected individuals whether they are within a hospital or otherwise.

6.24.7 Cost of the Attacks to the Attacker

There is no existing work that we have found which suggests the costs a terrorist organization would incur to carry out a biological attack. The lack of data would render any guess financially useless in comparison to the costs incurred by the city. As such, we will consider their results strictly in terms of the size of the attack and the resulting number of casualties, under the assumption that the cost is linear for the number of people infected.

6.24.8 Goals of our Pursuits

The objective our experiments are to consider a number of issues in light of an epidemic. First, we wish to consider an unbounded approach in which cost is not a factor and the objective is simply to provide a game-theoretical equilibrium in the form of an upper bound on the expected casualty rate in relation to all possible attacks. Second, we wish to consider the benefits of how many individuals are not infected vs. the cost to insure their health. Essentially, we will gauge the most cost effective approach that yields the greatest protection of the population per dollar spent. Finally, there is the issue of budget limitations on the governing body. All options will be considered in light of various levels of budgets. This is distinct from cost-benefit analysis as life is considered invaluable, but the resources are limited.

6.25 The Outcome of the Experiments

6.25.1 Casualties Only

Amidst all defensive choices, the lowest possible number of infections regardless of a change in attack vectors was a combination of 200,000 inoculations, closing of schools on day 1, +10% boost to recovery rates and -10% to infection rates on attack vector 2-9, with 29 infections. This was followed by the same setup with the exception of schools closed on day 2 and day 4, yielding 30 and 31 infections respectively. The nature of the social network in our experiment creates situations in which effects can be linearly effective but non-linearly dominant due to the chaotic nature of the system. A look at the day 3 closing shows that it actually had the same number of infections as a day 1 closing but if the enemy switches to attack vector 3-9 they can actually increase the number of casualties to 39, a 34% increase.

Looking at the change of infection upper bounds, we find that a few interesting situations arise. First, we note that at the 29th lowest upper bound there is a serious spike in the change from the prior scenario. This indicated the boundary between where the options were strictly 200,000 inoculations. However, starting at the 33rd place and moving onwards, the entries are not strictly ordered by inoculation. At the 61st entry, we find the first scenario where no inoculations occurred, resulting in a lower bound of 5,197 infections.

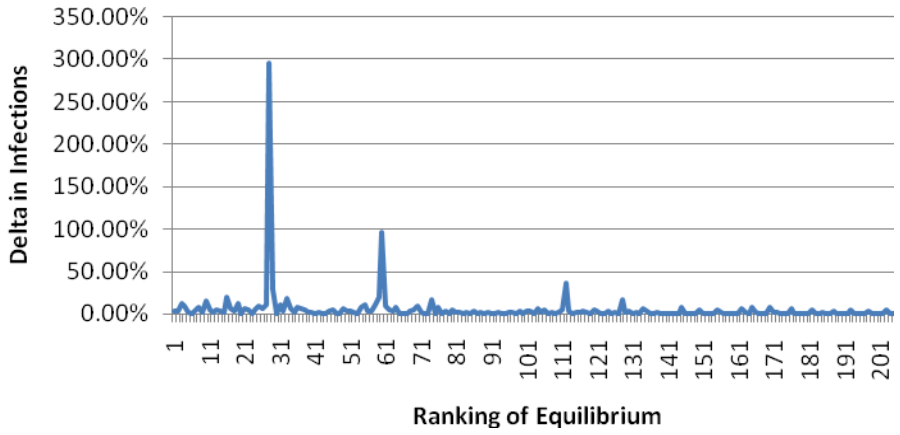

Figure 6.1. The increase in infections over a ranked list of upper-bounded scenarios.

If closings are not an option, the nature of the best possible choices changes only slightly. The lowest possible equilibrium in this instance is 35 infections amidst 200,000 inoculations with a recovery rate of +10% and an infection rate of -5% on attack vector 1-9. The next best scenarios are attack vector 3-9 with a maximal intervention rate yielding an upper bound of 38 and the same infection rate modification but only +5% to the recovery yielding 57 infected individuals. The rest of the equilibriums were ordered properly according to the inoculation count. However, the altercations to the infection and recovery rates varied on which one them were the upper boundary.

When vaccines are not available, the focus shifts to closing as many facilities as reasonably possible as the primary motivation. We added closing all facilities to our matrix of possibilities in this situation. The best scenario here is everything is closed on day 1 and recovery is bolstered to 10% with infection is minimally affected, yielding 5,198 infections total. When everything cannot be closed, the next best option becomes of course work and schools closed on day 1 with maximum intervention but resulting in 41,593 reported cases. School or no closings alone are not even considered until the first two approaches have been exhausted.

6.25.2 Cost Effective Defense

When costs are introduced, and equilibriums are based on the number of deaths achieved, the nature of the ideal situation changes. The cheapest scenario has a total of 35 infections and a cost of $624,062, with half of the city inoculated, an attack via vector 1-9, a recovery adjustment of +10% but an infection adjustment of only -5% and no closings at all. The next ideal scenario has the same inoculation count but an attack on vector 2-9 with schools closed on day 1 and maximum adjustments to infection and maximum intervention policies, at a cost of $625,284 but an infection count of only 35. Serious price increases arrive on the ranked list of scenarios when 100,000 inoculations are performed, as the infection count here enters the quadruple digits at 6188 and the cost skyrockets to $8,897,963. When businesses are considered, the best business closing scenario sees an increase of $321 million compared to the 'worse' non-work-closing scenario starting at a rank of 81st. From that point on, all options consider businesses closed.

Hypothetically, assume that the cost of the vaccine has now gone up due to concerns that other cities have over the same type of attack. Instead of a wholesale price of $2.76, the

price is increased by a factor of 10 to $27.60 because of demand. No change is found to be in the order of the list of options by cost; however, the cheapest option is now nearly $5.6 million. If cost of the vaccine increases by an additional factor of ten, inoculating 50% of the city is now considered too expensive when compared the benefits of only 25%. The minimum amount necessary to seriously consider a 25% inoculation is $85.50 per dose. When no vaccine is available at all, the cheapest option becomes closing schools on day 3 with minimal infection and recovery rate modifications, which has equilibrium at attack vector 3-9 with a staggering total of 73,718 infections.

6.25.3 Budgeted Defense

Judging an effective economic approach on limited funding can be particularly difficult. Regardless of how it is viewed, human life must be valued at a finite price in order to maximize the existing resources. We apply the cost incurred during the course of fighting the epidemic in regards to the number of individuals who do not become infected. This individual value essentially becomes the price of health. Applying this concept to our data, we find that the most cost-effective upper bound is through inoculation of half the city with no closings, a 10% boost to recovery and only a 5% discount to infection with the most effective attack being vector 1-9. This scenario only required $1.56 for every healthy individual. If no inoculations were available, the lowest price jumps to $236.98 per health individual, under attack vector 3-9, schools alone being closed on day 3, and minimal intervention.

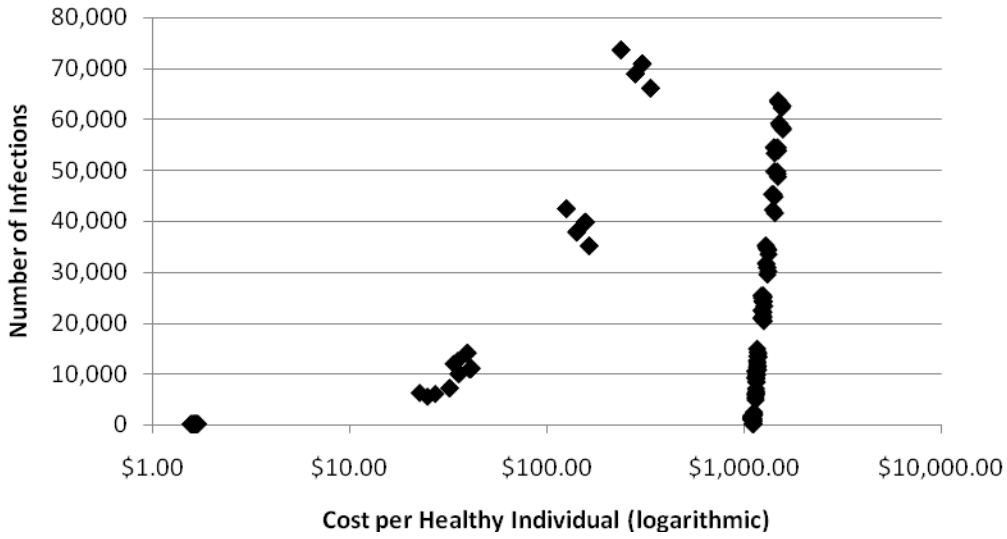

Figure 6.2. Logarithmic x-axis graph of the number of infections vs. the cost per healthy individual

Analysis of health costs and the resulting infections show an interesting trend. Price points for health begin at under two dollars. However, the cost continues to increase as the money invested cannot match pace with the yield. Some of the lowest infection counts can be as high as possible due to the nature of the techniques. In this case, several of the defensive techniques that resulted in under 100 infections were due to closing all social theatres in light of the threat.

Consider terrorists that must linearly scale up their efforts for the larger attack sizes per vector. For example, infecting 3 people at target 1 requires half as many resources as infecting 6 people. As stated previously, it is difficult to assign a value in the mindset of the terrorist. However, by simply dividing the number of individuals affected by the size of the attack, the costs are appropriately weighted. Generally speaking, this approach tends to result in fewer infections. The lowest equilibrium in the list is 12 infections with half of the population inoculated, full intervention methods, and schools closed on day 1 with attack

vector 2-3. When the list is resorted in terms of cost to the city, the first 7 top choices remain the same; however, 7th place shifts from 22 infections on the same recovery and attack vector but only +5% to recovery rates to 23 infections with a recovery of +10% and the infection rate now at -5%. If the same shift in cost evaluation is applied to the group's desire to inflict the highest amount of economic damage for their investment, the impact of their attack becomes even lower, with attack vector 1-3 yielding only 3 infections in an ideal setup of 50% inoculations, everything closed on day 1, and all but the infection rate at maximum intervention levels.

Overall, we found that the viability of options available to a defending country can be heavily dependent on the local economy. For many cities in relatively rich countries, these costs may seem to be a small price to pay for the safety of the citizens present. However, those which live in countries that lack available resources may find that sacrifices are an inevitable part of the situation. Additionally, if the cost of certain methods rise considerably, no city may be safe from the impact of a particularly virulent weapon. In the end, all options must be considered in light of the greater good to society. This said, with careful planning and the justification such a model as ours offers, it becomes possible to anticipate and ensure that the necessary resources are in place long before a real attack may occur.

CHAPTER 7

CONCLUSIONS

Our work on surveillance techniques demonstrated that automated data analysis remains a difficult problem. Replicating the ability of the human mind to process and correlate data is a complex task. Clearly, a country wishing to gather intelligence must rely even more on the resources available to human agents and the benefits of information sharing. The continuing flood of data as technology advances will undoubtedly pose a growing challenge to the world of data mining for security.

Our simulations of data sharing and cooperation have shown that it is indeed possible to achieve a situation in which peers successfully ensure the ideal choice of strategy by all participants. However, just as in many societies today, there is a continual need for a few to sacrifice their own gains to ensure the rest of the alliance will benefit. Finding ways in which peers can reward each other for adopting this behavior is crucial to ensuring an equilibrium can exist in the real world. However, if those that make this sacrifice are not properly valued and rewarded for their efforts, it is difficult to ascertain how long an ideal situation would last.

Throughout all experiments, it became clear that inoculations are a highly-effective and relatively low-cost way of fighting a biological attack. However, an actual attack would likely run the risk of alarming more than just one city. Unless the governing body has the power to fix the cost of the vaccine, there is a high probability that a free market will continue to drive up prices as not only governing bodies attempt to protect the population but

concerned individuals as well. Additionally, the inoculation method of protection operates under the assumption that there is a warning far enough in advance either directly or through intelligence gathering that indicates which contagion will be used. The truth is, even if this is known, a single vaccine may require months to prepare in order to deal with a weaponized or exotic strain.

Additionally, some vaccines carry risks. The smallpox vaccine alone is recognized by the CDC for having potentially life-threatening side effects and in some cases is not recommended unless an emergency need arises. [75] In the event that panic becomes the motivating factor for a governing body, mass inoculations further risk those being protected when not enough attention is being given to the patient to watch for the side effects. According to the CDC report, 1 out of every 1,000 patients administered the smallpox vaccine faced serious reactions. If half of the population in our virtual city was inoculated, that suggests that roughly 200 of them may experience these side effects. In the event that the attack does not occur, one may argue that the vaccine itself did more harm than the terrorists themselves. Clearly, vaccinations must be considered carefully before being applied to the situation.

Judging a defensive strategy through a cost-benefit analysis appears initially to be a strictly economic maneuver when lives and well-being are at stake. The value of human life is a widely debated topic. However, it is important to consider that a bioterrorism event may be one of several challenges ahead for a governing entity to consider. Funding is limited regardless of how much is made, and assigning every resource to deal with a single event can be potentially hazardous should a future one arrive. This considered, recent history has shown that the power of epidemics should never be underestimated. With some diseases,

failure to act may very well result in a pandemic that would either leave little only a fraction of the city's inhabitants alive or take a considerable toll on the rest of the world.

The intervention techniques of modifying the infection and recovery rates seemed to be too small to be of use. Although they did have an impact, it appears that they were regularly 'drowned' in the mathematical sense within precision errors and the rest of the factors involved. In essence, whenever the simulation appeared to have a relatively small number of infections due to other methods, the cost-benefit of using such low levels of intervention were negligible at best. It is our belief from analysis of these actions in our simulation that suggests that this technique does not stack well with others despite being mathematically compatible.

One of the most important aspects of this simulation's relevance is the availability of real-world data to simulate a given city. Traditional census data can be used to provide detailed snapshots of how individuals spend their time over the course of a day at the various social theatres. Family sizes and population distribution can be used to construct home social theatres. School enrollment records, public service records, and employment information can all be combined in such a way to represent an entire social network within the model itself. The end result is that the model itself can be further enhanced with readily accessible public information that is traditionally available at most state or county levels. Essentially, any city of virtually any size can be evaluated in light of this model with an appropriate set of computational resources and accurate data.

A powerful factor that we repeatedly observed in our results in the epidemiology models was the small world phenomenon. The small world effect is when the average number of hops on even the largest of networks tends to be very small. In several cases, the infection spread to

enough individuals within 4 days to pose a serious threat that could not be easily contained. The results from closing social theatres made this particularly clear, as many closings beyond the third day did little to slow the advance of many epidemics. It is crucial that research continues in the areas of epidemic identification, intelligence gathering, and particularly early warning systems to ensure that action can be taken within a short period of time.

It is important to realize that not all intervention methods are available in every country. It is important to understand how local governmental powers, traditions and ethics can impact the options available in a simulation. In some countries, a government may be able to force citizens to be vaccinated, while others may have no power at all and must rely on the desire for protection to motivate action. In other situations, closing any social theatre may be an explicit power of the state, in contrast to governing entities that may have varying amount of abilities to do the same but will not consider it due to a severe social backlash. The impact on society must be carefully considered beyond economical cost in any course of action, and there is rarely a clear choice. These answers are outside the scope of our work. However, our model helps governing bodies consider these efforts carefully in light of public safety and the expenditure of available resources.

The ultimate goal of our research has and continues to be an effort to take advantage of the benefits of computer science and provide a means to further pursue a higher level of security against malicious activities. This work is a culmination of years of research into the application of social sciences to computer science in the realm of modeling and simulation. With detailed demographic data and knowledge of an impending biological attack, this model provides the means to both anticipate the impact on a population and potentially prevent a serious epidemic. An emphasis on cost-benefit analysis of the results could

potentially save both lives and resources, both of which can be invested in further refining security for a vulnerable population.

The fight for security has been a long journey for the field of science. The increasing sophistication of would-be attackers continues to grow. It is unfortunately not inconceivable for a biological weapon to be used to advance an ideology or agenda against innocent civilians in this age. There is also little doubt that new threats and forms of attack will emerge due to the tenacious nature and ingenuity of mankind. Therefore, those seeking to improve security must never be lax in their pursuits for new means to protect. It is our hope that techniques such as those outlined in our work are never needed to face a real bioterrorism threat. We hope that the future holds a day where such attacks are never considered a remote possibility. However, to quote Nathaniel Brandon, “In a world in which the total of human knowledge is doubling about every ten years, our security can rest only on our ability to learn.”

BIBLIOGRAPHY

[1] Ben-Dov, Moty, et al., "Improving Knowledge Discovery by Combining Text-Mining and Link-Analysis Techniques." *SIAM International Conference on Data Mining.* [Online] 2004. [Cited: February 8, 2005.] http://www.uclic.ucl.ac.uk/paul/research/Moty1.pdf.

[2] Horrocks, Ian and Patel-Schneider, Peter., "Three Theses of Representation in the Semantic Web." 2003. Proceedings of the twelfth international conference on the World Wide Web. pp. 39-47.

[3] Axelrod, Robert., *The Evolution of Cooperation.* New York : Basic Books, 1985.

[4] Gupta, Rohit and Somani, A. K., "Game theory as a tool to strategize as well as predict nodes' behavior in peer-to-peer networks." 2005. Proceedings from the 11th International Conference on Parallel and Distributed Systems.

[5] Buragohain, C., Agrawal, D. and Suri, S., "A game theoretic framework for incentives in P2P systems." 2003. Proceedings from the Third International Conference on Peer-to-Peer Computing.

[6] M. Seredynski, P. Bouvry, M. A. Klopotek., "Modelling the Evolution of Cooperative Behavior in Ad Hoc Networks using a Game Based Model." *Computation Intelligence and Games.* 2007, pp. 96-103.

[7] R.G.Cascella., "The 'Value' of Reputation in Peer-to-Peer Networks." *Consumer Communications and Networking Conference.* 2008, pp. 516-520.

[8] J.C. Oh, N. Gemelli, R. Wright., "A Rationality-based Modeling for Coalition Support." *Intelligent Systems.* 2004, pp. 172-177.

[9] Carley, K., et al., "BioWar: scalable agent-based model of bioattacks." *Transactions on Systems, Man and Cybernetics, Part A.* Vol. 6, 2.

[10] Eubank, Stephen., "Network Based Models of Infectious Disease Spread." *Japan Journal of Infectious Diseases.* 2005, Vol. 6, 58, pp. 9-13.

[11] Eubank, S., et al., "Modeling disease outbreaks in realistic urban social networks." *Nature.* May 2004, Vol. 429, pp. 180-184.

[12] Bonnett, J., "High Performance Computing: An Agenda for the Social Sciences and the Humanities in Canada." *Social Sciences and Humanities Research Council of*

Canada. [Online] 2006. http://www.sshrc.ca/web/about/publications/computing_final_e.pdf.

[13] Satuma, J., et al., "Extending the SIR epidemic model." Physica A, 2004, Vol. 336, pp. 369-375.

[14] Fuk´s, H., Lawniczak, A. and Duchesne, R., "Effects of population mixing on the spread of SIR epidemics." European Physical Journal, 2006, Vol. 50, pp. 209-214.

[15] Kress, M., "The Effect of Social Mixing Controls on the Spread of Smallpox—A Two-Level Model." Health Care Management Science, 2005, Vol. 8, pp. 277-289.

[16] Pequegnat, W., et al., "Conducting Internet-Based HIV/STD Prevention Survey Research: Considerations in Design and Evaluation." Aids Behavior, 2007, Vol. 11, pp. 505-521.

[17] Stattenspiel, L. and Herring, D., "Simulating the Effect of Quarantine on the Spread of the 1918-19 Flu in Central Canada." Vol. 65, pp. 1-26.

[18] Ryder, J. J., et al., "Meassuring the transmission dynamics of sexually transmitted disease." Proceedings of the National Academy of Sciences in the United States of America, 2005, Issue 42, Vol. 102, pp. 15140-15143.

[19] Stattenspiel, L. and Dietz, K., "A Structured Epidemic Model Incorporating Geographic Mobility Among Regions." Mathmatical Biosciences, 1995, Vol. 128, pp. 71-91.

[20] Bower, B., "Sniffle-Busting Personalities: Positive mood guards against getting colds." *Science News.* 2006, Vol. 170, 25, p. 387.

[21] Moghadas, S., "Gaining insights into human viral diseases through mathematics." European Journal of Epidimiology, 2006, Vol. 21, pp. 337-342.

[22] Meyers, L. A., "Contact network epidemiology: Bond percolation applied to infectious disease prediction and control." Bulletin of the American Mathematical Society, 2007, Vol. 44, pp. 63-86.

[23] Patlolla, Padmavathi, et al., "Agent-Based Simulation Tools in Computational Epidemiology." *Lecture Notes in Computer Science.* Heidelberg : Springer Berlin, 2006, pp. 212-223.

[24] Cojocaru, M., Bauch, C and Johnston, M., "Dynamics of Vaccination Strategies via Projected Dynamical Systems." Bulletin of Mathematical Biology, 2007, Vol. 69, pp. 1453-1476.

[25] Banks, D. and Anderson, S., "Game Theory and Risk Analysis in the Context." Statistical Methods in Counterterrorism, 2006, pp. 9-22.

[26] Skillicorn, David., Detecting Unusual and Deceptive Communication in Email. *Queen's University.* [Online] [Cited: June 15, 2005.] http://www.cs.queensu.ca/TechReports/Reports/2005-498.pdf.

[27] Layfield, Ryan., "Link Analysis of Social Activity and Suspicious Topic Propagation." Presentation at the 2005 South Central Information Security Symposium.

[28] Han, Jiawei and Kamber, Micheline., *Data Mining: Concepts and Techniques.* s.l. : Morgan Kaufman, 2000.

[29] Will, Todd., Introduction to Singular Value Decomposition. *The University of Wisconsin La Crosse.* [Online] November 3, 2003. [Cited: February 10, 2005.] http://www.uwlax.edu/faculty/will/svd/index.html.

[30] Internet Usage Statistics – The Big Picture. *Miniwatts, Inc.* [Online] March 24, 2005. [Cited: April 12, 2005.] http://www.internetworldstats.com/stats.htm.

[31] Global Internet Statistics (by Language). *Global Reach.* [Online] [Cited: April 12, 2005.] http://www.glreach.com/globstats/.

[32] Goldberg, David, et al., "Collaborative Filtering to Weave an Information Tapestry." *Communications of the ACM.* 1992, Vol. 35, 12, pp. 61-70.

[33] Pazzani, Michael., "Representation of Electronic Mail Filtering Profiles: A User Study." 2000. International Conference on Intelligent User Interfaces. pp. 202-206.

[34] Krebis, Vladis., An Introduction to Social Network Analysis. *orgnet.com.* [Online] 2005. [Cited: April 12, 2005.] http://www.orgnet.com/sna.html.

[35] Perin, Constance., "Electronic Social Fields in Bureaucracies." *Communications of the ACM.* 1991, Vol. 35, 12, pp. 75-82.

[36] Auguston, J. Gary and Minker, Jack., "An analysis of some graph theoretical clustering techniques." *Journal of the ACM.* October 1970, Vol. 17, 4, pp. 571-588.

[37] Kishnamurthy, Balachander, and Jia Wang., "Topology Modeling via Cluster Graphs." *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement.* 2001, pp. 19-23.

[38] A.De Paola, A. Tamburo,., "Reputation Management in Distributed Systems." *2008 IEEE Symposium on Game theory, evolutionary approach, distributed systems.* March 2008.

[39] S. Kamvar, M. Schlosser, H. Garcia-Molina., "The Eigentrust algorithm for reputation management in P2P networks." *Proceedings of the 12th international conference on World Wide Web.* 2003, pp. 640-651.

[40] R. Morselli, J. Katz, B. Bhattacharjee., "A Game-Theoretic Framework for Analyzing Trust-Inference Protocols." *Workshop on Economics of Peer-to-Peer Systems.* 2004.

[41] Drew Fudenberg, Jean Tirole., *Game Theory.* Cambridge : MIT Press, 1991.

[42] Sepandar D. Kamvar, Mario T. Schlosser, Hector Garcia-Molina., "The EigenTrust Algorithm for Reputation Management in P2P Networks." Budapest : s.n., 2003. Proceedings of the Twelfth International World Wide Web Conference.

[43] Welcome. *The Colt Project.* [Online] CERN - European Organization for Nuclear Research, September 9, 2004. [Cited: August 1, 2008.] http://acs.lbl.gov/~hoschek/colt/.

[44] Grossman, W., "New Tack Wins Prisoner's Dilemma." *Wired Magazine.* 10 13, 2004.

[45] Shadel, B., et al., "Infection control practitioners' perceptions and educational needs regarding bioterrorism: Results from a national needs assessment survey." American Journal of Infection Control, 2003, Vol. 31, pp. 129-134.

[46] Booth, Christopher M., et al., "Clinical Features and Short-term Outcomes of 144 Patients With SARS in the Greater Toronto Area." Journal of the American Medical Association, May 6, 2003, Issue 21, Vol. 289, pp. 2801-2809.

[47] Shulgin, B., Stone, L. and Agur, Z., "Pulse Vaccination Strategy in the SIR Epidemic Model." 1998, Vol. 60, pp. 1123-1148.

[48] Floyd, W., Kay, L. and Shapiro, and M., "Some elementary properties of SIR networks or, can i get sick because you got vaccinated?" Bulletins of Mathmatical Biology, December 1, 2007, Issue 3, Vol. 70, pp. 713-27.

[49] Eames, K. T. and Keeling, M. J., "Modeling dynamic and network heterogeneities in the spread of sexually transmitted diseases." Proceedings of the National Academy of Sciences in the United States of America, 2002, Issue 20, Vol. 99, pp. 13330-13335.

[50] Khuroo, M., "Discovery of Hepatitis E Virus – The Untold Story." JK-Practitioner, 2004, Issue 3, Vol. 11, pp. 291-294.

[51] Salovey, P., et al., "Emotional States and Physical Health." American Psychologist, 2000, Issue 1, Vol. 55, pp. 110-121.

[52] Granovetter, M. S., "Changing jobs: Channels of mobility information in a suburban community." *Unpublished PhD dissertation.* s.l. : Harvard University, 1970.

[53] Layfield, R., Kantarcioglu, M. and Thuraisingham, B., "Simulating Bioterrorism through Epidemiology Approximation." IEEE International Conference on Intelligence and Security Informatics, 2008, pp. 82-87.

[54] Dillon, K. M., Minchoff, B. and Baker, K. H., "Positive emotional states and enhancement of the immune system." International Journal of Psychiatry in Medicine, 1985, Vol. 15, pp. 13-18.

[55] Baron, D. A., Hay, D. A. and Easom, H. M., "Treating Patients in Primary Care: The Impact of Mood, Behavior, and Thought Disturbances." Journal of the American Osteopathic Association, 2003, Issue 7, Vol. 103, pp. 319-329.

[56] Waldschmidt, T., Cook, R. and Kovacs, E., "Alcohol and Inflammation & Immune Responses: Summary of the 2006 Alcohol and Immunology Research Interest Group (AIRIG) meeting." Alcohol, 2006, Issue 2, Vol. 42, pp. 137-142.

[57] Borum, Randy., *Psychology of Terrorism.* Tampa : University of Florida, 2004.

[58] Myerson, Roger B., *Game Theory: Analysis of Conflict.* s.l. : Harvard University Press, 1997.

[59] *Bioterrorism Preparedness.* Austin : Texas Institute for Health Policy Research, 2001.

[60] "The Business of Terror: Conceptualizing Terrorist Organizations." *Center for Defense Information.* [Online] May 23, 2005. http://www.cdi.org/pdfs/terrorist-business-model.pdf.

[61] "OECD CONFERENCE ON CATASTROPHIC RISKS AND INSURANCE." Paris : Organisation for Economic Cooperation and Development, 2004. Catastrophic Risks and Insurance. pp. 1-3.

[62] "Terrorism Risk Insurance Program." *United States Department of Treasury.* [Online] January 8, 2007. http://www.ustreas.gov/offices/domestic-finance/financial-institution/terrorism-insurance/.

[63] Osbourne, Martin J. and Rubinstein, A., *A Course in Game Theory.* Cambridge, Mass. : MIT Press, 1994.

[64] Meltzer, Martin I., Damon, Inger and LeDuc, James W., "Smallpox as a Bioterrorist Weapon." Emerging Infectious Disease, 2001.

[65] Li, Liuyi, Cheng, Suhua and Gu, Jiang., "SARS Infection Among Health Care Workers in Beijing, China." Journal of American Medical Association, 2003, Vol. 290, pp. 2662-2663.

[66] Guidance for Persons Who May Have Been Exposed to Severe Acute Respiratory Syndrome (SARS). *Center for Disease Control.* [Online] May 3, 2005. http://www.cdc.gov/ncidod/sars/pdf/exposuremanagement-sars.pdf.

[67] Borenstein, Seth., "AP IMPACT: An American life worth less today." *Associated Press.* July 10, 2008.

[68] "Deaths: Final Data for 2005." National Vital Statistics Report, April 24, 2008, Issue 2, Vol. 54, pp. 1-121.

[69] Hourly Rate Survey Report for Job: Registered Nurse. *PayScale.* [Online] October 30, 2008. http://www.payscale.com/research/US/Job=Registered_Nurse_(RN)/Hourly_Rate.

[70] HHS AWARDS $428 MILLION CONTRACT TO PRODUCE SMALLPOX VACCINE. *United States Department of Health & Human Services.* [Online] November 28, 2001. http://www.hhs.gov/news/press/2001pres/20011128.html.

[71] 2007 World Population Datasheet. *Population reference bureau.* [Online] http://www.prb.org/Publications/Datasheets/2007/2007WorldPopulationDataSheet.aspx.

[72] The World Factbook. *Central Intelligence Agency.* [Online] October 23, 2008. https://www.cia.gov/library/publications/the-world-factbook/.

[73] Tax Stats at a Glance. *Internal Revenue Service, United States Department of Treasury.* [Online] June 4, 2008. http://www.irs.gov/taxstats/article/0,,id=102886,00.html.

[74] State Government Tax Collections: 2007. *US Censur Bureau.* [Online] February 2008, 2007. http://www.census.gov/govs/statetax/0700usstax.html.

[75] Smallpox fact sheet: vaccine overview. *Center for Disease Control.* [Online] February 7, 2007. http://www.bt.cdc.gov/agent/smallpox/vaccination/facts.asp.

[76] Skillicorn, David., "Keyword Filtering for Message and Conversation Detection." *Queen's University.* [Online] [Cited: February 14, 2005.] http://www.cs.queensu.ca/home/skill/beyondkeywords.pdf.

[77] Kishnamurthy, Balachander and Wang, Jia., "Topology Modeling via Cluster Graphs." Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement, 2001, pp. 19-23.

[78] Cohen, Brian., "Incentives Build Robustness in BitTorrent." 2003. Proceedings of the 1st Workshop on Economics of Peer-to-Peer Systems.

[79] Halpern, Joseph and Teague, V., "Rational secret sharing and multiparty computation: extended abstract." 2004. Proceedings of the thirty-sixth annual ACM symposium on Theory of computing.

[80] Harsanyi, John C., "Games with incomplete information played by ‘Bayesian’ players." Management Science, 1967.

[81] Nash, John., "Equilibrium Points in n-Person Games." Proceedings of the National Academy of Sciences USA, 1950, Vol. 36, pp. 48-49.

[82] Scott, John., *Social Network Analysis : a Handbook.* London : Sage Publications, 1985.

[83] Attayoor, S. P., et al., "A Cholera Epidemic Among the Nicobarese Tribe of Nancowry, Andaman, and Nicobar, India." American Society of Tropical Medicine and Hygene, 2004, Issue 6, Vol. 71, pp. 822-827.

[84] Booth, C., "Clinical Features and Short-term Outcomes of 144 Patients With SARS in the Greater Toronto Area." Journal of the American Medical Association, 2003, Issue 21, Vol. 289.

[85] Kress, M., "The Effect of Social Mixing Controls on the Spread of Smallpox—A Two-Level Model." Health Care Management Science, 2005, Vol. 8, pp. 277-289.

[86] Miyoshi, T., et al., "Characteristics of Norovirus Outbreaks during a Non-Epidemic Season." Japan Journal of Infectious Diseases, 2006, Vol. 59, pp. 140-141.

[87] Moghadas, S., "Gaining insights into human viral diseases through mathematics." European Journal of Epidimiology, 2006, Vol. 21, pp. 337-342.

[88] M. Seredynski, P. Bouvry, M.A. Klopotek., "Modelling the Evolution of Cooperative Behavior in Ad Hoc Networks using a Game Based Model." *Computation Intelligence and Games.* 2007, pp. 96-103.

[89] R. Layfield, M. Kantarcioglu, B. Thuraisingham., "Proceedings of the 21st Annual Working Conference on Data and Applications Security." 2007. Enforcing Honesty in Assured Information Sharing Within a Distributed System.

Part III-A

Handling Untrustworthy Partners: Defensive Operations

Collected Papers on Data Mining for Cyber Security Applications

Flow-based Identification of Botnet Traffic by Mining Multiple Log Files

Mohammad M. Masud[1], Tahseen Al-khateeb[2], Latifur Khan[3]
Bhavani Thuraisingham[4], Kevin W. Hamlen[5]
Department of Computer Science, The University of Texas at Dallas
Richardson, TX 75080, USA
{[1]mehedy,[2]tahseen,[3]lkhan}@utdallas.edu
{[4]bhavani.thuraisingham,[5]hamlen}@utdallas.edu

***Abstract*—Botnet detection and disruption has been a major research topic in recent years. One effective technique for botnet detection is to identify Command and Control (C&C) traffic, which is sent from a C&C center to infected hosts (bots) to control the bots. If this traffic can be detected, both the C&C center and the bots it controls can be detected and the botnet can be disrupted. We propose a multiple log-file based temporal correlation technique for detecting C&C traffic. Our main assumption is that bots respond much faster than humans. By temporally correlating two host-based log files, we are able to detect this property and thereby detect bot activity in a host machine. In our experiments we apply this technique to log files produced by `tcpdump` and `exedump`, which record all incoming and outgoing network packets, and the start times of application executions at the host machine, respectively. We apply data mining to extract relevant features from these log files and detect C&C traffic. Our experimental results validate our assumption and show better overall performance when compared to other recently published techniques.**

***Keywords*- Malware, botnet, intrusion detection, data mining.**

I. Introduction

Botnets are emerging as "the biggest threat facing the internet today" [1] because of their enormous volume and sheer power. Botnets containing thousands of *bots* (compromised hosts) have been tracked by several different researchers [2], [3]. Bots in these botnets are controlled from a *Command and Control* (C&C) center, operated by a human *botmaster* or *botherder*. The botmaster can instruct these bots to recruit new bots, launch coordinated DDoS attack against specific hosts, steal sensitive information from infected machines, send mass spam emails, and so on. Fig. 1 illustrates a typical botnet architecture.

Numerous researchers are working hard to combat this threat and have proposed various solutions [4], [5], [2]. One major research direction attempts to detect the C&C center and disable it, preventing the botmaster from controlling the botnet. Locating the C&C center requires identifying the traffic exchanged between it and the bots. Our work adopts this approach by using a data mining based technique to identify temporal correlations between multiple log files. We maintain two different log files for each host machine: (i) a network packet trace or `tcpdump`, and (ii) an application execution trace or `exedump`. The `tcpdump` log file records all network packets that are sent/received by the host, and the `exedump` log file records the start times of application program executions on the host machine. Our main assumption is that bots respond to commands much faster than humans do. Thus, the command latency (i.e., the time between receiving a command and taking actions) should be much lower, and this should be reflected in the `tcpdump` and `exedump` log files.

Bot commands that have an observable effect upon the log files we consider can be grouped into three categories: those that solicit a response from the bot to the botmaster, those that cause the bot to launch an application on the infected host machine, and those that prompt the bot to communicate with some other host (e.g., a victim machine or a code server). This botnet command categorization strategy is explained in more detail in Section III. We apply data mining to learn temporal correlations between an incoming packet and (i) an outgoing packet, (ii) a new outgoing connection, or (iii) an application startup. Any incoming packet correlated with one of these logged events is considered a possible botnet command packet. Our approach is flow-based because rather than classifying a single packet as C&C or normal traffic, we classify an entire flow (or connection) to/from a host as C&C or normal. This makes the detection process more robust and effective. Our system is first trained with log files obtained from clean hosts and hosts infected with a known bot, then tested with logs collected from other hosts. The testing methodology is explained in detail in Section IV.

Our technique is different from other botnet detection techniques [5], [6], [2] in two ways. First, we do not impose any restriction on the communication protocol. Our approach should therefore also work with C&C protocols other than those that use IRC as long as the C&C traffic possesses the observable characteristics defined above. Second, we do not rely on command string matching. Thus, our method should work even if the C&C payloads are not available.

Our work makes two main contributions to botnet detection research. First, we introduce multiple log correlation for C&C traffic detection. We believe this idea could be successfully extended to additional application-level logs such as those that track process/service execution, memory/CPU utilization, and disk accesses. Second, we have proposed a way to classify botmaster commands into different categories, and we show

Fig. 1. A typical IRC-based botnet architecture

how to utilize these command characteristics to detect C&C traffic. An empirical comparison of our technique with another recent approach [5] shows that our strategy is more robust in detecting real C&C traffic.

The rest of the paper is organized as follows: Section II discusses related work on botnet detection. Section III discusses our system setup and data collection process. Section IV explains our botnet detection architecture and discusses the details of the detection process. Section V evaluates our system. Finally, Section VI concludes by summarizing our work and suggesting future research directions.

II. Related Work

Botnet defenses are being approached from at least three major perspectives: analysis, tracking, and detection. Barford and Yegneswaran [7] present a comprehensive analysis of several botnet codebases and discuss various possible defense strategies that include both reactive and proactive approaches. Grizzard et al. [4] analyze botnets that communicate using peer-to-peer networking protocols, concluding that existing defense techniques that assume a single, centralized C&C center are insufficient to counter these decentralized botnets.

Freiling et al. [3] summarize a general botnet-tracking methodology for manually identifying and dismantling malicious C&C centers. Rajab et al. [2] put this into practice for a specific IRC protocol. They first capture bot malware using a honeynet and related techniques. Captured malware is next executed in a controlled environment to identify the commands that the bot can receive and execute. Finally, *drone machines* are deployed that track botnet activity by mimicking the captured bots to monitor and communicate with the C&C server. Dagon et al. [8] track botnet activity as related to geographic region and time zone over a six month period. They conclude that botnet defenses such as those described above can be more strategically deployed if they take into account the diurnal cycle of typical botnet propagation patterns.

Our research presented in this article is a detection technique. Cooke et al. [9] discuss various botnet detection techniques and their relative merits. They conclude that monitoring C&C payloads directly does not typically suffice as a botnet-detection strategy because there are no simple characteristics of this content that reliably distinguish C&C traffic from normal traffic. However, Goebel and Holz [6] show that botnets that communicate using IRC can often be identified by their use of unusual IRC channels and IRC user nicknames. Livadas et al. [5] use additional features including packet size, flow duration, and bandwidth. Their technique is a two-stage process that first distinguishes IRC flows from non-IRC flows, and then distinguishes C&C traffic from normal IRC flows. While these are effective detection techniques for some botnets, they are specific to IRC-based C&C mechanisms and require access to payload content for accurate analysis and detection. In contrast, our method does not require access to botnet payloads and is not specific to any particular botnet communication infrastructure. Karasaridis et al. [10] consider botnet detection from an ISP or network administrator's perspective. They apply statistical properties of C&C traffic to mine large collections of network traffic for botnet activity. Our work focuses on detection from the perspective of individual host machines rather than ISP's.

III. Bot traffic analysis

In this section we describe our system setup, data collection process, and approach to categorizing bot commands.

A. System setup

We tested our approach on two different IRC-based bots—SDBot version 05a [11] and RBot version 0.5.1 [12]. The testing platform consisted of five virtual machines running atop a Windows XP host operating system. The host hardware consisted of an Intel Pentium-IV 3.2GHz dual core processor with 2GB RAM and 150GB Hard Disk. Each virtual machine ran Windows XP with 256 MB virtual RAM and 8GB virtual Hard Disk space. The five virtual machines played the role of a botmaster, a bot, an IRC server, a victim, and a code server, respectively. As with a typical IRC-based botnet, the IRC server served as the C&C center through which the botmaster issued commands to control the bot. The IRC server we used was the latest version of Unreal IRCd Daemon [13], and the botmaster's IRC chat client was MIRC. The code server ran Apache Tomcat, and contained different versions of bot malware code and other executables. The victim machine was a normal Windows XP machine. During the experiment the botmaster instructed the bot to target the victim machine with udp and ping attacks. All five machines were interconnected in an isolated network as illustrated in Fig. 1.

B. Data collection

Data collection was performed in three steps. First, we implemented a client for the botmaster that automatically sent all possible commands to the bot. Second, we ran Windump [14] to generate a `tcpdump` log file, and ran our own implementation of a process tracer to generate a `exedump` log file. Third, we ran each bot separately on a fresh virtual machine, collected the resulting traces from the log files, and

TABLE I
SDBOT AND RBOT COMMAND CHARACTERISTICS

Observable effects	Commands					
	addalias	about	execute	udp	cmd	download
Bot-app	×	×	✓	×	✓	✓
Bot-response	×	✓	×	✓	✓	✓
Bot-other	×	×	×	✓	×	✓

then deleted the infected virtual machine. Traces were also collected from some uninfected machines connected to the internet. Each trace spanned a 12-hour period. The tcpdump traces amounted to about 3GB in total. Finally, these traces were used for training and testing.

C. Bot command categorization

Not all bot commands have an observable effect upon the log files we consider. We say that a command is *observable* if it matches one or more of the following criteria:

1) **Bot-response:** The command solicits a reply message from the bot to the C&C center. This reply is logged in the tcpdump. For example, the SDbot commands about and sysinfo are observable according to this criterion.
2) **Bot-app:** The command causes the bot to launch an executable application on the infected host machine. The application start event will be logged in the exedump. The execute command from SDbot is an example of such a command.
3) **Bot-other:** The command causes the bot to contact some host other than the C&C center. For example, the command might instruct the bot to send UDP packets as part of a DoS attack, send spam emails to other hosts, or download new versions of bot malware from a code server. Such events are logged in the tcpdump.

Some of the SDBot and RBot commands are listed in Table 1 and categorized using the above mentioned criteria. For a comprehensive description of these commands, please refer to [11], [12].

IV. ARCHITECTURE

Data collected via the procedure described in Section III was used for training and testing using the architecture illustrated in Fig. 2. For training, we first label each flow—i.e., each $(ip{:}port, ip'{:}port')$ pair—as a bot flow (conversation between a bot and its C&C center), or a normal flow (all other connections). Second, we compute several packet-level features (detailed below) for each incoming packet. Third, we compute flow-level features for each flow by aggregating the packet-level features. Finally, these flow-level features are used to train a classifier and obtain a classification model. For testing, we take an unlabeled flow and compute its flow-level features in the same way. Then we test the feature values against the classification model and label it a normal flow or a bot flow. The feature extraction process is explained next.

A. Feature Extraction

First we discuss the packet-level features, and then discuss the flow-level features. The intuitive idea behind these features is that human response to a command/request (e.g., a request to send a file or execute an application by his peer) should be much slower than a bot. In what follows, we refer to a packet as *incoming* if its destination is the host being monitored, and as *outgoing* if it originates from the monitored host.

a) Packet-level features.: The packet-level features we consider can be summarized as follows:

- **Bot-response (BR)** (boolean-valued): An incoming packet possesses this feature if it originated from some $ip{:}port$ and there is an outgoing packet to the same $ip{:}port$ within 100 ms of arrival of the incoming packet. This indicates that it is a potential command packet. The 100 ms threshold has been determined by our observation of the bots. We will refer to these incoming packets as *BR packets*.
- **BRtime** (real-valued): This feature records the time difference between a BR packet and its corresponding outgoing packet. This is an important characteristic of a bot.
- **BRsize** (real-valued): This feature records the length (in KB) of a BR packet. We observe that command packets typically have lengths of 1KB or less, whereas normal packets have unbounded size.
- **Bot-other (BO)** (boolean-valued): An incoming packet possesses this feature if it originated from some $ip{:}port$ and there is an outgoing packet to some $ip'{:}port'$ within 200 ms of the arrival of the incoming packet, where $ip' \neq ip$. This is also a potential command packet. The 200 ms threshold has also been determined by our observation of the bots. We will refer to these incoming packets as *BO* packets.
- **BODestMatch** (boolean-valued): A BO packet possesses this feature if outgoing destination ip' is found in its payload. This indicates that the BO packet is possibly a command packet that tells the bot to establish connection with host ip'.
- **BOtime** (real-valued): This feature records the time difference between a BO packet and its corresponding outgoing packet. This is also an important characteristic of a bot.
- **Bot-App (BA)** (boolean-valued): An incoming packet possesses this feature if an application starts on the host machine within 3 seconds of arrival of the incoming packet. This indicates that it is potentially command packet that instructs the bot to run an application. The 3

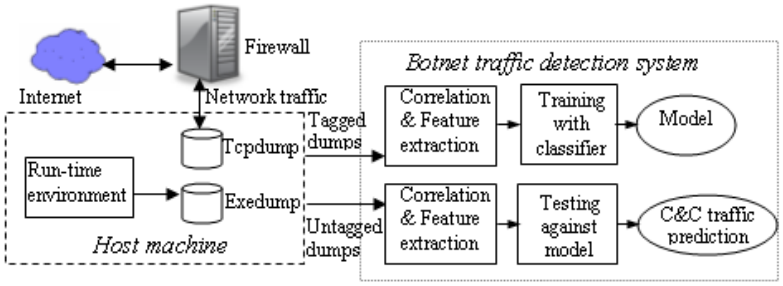

Fig. 2. System architecture

TABLE II
FLOW-LEVEL FEATURE SET

Feature	Description
AvgPktLen **VarPktLen**	Average and Variance of length of packets in KB
Bot-app	Number of BA packets as percentage of total packets
AvgBAtime **VarBAtime**	Average and Variance of BAtime of all BA packets
Bot-reply	Number of BR packets as percentage of total packets
AvgBRtime **VarBRtime**	Average and Variance of BRtime of all BR packets
AvgBRsize **VarBRsize**	Average and Variance of BRsize of all BR packets
Bot-other	Number of BO packets as percentage of total packets
AvgBOtime **VarBOtime**	Average and Variance of BOtime of all BO packets

second threshold has been determined by our observation of the bots. We will refer to these incoming packets as *BA* packets.

- **BAtime** (real-valued): This feature records the time difference between receiving a BA packet and the launching of the corresponding application.
- **BAmatch** (boolan-valued): A BA packet possesses this feature if its payload contains the name of the application that was launched.

b) Flow-level features.: As explained earlier, the flow-level features of a flow are the aggregations of packet-level features in that flow. They are summarized in Table 2. All flow-level features are real-valued. Also note that we do not use any flow-level feature that requires payload analysis.

B. Log file Correlation

Fig. 3 shows an example of multiple log file correlation. Portions of the `tcpdump` (left) and `exedump` (right) log files are shown in this example, side by side. Each record in the `tcpdump` file contains the packet number (No), arrival/departure time (Time), source and destination addresses (Src/Dest), and payload or other information (Payload/Info). Each record in the `exedump` file contains two fields: the process start time (Start Time), and process name (Process). The first packet (#10) shown in the `tcpdump` is a command packet that instructs the bot to download an executable from the code server and run it. The second packet (#11) is a response from the bot to the botmaster, so the command packet is a BR packet having $BRtime = 1ms$. The bot quickly establishes a TCP connection with the code server (other host) in packets #12–14. Thus, the command packet is also a BO packet having $BOtime = 7ms$ (the time difference between the incoming command and the first outgoing packet to another host). After downloading, the bot runs the executable `mycalc.exe`. Thus, this command packet is also a BA packet having $BAtime = 2.283s$.

C. Classification

We use a Support Vector Machine (SVM), Bayes Net, decision tree (J48), Naïve Bayes, and Boosted decision tree (Boosted J48) for the classification task. In our previous work [15] we have found that each of these classifiers demonstrates good performance for malware detection problems. Specifically, SVM is robust to noise and high dimensionality and can be fine-tuned to perform efficiently on a specific domain. Decision trees have a very good feature-selection capability and are much faster than many other classifiers both in training and testing time. Bayes Nets are capable of finding the inter-dependencies between different attributes. Naïve Bayes is also fast, and performs well when the features are independent of one another. Boosting is particularly useful because of its ensemble methods. Thus, each of these classifiers has its own virtue. In a real deployment, we would actually use the best among them.

D. Packet Filtering

One major implementation issue related to examining the packet traces is the large volume of traffic that needs to be scanned. We try to reduce unnecessary scanning of packets by filtering out the packets that are not interesting to us, such as the TCP handshaking packets (`SYN`,`ACK`,`SYNACK`) and NetBios session request/response packets.

V. EVALUATION

For evaluation, we tag all the flows as either bot flows or normal flows depending on whether the flow is between a bot and its C&C center or not. Then we extract feature-values

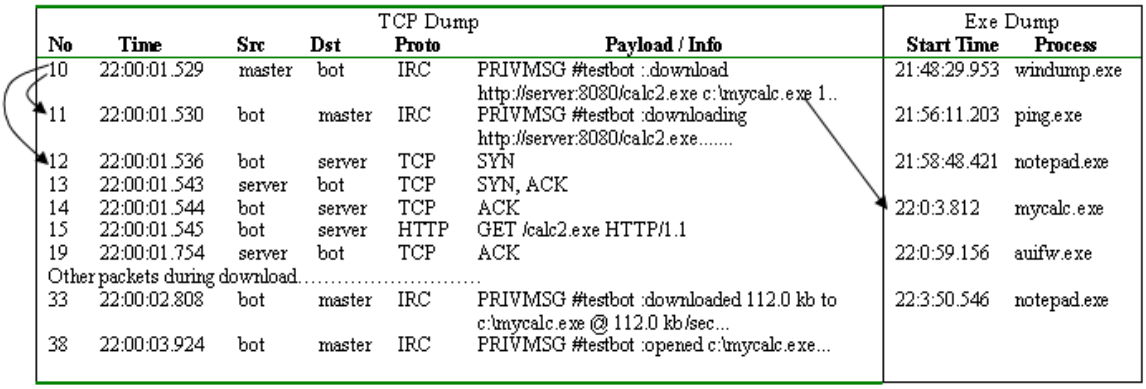

Fig. 3. Multiple log file correlation

TABLE III
PERFORMANCES OF DIFFERENT CLASSIFIERS ON FLOW-LEVEL FEATURES

Dataset	Metric	Boosted-J48	Bayes-Net	Naive-Bayes	J48	SVM
SDBot	ACC%	98.9	99.0	98.9	98.8	97.8
	FP%	1.5	1.3	1.5	1.6	3.0
	FN%	0.0	0.0	0.0	0.0	0.0
RBot	ACC%	98.8	96.4	95.2	96.4	96.4
	FP%	1.5	3.0	3.1	3.2	3.0
	FN%	0.0	4.2	6.5	4.0	4.2

TABLE IV
COMPARING PERFORMANCES BETWEEN OUR METHOD (TEMPORAL) AND THE METHOD OF LIVADAS ET AL. ON THE COMBINED DATASET

Method	Metric	Boosted-J48	Bayes-Net	Naive-Bayes	J48	SVM
Temporal	ACC%	99.9	99.5	99.1	99.2	99.1
Livadas	ACC%	97.0	99.7	97.1	97.5	99.0
Temporal	FP%	0.0	0.0	0.0	0.0	0.0
Livadas	FP%	0.3	0.0	0.0	0.0	0.0
Temporal	FN%	0.2	0.9	1.9	1.7	1.9
Livadas	FN%	6.5	0.6	6.3	5.9	2.1

for each flow using the technique described in Section IV-A. Finally, we apply five-fold cross validation on the data and report the accuracy and false alarm rates. We use the Weka ML toolbox [16] for classification.

A. Performance on different data sets

Table 3 reports the classification accuracies (ACC), false positive rates (FP), and false negative rates (FN) for each of the classifiers for different datasets. The datasets SDBot and RBot correspond to those where the bot-flows are generated only from SDBot and RBot, respectively; and normal flows are generated from uninfected machines. The results are obtained by applying five-fold cross validation on the datasets. Boosted J48 has the best detection accuracy (98.8%) for RBot, whereas Bayes Net has the best detection accuracy (99.0%) for SDBot. However, it is evident that Boosted J48 is less dataset-sensitive since it performs consistently on both datasets, and Bayes Net is only 0.1% better than Boosted J48 for the SDBot dataset. Thus, we conclude that BoostedJ48 has overall better performance than other classifiers. This is also supported by the results presented next.

B. Comparison with other techniques

We also compare our technique with another machine-learning technique applied by Livadas et al. [5]. They extract several flow-based features, such as a histogram of packet sizes, flow duration, bandwidth etc., but these are different from our feature set. They first identify IRC flows and then detect bot flows in the IRC flows. We don't need to identify IRC flows to detect C&C traffic using our analysis, but in order to perform a fair comparison we filter out non-IRC flows. We then extract their optimal set of features from the filtered data, apply five-fold cross validation, and report the accuracy and false alarm rates.

The rows labeled "Temporal" and "Livadas" in Table 4 report the classification accuracies (ACC), false positive rates (FP), and false negative rates (FN) of our technique and the technique of Livadas et al. [5], respectively. The comparison reported is for the combined dataset that consists of bot-flows from both SDBot and RBot infected machines, and all the normal flows from uninfected machines (with non-IRC flows filtered out). We see that Temporal performs consistently across all classifiers having accuracy $> 99\%$, whereas Livadas has $\leq 97.5\%$ accuracy in three classifiers and shows slightly better accuracy (0.2% higher) than Temporal only with Bayes Net. Bayes Net tends to perform well on a feature set if there are dependencies among the features. Since it is likely that there are dependencies among the features used by Livadas, we infer that the overall detection accuracy of Livadas is probably sensitive to classifiers, whereas Temporal is robust to all classifiers. Additionally, Temporal outperforms Livadas in false negative rates for all classifiers except Bayes Net. Finally, we again find that BoostedJ48 has the best performance among all classifiers, so we conclude that our Temporal method with BoostedJ48 has the best overall performance.

Fig. 4 presents the *receiver operating characteristic* (ROC) curves corresponding to the combined dataset results. ROC curves plot true positive rate against false positive rate. An

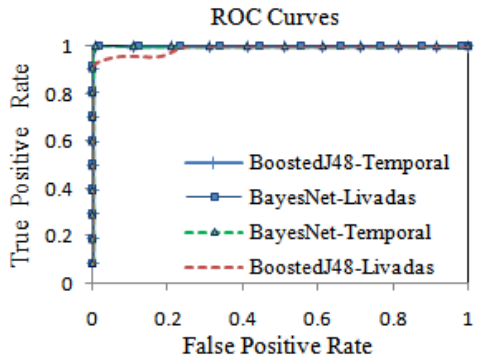

Fig. 4. ROC curves of Bayes Net and Boosted J48 on the combined data

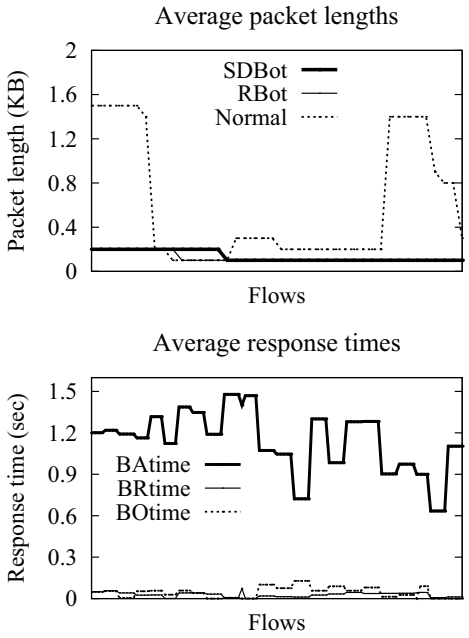

Fig. 5. Flow summary statistics. (above): Average packet lengths of normal and bot-flows, (below): Average *BRtime*, *BOtime*, and *BAtime* of bot-flows

ROC curve is better if the *area under the curve* (AUC) is higher, which indicates a higher probability that an instance will be correctly classified. In this figure, the ROC curve labeled as "Bayes Net-Livadas" corresponds to the ROC curve of Bayes Net on the combined data set for the Livadas et al. technique, and so on. We see that all of the ROC curves are almost co-incidental, except BoostedJ48-Livadas, which is slightly worse than the others. The AUC of "BoostedJ48-Livadas" is 0.993, whereas the AUC of all other curves are greater than or equal to 0.999.

C. Further analysis

Fig. 5 shows statistics of several features. The upper chart plots the average packet length (in KB) of each flow that appears in the dataset. Bot-flows and normal flows are shown as separate series. A data point (X, Y) represents the average packet length Y of all packets in flow X of a particular series (bot-flow or normal). It is clear from the chart that bot-flows have a certain packet length ($\leq 0.2KB$), whereas normal flows have rather random packet lengths. Thus, our assumption about packet lengths is validated by this chart. The lower chart plots three different response times: Bot-response time (*BRtime*), Bot-other time (*BOtime*), and Bot-app time (*BAtime*) for each bot-flow. It is evident that average *BRtime* is less than 0.1 second, average *BOtime* is less than 0.2 seconds and average *BAtime* is between 0.6 and 1.6 seconds. The threshold values for these response times were chosen according to these observations.

VI. Conclusion

We presented the novel idea of correlating multiple log files and applying data mining for detecting botnet C&C traffic. Our idea is to utilize the temporal correlation between two different log files: `tcpdump`, and `exedump`. The `tcpdump` file logs all network packets that are sent/received by a host, whereas the `exedump` file logs the start times of application program executions on the host. We implement a prototype system and evaluate its performance using five different classifiers: support vector machines, decision trees, Bayes Nets, Boosted decision trees, and Naïve Bayes. Comparison with another technique by Livadas et al. [5] for C&C traffic detection shows that our method has overall better performance when used with a Boosted decision tree classifier. The technique used by Livadas et al. first identifies IRC flows and then detects botnet traffic from the IRC flows. Our technique is more general because it does not need to identify IRC traffic and is therefore applicable to non-IRC botnet protocols, as long as certain realistic assumptions about the command-response timing relationships (detailed in Section III) remain valid.

In future work we intend to apply this temporal correlation technique to more system level logs such as those that track process/service executions, memory/CPU utilization, disk reads/writes, and so on. We also would like to implement a real-time C&C traffic detection system using our approach.

References

[1] T. Ferguson, "Botnets threaten the internet as we know it," *ZDNet Australia*, April 2008.

[2] M. Rajab, J. Zarfoss, F. Monrose, and A. Terzis, "A multifaceted approach to understanding the botnet phenomenon," in *Proc. 6th ACM SIGCOMM Conference on Internet Measurement (IMC'06)*, 2006, pp. 41–52.

[3] F. Freiling, T. Holz, and G. Wicherski, "Botnet tracking: Exploring a root-cause methodology to prevent distributed denial-of-service attacks," in *Proc. 10th European Symposium On Research In Computer Security (ESORICS)*, vol. Lecture Notes in Computer Science 3676, September 2005, pp. 319–335.

[4] J. B. Grizzard, V. Sharma, C. Nunnery, B. B. Kang, and D. Dagon, "Peer-to-peer botnets: Overview and case study," in *Proc. 1st Workshop on Hot Topics in Understanding Botnets*, 2007, p. 1.

[5] C. Livadas, B. Walsh, D. Lapsley, and W. Strayer, "Using machine learning techniques to identify botnet traffic," in *Proc. 31st IEEE Conference on Local Computer Networks (LCN'06)*, November 2006, pp. 967–974.

[6] J. Goebel and T. Holz, "Rishi: Identify bot contaminated hosts by irc nickname evaluation," in *Proc. 1st Workshop on Hot Topics in Understanding Botnets*, 2007, p. 8.
[7] P. Barford and V. Yegneswaran, *An Inside Look at Botnets*, ser. Advances in Information Security. Springer, 2006.
[8] D. Dagon, C. Zou, and W. Lee, "Modeling botnet propagation using time zones," in *Proc. 13th Network and Distributed System Security Symposium (NDSS'06)*, 2006.
[9] E. Cooke, F. Jahanian, and D. McPherson, "The zombie roundup: Understanding, detecting, and disrupting botnets," in *Proc. Steps to Reducing Unwanted Traffic on the Internet Workshop (SRUTI'05)*, 2005, p. 6.
[10] A. Karasaridis, B. Rexroad, and D. Hoeflin, "Wide-scale botnet detection and characterization," in *Proc. 1st Workshop on Hot Topics in Understanding Botnets*, 2007, p. 7.
[11] (2006) SDBOT information webpage. [Online]. Available: www.megasecurity.org/trojans/s/sdbot/Sdbot0.5a.html.
[12] (2006) RBOT information webpage. [Online]. Available: http://jarryd.onestop.net/rxbot-howto.html.
[13] (2007) The Unreal IRC Daemon website. [Online]. Available: http://www.unrealircd.com/.
[14] (2007) The Windump website. [Online]. Available: http://www.winpcap.org/windump/.
[15] M. M. Masud, L. Khan, and B. Thuraisingham, "A scalable multi-level feature extraction technique to detect malicious executables," *Information Systems Frontiers*, vol. 10, no. 1, pp. 33–45, March 2008.
[16] (2008) The WEKA Data Mining with Open Source Software website. [Online]. Available: http://www.cs.waikato.ac.nz/ml/weka/.

Chapter 15

DETECTING REMOTE EXPLOITS USING DATA MINING

Mohammad Masud, Latifur Khan, Bhavani Thuraisingham,
Xinran Wang, Peng Liu and Sencun Zhu

Abstract This paper describes the design and implementation of DExtor, a data-mining-based exploit code detector that protects network services. DExtor operates under the assumption that normal traffic to network services contains only data whereas exploits contain code. The system is first trained with real data containing exploit code and normal traffic. Once it is trained, DExtor is deployed between a web service and its gateway or firewall, where it operates at the application layer to detect and block exploit code in real time. Tests using large volumes of normal and attack traffic demonstrate that DExtor can detect almost all the exploit code with negligible false alarm rates.

Keywords: Server attacks, exploit code, data mining, attack detection

1. Introduction

Remote exploits are often used by attackers to gain control of hosts that run vulnerable services or software. Typically, an exploit is sent as an input to a remote vulnerable service to hijack the control flow of machine instruction execution. Attackers sometimes inject executable code in the exploit that is run after a successful hijacking attempt. We refer to such remote code-carrying exploits as "exploit code."

Several approaches have been proposed for analyzing network flows to detect exploit code [1, 4, 8–11]. An attack can be prevented if an exploit is detected and intercepted while it is in transit to a server. This approach is compatible with legacy code and does not require changes to the underlying computing infrastructure. Our solution, DExtor, follows this strategy. In particular, it uses data mining to address the general problem of exploit code detection.

Please use the following format when citing this chapter:

Masud, M., Khan, L., Thuraisingham, B., Wang, X., Liu, P. and Zhu, S., 2008, in IFIP International Federation for Information Processing, Volume 285; *Advances in Digital Forensics IV*; Indrajit Ray, Sujeet Shenoi; (Boston: Springer), pp. 177–189.

A Practical Approach to Classify Evolving Data Streams: Training with Limited Amount of Labeled Data

Mohammad M. Masud[†] mehedy@utdallas.edu
Jing Gao[‡] jinggao3@uiuc.edu
Latifur Khan[†] lkhan@utdallas.edu
Jiawei Han[‡] hanj@cs.uiuc.edu
Bhavani Thuraisingham[†] bhavani.thuraisingham@utdallas.edu

[†]Department of Computer Science, University of Texas at Dallas
[‡]Department of Computer Science, University of Illinois at Urbana-Champaign

Abstract

Recent approaches in classifying evolving data streams are based on supervised learning algorithms, which can be trained with labeled data only. Manual labeling of data is both costly and time consuming. Therefore, in a real streaming environment, where huge volumes of data appear at a high speed, labeled data may be very scarce. Thus, only a limited amount of training data may be available for building the classification models, leading to poorly trained classifiers. We apply a novel technique to overcome this problem by building a classification model from a training set having both unlabeled and a small amount of labeled instances. This model is built as micro-clusters using semi-supervised clustering technique and classification is performed with κ-nearest neighbor algorithm. An ensemble of these models is used to classify the unlabeled data. Empirical evaluation on both synthetic data and real botnet traffic reveals that our approach, using only a small amount of labeled data for training, outperforms state-of-the-art stream classification algorithms that use twenty times more labeled data than our approach.

1 Introduction

Stream data classification is a challenging problem because of two important properties: its infinite length and evolving nature. Data streams may evolve in several ways: the prior probability distribution $p(c)$ of a class c may change, or the posterior probability distribution $p(c|x)$ of the class may change, or both the prior and posterior probabilities may change. In either case, the challenge is to build a classification model that is consistent with the current concept. Traditional learning algorithms that require several passes on the training data, cannot be directly applied to the streaming environment, because the number of training examples would be infinite. To solve this problem, ensemble classification techniques have been proposed.

Ensemble approaches have the advantage that they can be updated efficiently, and they can be easily made to adopt the changes in the stream. Several ensemble approaches have been devised for classification of evolving data streams [7, 10]. The general technique practiced by these approaches is that the data stream is divided into equal-sized chunks. Each of these chunks is used to train a classifier. An ensemble of L such classifiers are used to test unlabeled data. However, these ensemble approaches are based on supervised learning algorithms, and can be trained only with labeled data. But in practice, labeled data in streaming environment are rare.

Manual labeling of data is usually costly and time consuming. So, in an streaming environment, where data appear at a high speed, it may not be possible to manually label all the data as soon as they arrive. Thus, in practice, only a small fraction of each data chunk is likely to be labeled, leaving a major portion of the chunk as unlabeled. So, a very limited amount of training data will be available for the supervised learning algorithms. Considering this difficulty, we propose an algorithm that can handle "partially labeled" training data in a streaming environment. By "partially labeled" we mean only a fraction (e.g. 5%) of the training instances are labeled, and by "completely labeled" we mean all (100%) the training instances are labeled. Our approach is capable of producing the same (or even better) results with *partially labeled* training data compared to other approaches that use *completely labeled* training data having twenty times more labeled data than our approach.

Naturally, stream data could be stored in buffer and pro-

cessed when the buffer is full, so we divide the stream data into equal sized chunks. We train a classification model from each chunk. We propose a semi-supervised clustering algorithm to create K clusters from the partially labeled training data. A summary of the statistics of the instances belonging to each cluster is saved as a "micro-cluster". These micro-clusters serve as a classification model. To classify a test instance using this model, we apply the κ-nearest neighbor (κ-NN) algorithm to find the Q nearest micro-clusters from the instance and select the class that has the highest frequency of labeled data in these Q clusters. In order to cope with the stream evolution, we keep an ensemble of L such models. Whenever a new model is built from a new data chunk, we update the ensemble by choosing the best L models from the L+1 models (previous L models and the new model), based on their individual accuracies on the labeled training data of the new data chunk. Besides, we refine the existing models in the ensemble whenever a new class of data evolves in the stream.

It should be noted that when a new data point appears in the stream, it may not be labeled immediately. We defer the ensemble updating process until some data points in the latest data chunk have been labeled, but we keep classifying new unlabeled data using the current ensemble. For example, consider the online credit-card fraud detection problem. When a new credit-card transaction takes place, its class ($\{$fraud,authentic$\}$) is predicted using the current ensemble. Suppose a 'fraud' transaction has been mis-classified as 'authentic'. When the customer receives the bank statement, he will identify this error and report to the authority. In this way, the actual labels of the data points will be obtained, and the ensemble will be updated accordingly.

We have several contributions. First, we propose an efficient semi-supervised clustering algorithm based on cluster-impurity measure. Second, we apply our technique to classify evolving data streams. To our knowledge, there are no stream data classification algorithms that apply semi-supervised clustering. Third, we provide a solution to the more practical situation of stream classification when labeled data are scarce. We show that our approach can achieve better classification accuracy than other stream classification approaches, utilizing only a fraction (e.g. 5%) of the labeled instances used in those approaches. Finally, we apply our technique to detect botnet traffic, and obtain 98% classification accuracy on average. We believe that the proposed method provides a promising, powerful, and practical technique to the stream classification problem in general.

The rest of the paper is organized as follows: section 2 discusses related work, section 3 describes the semi-supervised clustering technique, section 4 discusses the ensemble classification with micro-clusters, section 5 discusses the experiments and evaluation of our approach, and section 6 concludes with directions to future works.

2 Related work

Our work is related to both stream classification and semi-supervised clustering techniques. We briefly discuss both of them.

Semi-supervised clustering techniques utilize a small amount of knowledge available in the form of pairwise constraints (*must-link, cannot-link*), or class labels of the data points. Recent approaches for semi-supervised clustering incorporated pairwise constraints on top of the unsupervised K-means clustering algorithm and formulated a constraint-based K-means clustering problem [2, 9], which was solved with an Expectation-Maximization (E-M) framework. Our approach is different from these approaches because rather than using pair-wise constraints, we utilize a cluster-impurity measure based on the limited labeled data contained in each cluster. If pair-wise constraints are used, then the running time per E-M step is quadratic in total number of labeled points, whereas the running time is linear if impurity measures are used. So, the impurity measures are more realistic in classifying a high-speed stream data.

There have been many works in stream data classification. There are two main approaches - single model classification, and ensemble classification. Single model classification techniques incrementally update their model with new data to cope with the evolution of the stream [4, 5]. These techniques usually require complex operations to modify the internal structure of the model and may perform poorly if there is concept-drift in the stream. To solve these problems, several ensemble techniques for stream data mining have been proposed [7, 10]. These ensemble approaches have the advantage that they can be more efficiently built than updating a single model and they observe higher accuracy than their single model counterpart [8].

Our approach is also an ensemble approach, but it is different from other ensemble approaches in two aspects. First, previous ensemble-based techniques use the underlying learning algorithm (such as decision tree, Naive Bayes, etc.) as a black-box and concentrate only on building an efficient ensemble. But we concentrate on the learning algorithm itself, and try to construct efficient classification models in an evolving scenario. In this light, our work is more closely related with the work of Aggarwal et al [1]. Secondly, previous techniques (including [1] require *completely labeled* training data. But in practice, a very limited amount of labeled data may be available in the stream, leading to poorly trained classification models. So our approach is more realistic in a stream environment. Our model is also different from Aggarwal et al. [1] in one more aspect: Aggarwal et al. apply horizon-fitting to classify evolving data streams, whereas we use a fixed-sized ensemble of classifiers, which requires less memory since we do not need to store snapshots.

3 Impurity-based clustering with small number of labeled data

In the semi-supervised clustering problem, we are given m data points $\mathcal{X} = \{\boldsymbol{x_1}, \boldsymbol{x_2}, ..., \boldsymbol{x_m}\}$, and their corresponding class labels, $\mathcal{Y} = \{\boldsymbol{y_1}, \boldsymbol{y_2}, ..., \boldsymbol{y_m}\}$, $y_j \in \{\phi, 1, ..., C\}$ where C is the total number of classes. If a data point $x_j \in \mathcal{X}$ has $y_j = \phi$, then it is unlabeled. We are to create K clusters, maintaining the constraint that all *labeled* points in the same cluster have the same class label. Given a limited amount of labeled data, the goal of impurity-based clustering is to create K clusters by minimizing the intra-cluster dispersion (same as unsupervised K-means) and at the same time minimizing the impurity of each cluster. We will refer to this problem as K-means with Minimization of Cluster Impurity (MCI-Kmeans). A cluster is completely pure if it contains labeled data points from only one class (along with some unlabeled data). Thus, the objective function should penalize each cluster for being impure. The general form of the objective function is as follows:

$$\mathcal{O}_{MCIKmeans} = \sum_{i=1}^{K} \sum_{\boldsymbol{x} \in \mathcal{X}_i} ||\boldsymbol{x} - \boldsymbol{u_i}||^2 + \sum_{i=1}^{K} \mathcal{W}_i * Imp_i \quad (1)$$

where $\mathcal{W}_i$ is the weight associated with cluster i and Imp_i is the impurity of cluster i. In order to ensure that both the intra-cluster dispersion and cluster impurity are given the same importance, the weight associated with each cluster should be adjusted properly. Besides, we would want to penalize each data point that contributes to the impurity of the cluster (i.e., the labeled points). So, the weight associated with each cluster is chosen to be

$$\mathcal{W}_i = |\mathcal{L}_i| * \bar{D}_{\mathcal{L}_i} \quad (2)$$

where $\mathcal{L}_i$ is the set of all labeled data points in Cluster i and $\bar{D}_{L_i}$ is the average dispersion from each of these labeled points to the cluster centroid. Thus, each labeled point has a contribution to the total penalty, which is equal to the cluster impurity multiplied by the average dispersion of the labeled points from the centroid. We observe that equation (2) is equivalent to the sum of dispersions of all labeled points from the cluster centroid, i.e., $\mathcal{W}_i = \sum_{\boldsymbol{x} \in \mathcal{L}_i} ||\boldsymbol{x} - \boldsymbol{u_i}||^2$. Substituting this value of $\mathcal{W}_i$ in (1) we obtain:

$$\mathcal{O}_{MCIKmeans} = \sum_{i=1}^{K} \sum_{\boldsymbol{x} \in \mathcal{X}_i} ||\boldsymbol{x} - \boldsymbol{u_i}||^2 + \sum_{i=1}^{K} \sum_{\boldsymbol{x} \in \mathcal{L}_i} ||\boldsymbol{x} - \boldsymbol{u_i}||^2 * Imp_i$$

$$= \sum_{i=1}^{K} (\sum_{\boldsymbol{x} \in \mathcal{X}_i} ||\boldsymbol{x} - \boldsymbol{u_i}||^2 + \sum_{\boldsymbol{x} \in \mathcal{L}_i} ||\boldsymbol{x} - \boldsymbol{u_i}||^2 * Imp_i) \quad (3)$$

Impurity measures: Equation (3) should be applicable to any impurity measure in general. We use the following impurity measure: $Imp_i = \mathcal{ADC}_i * Ent_i$, where $\mathcal{ADC}_i$ is the "aggregated dissimilarity count" of cluster i and Ent_i is the entropy of cluster i. In order to understand $\mathcal{ADC}_i$, we first need to define "Dissimilarity count".

Definition 1 (Dissimilarity count) *Dissimilarity count $\mathcal{DC}_i(\boldsymbol{x}, y)$ of a data point $\boldsymbol{x}$ in cluster i having class label y is the total number of labeled points in that cluster belonging to classes other than y. If $\boldsymbol{x}$ is unlabeled (i.e., $y = \phi$), then $\mathcal{DC}_i(\boldsymbol{x}, y)$ is zero.*

In other words, $\mathcal{DC}_i(\boldsymbol{x}, y) = 0$, if $\boldsymbol{x}$ is unlabeled, and $\mathcal{DC}_i(\boldsymbol{x}, y) = |\mathcal{L}_i| - |\mathcal{L}_i(c)|$, if $\boldsymbol{x}$ is labeled and its label $y = c$, where $\mathcal{L}_i(c)$ is the set of labeled points in cluster i belonging to class c. Note that $\mathcal{DC}_i(\boldsymbol{x}, y)$ can be computed in constant time, if we keep an integer vector to store the counts $|\mathcal{L}_i(c)|, c \in \{1, .., C\}$. "Aggregated dissimilarity count" or $\mathcal{ADC}_i$ is the sum of the dissimilarity counts of all the points in cluster i: $\mathcal{ADC}_i = \sum_{x \in \mathcal{L}_i} DC_i(\boldsymbol{x}, y)$. Entropy of a cluster i is computed as: $Ent_i = \sum_{c=1}^{C} (-p_c^i * log(p_c^i))$, where p_c^i is the prior probability of class c, i.e., $p_c^i = \frac{|\mathcal{L}_i(c)|}{|\mathcal{L}_i|}$.

The use of Ent_i in the objective function ensures that clusters with higher entropy get higher penalties. However, if only Ent_i had been used as the impurity measure, then each labeled point in the same cluster would have received the same penalty. But we would like to favor the labeled points belonging to the majority class in a cluster, and disfavor the points belonging to the minority classes. Doing so would force more labeled points of the majority class to be moved into the cluster, and more labeled points of the minority classes to be moved out of the cluster, making the clusters purer. This is ensured by introducing $\mathcal{ADC}_i$ to the equation. We call the combination of $\mathcal{ADC}_i$ and Ent_i as "compound impurity measure" since it can be shown that $\mathcal{ADC}_i$ is proportional to the "gini index" of cluster i:

$$\mathcal{ADC}_i = \sum_{c=1}^{C} (|\mathcal{L}_i(c)|)(|\mathcal{L}_i| - |\mathcal{L}_i(c)|) = (|\mathcal{L}_i|)^2 \sum_{c=1}^{C} (p_c^i)(1 - p_c^i)$$

$$= (|\mathcal{L}_i|)^2 (1 - \sum_{c=1}^{C} (p_c^i)^2) = (|\mathcal{L}_i|)^2 * Gini_i$$

where $Gini_i$ is the gini index of cluster i.

The problem of minimizing equation (3) is an *incomplete-data problem* because the cluster labels and the centroids are all unknown. The common solution to this problem is to apply E-M [3]. The E-M algorithm consists of three basic steps: initialization, E-step and M-step. The technical details of these steps can be found in [6].

4 Micro-clustering and ensemble training

After creating K clusters using the semi-supervised algorithm, we extract and save summary of the statistics of the data points in each cluster as a "micro-cluster" and discard the raw data points. We will refer to the K micro-clusters built from a data chunk as a classification model, since we use these micro-clusters to classify unlabeled data. We keep an ensemble of L such models.

4.1 Storing the cluster summary information as micro-clusters

Each model $M^i \in M$ contains K micro-clusters $\{M_1^i, ..., M_K^i\}$, where each micro-cluster M_j^i is a summary of

the statistics of the data points $\mathcal{X}_j^i = \{\boldsymbol{x}_{j_1}^i, ...\boldsymbol{x}_{j_N}^i\}$ belonging to that cluster. The summary contains the following statistics: i) N: the total number of points; ii) Lt: the total number of labeled points; iii) $\{Lp[c]\}_{c=1}^{C}$: a vector containing the total number of labeled points belonging to each class. iv) $\boldsymbol{u}$: the centroid of the cluster; v) $\{Sum[r]\}_{r=1}^{d}$: a vector containing the sum of each dimension of the data points in the cluster, where $Sum[r]$ contains the sum of the values of the r^{th} dimension. This vector is required to re-compute the cluster centroid after merging two micro-clusters.

4.2 Updating the ensemble

Every time a new data chunk D_n appears, we train a new model M^n from D_n and update the ensemble by choosing the best L models from the existing L+1 models ($M \cup \{M^n\}$). Algorithm 1 sketches this updating process.

Algorithm 1 Ensemble-Update

Input: $\mathcal{X}^n, \mathcal{Y}^n$: training data points and class labels associated with some of these points in chunk D_n
$\mathcal{Z}^n$: test data points in chunk D_n
K: number of clusters to be created
M: current ensemble of L models $\{M^1, ..., M^L\}$

Output: Updated ensemble M

1: Obtain K clusters $\{\mathcal{X}_1^n, ..., \mathcal{X}_K^n\}$ using E-M algorithm. and compute their summary of statistics $\{M_1^n, ..., M_K^n\}$
2: **if** no cluster $M_j^i \in M$ contains some class c that is seen in the new model M^n **then**
3: Refine-Ensemble(M, M^n)
4: **end if**
5: Test each model $M_j^i \in M$ and M^n on the labeled data of $\mathcal{X}^n$ and obtain its accuracy
6: $M \leftarrow$ Best L models in $M \cup \{M^n\}$ based on accuracy.
7: Predict the class labels of data points in $\mathcal{Z}^n$ with M.

Description of the algorithm "Ensemble-Update": Assuming that the new data chunk D_n has *some* labeled data, we first randomly divide it into two subsets; $\mathcal{X}^n$: the training set and $\mathcal{Z}^n$: the test set. We include all the labeled instances and a few unlabeled instances from D_n in the training set. The rest of the unlabeled instances in D_n are included in the test set. We create K clusters using $\mathcal{X}^n$ with the clustering technique described in section 3. We then extract the summary of statistics from each cluster $\mathcal{X}_j^n$ and store it as a micro-cluster M_j^n (line 1). We handle a special case in lines (2-4) that deals with the evolving data streams. It is possible that in the new data chunk, suddenly a new class has appeared that never appeared in the stream before. Or it may happen that a class has appeared, which has not been in the stream for a long time. In either case, the class is unknown to the existing ensemble of models M. So, we refine the models in M so that they can correctly classify the instances belonging to that class. This refinement process will be explained shortly. Since we have L+1 models now, one of them must be discarded. This is done by testing the accuracy of each of these models on the labeled data points in the training data $\mathcal{X}^n$, and removing the worst of them (lines 5-6). Finally, we predict the classes of the test data $\mathcal{Z}^n$ with the new ensemble M (line 7).

Ensemble refinement: The ensemble M is refined using the newly built model M^n. The refinement procedure first looks into each micro-cluster M_j^n of the model M^n. If any micro-cluster has some labeled data and majority of the labeled data are in class $\hat{c}$, but no model in the ensemble M has any micro-cluster containing labeled data of class $\hat{c}$, then we do the following: for each model $M^i \in M$, we inject the micro-cluster M_j^n in M^i with some probability, called the *probability of injection*, or ρ. To inject a micro-cluster, we first merge two nearest micro-clusters in M^i having the same majority class. Then we add the new micro-cluster M_j^n to M^i. This ensures that total number of micro-clusters in the model remains constant.

The reasoning behind this refinement is as follows. Since no model in ensemble M has knowledge of the class $\hat{c}$, the models will certainly miss-classify any data belonging to the class. By injecting micro-clusters of the class $\hat{c}$, we introduce some data from this class into the models, which reduces their miss-classification rate. It is obvious that for higher values of ρ, more training instances will be provided to a model, which will probably induce more error reduction. So, when $\rho = 1$, we will probably have maximum reduction in prediction error for a *single model*. However, if the same set of micro-clusters are injected in all the models, then the correlation among them may increase, resulting in reduced prediction accuracy of the *ensemble* [8]. Lemma 1 states that the ensemble error is the lowest when $\rho = 0$.

Lemma 1 *Let $\mathcal{E}_M$ be the added error of the ensemble M when $\rho \geq 0$ and $\mathcal{E}_M^0$ is the added error of the ensemble M when $\rho = 0$. Then $\mathcal{E}_M \geq \mathcal{E}_M^0$ for any $\rho \geq 0$.*

Proof: See [6]. □

So, in summary, if we increase ρ, single model error decreases but the ensemble error increases. So, the net effect is that when ρ is initially increased from zero, the overall error keeps decreasing upto a certain point. After that point, increasing ρ hurts performance (i.e., the total error starts increasing) due to increased correlation among the models. This trade-off is also discussed in our experimental results (section 5.3). So, we have to choose a value of ρ that can minimize the overall error. In our experiments, the best value was found to be within 0.5-0.75.

4.3 Ensemble classification using κ-nearest neighbor

In order to classify an unlabeled data point $\boldsymbol{x}$ with a model M^i, we perform the following steps: i) find the Q-nearest *labeled* micro-clusters from $\boldsymbol{x}$ in M^i, by computing the distance between the point and the centroids of the micro-clusters. A micro-cluster is assumed to be *labeled* if

it has at least one labeled data point. ii) select the class with the highest "cumulative normalized-frequency (CNFrq)" in these Q clusters as the predicted class of $\boldsymbol{x}$. The "normalized frequency" of a class c in a micro-cluster is the number of instances of class c divided by the total number of labeled instances in that micro-cluster. CNFrq of a class c is the sum of the normalized frequencies of class c in all the Q clusters. In order to classify $\boldsymbol{x}$ with the *ensemble* M, we perform the following steps: i) find the Q-nearest *labeled* micro-clusters from $\boldsymbol{x}$ in each model $M^i \in M$, ii) select the class with the highest CNFrq in these $L * Q$ clusters as the predicted class.

5 Experiments

In this section we discuss the data sets used in the experiments, the system setup, the results, and analysis.

5.1 Datasets and experimental setup

We apply our technique on real botnet dataset generated in a controlled environment, and also on synthetic datasets. Details of these datasets are discussed in [6].

Each dataset is divided into two equal subsets: training and testing, such that every training instance is followed by a test instance. Our algorithm will be mentioned henceforth as "SmSCluster", which is the acronym for Semi-supervised Stream Clustering. Parameter settings of SmSCluster are as follows, unless mentioned otherwise: K (number of micro-clusters) = 50; Q (number of nearest neighbors for the κ-NN classification) = 1; ρ (probability of injection) = 0.75; Chunk-size = 1,600 records for botnet dataset, and 1,000 records for synthetic dataset; L (ensemble size) = 8; P (Percentage of labeled points): 5% in all datasets, meaning only 5% (randomly selected) of the training data are assumed to have labels;

We compare our algorithm with that of Aggarwal et al [1]. We will refer to this approach as "On Demand Stream". For the On Demand Stream, we use all the default values of its parameters. We use the same set of training and test data for both On Demand Stream and SmSCluster with the only difference that in SmSCluster, only 5% data in the training set have labels, but in On Demand Stream, 100% data in the training set have labels. So, if there are *100 data points in a training set, then On Demand Stream has 100 labeled training data points, but SmSCluster has only 5 of them labeled and 95 of them unlabeled.* Also, for a fair comparison, the chunk-size of SmSCluster is made equal to the buffer size of On Demand Stream. We run our own implementation of the On Demand Stream and report the results.

5.2 Comparison with baseline methods

Figure 1(a) shows the accuracy of SmSCluster and On Demand Stream (for "stream_speed"=80, "buffer_size"=1,600, and k_{fit}=80) on the botnet data. We also obtain similar results for other values of "stream_speed" and "buffer_size" for On Demand Stream. The X-axis represents stream in time units and the Y-axis represents accuracy. Here each time unit is equal to 80 data points. For example, the left bar at time unit 120 (X=120) shows the accuracy of of SmSCluster at that time, which is 98%. The right bar at the same time unit shows the accuracy of On Demand Stream, which is 94%. SmSCluster has 4% or better accuracy than On Demand Stream in all the five time-stamps shown in the chart. Figure 1(b) shows

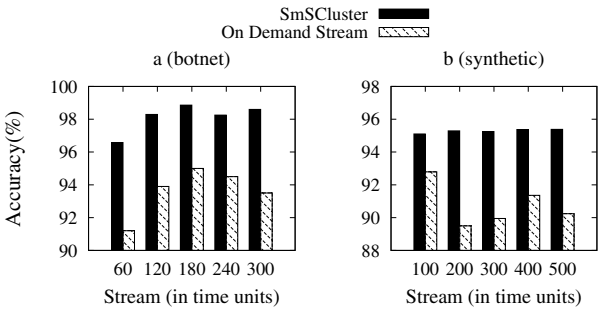

Figure 1. Accuracy comparison on (a) botnet data, and (b) synthetic data

the accuracies of SmSCluster and On Demand Stream (for "stream_speed"=200, "buffer_size"=1,000 , and k_{fit}=50) on synthetic data (100K, C10, D40). We also obtain similar results for other values of "stream_speed" and "buffer_size" for On Demand Stream. SmSCluster has 4% or better accuracy than On Demand Stream in all time units except at time 100, when the difference is 2.3%.

From the above results, we can conclude that SmSCluster outperforms On Demand Stream in all datasets. There are two main reasons behind this. First, SmSCluster considers both the dispersion and impurity measures in building clusters, but On Demand Stream considers only purity, since it applies supervised K-means algorithm. Besides, SmSCluster uses proportionate initialization, so that more clusters are formed for the larger classes (i.e., classes having more instances). But On Demand Stream builds equal number of clusters for each class, so clusters belonging to larger classes may be bigger (and more sparse). Thus, the clusters of SmSCluster are likely to be more compact than those of the On Demand Stream. As a result, the κ-nearest neighbor classification gives better prediction accuracy in SmSCluster. Second, SmSCluster applies ensemble classification, rather than the "horizon fitting" technique used in On Demand Stream. Horizon fitting selects a horizon of training data from the stream that corresponds to a variable-length window of the most recent (contiguous) data chunks. It may be possible that one or more chunks in that window have been outdated, resulting in a less accurate classification model. This is because the set of training data that is the best representative of the current concept are not necessarily contiguous. But SmSCluster always keeps the best

training data (or models) that are not necessarily contiguous. So, the ensemble approach is more flexible in retaining the most up-to-date set of training data, resulting in a more accurate classification model.

5.3 Running times and sensitivity to parameters

The processing speed of SmSCluster for botnet data is 4,000 instances per second, and for synthetic data (300K, C10, D20) 2,500 instances per second, including training and testing instances. Speed is faster for the botnet data since it has only 2 classes, as opposed to 10 classes for the synthetic data. Besides, experimental results show that [6] running time of SmSCluster scales linearly to higher dimensionality and class labels.

All the following results are obtained from the synthetic data (300K, C10, D20), but these are the general trends in any dataset. Figure 2(a) shows how the classification accuracy varies for SmSCluster with the number of micro-clusters (K), and the number of nearest neighbors (Q) for the κ-NN algorithm. We observe that higher values of K lead to better classification accuracies. This may happen because when K is larger, smaller and more compact clusters are formed, leading to a finer-grained classification model for the κ-NN algorithm. However, there is no significant improvement after K=50. We also observe the effect of Q from this chart. It is evident that Q=1 has the highest accuracy, meaning, we need to apply only 1-nearest neighbor. This is true for any value of K. Figure 2(b) shows how the classification accuracy varies for SmSCluster with percentage of labeled data (P) in the training set and the number of micro-clusters (K). We see that the accuracy increases with increasing number of labeled data in the training set. This is desirable, because more labeled data means better guidance for clustering, leading to reduced error. However, after a certain point (20%), there is no real improvement. This is because, probably this amount of labeled data is sufficient for the model. The parameters ρ (injection probability) and L (ensemble size) also have effects on accuracy. We observe that increasing ρ (injection probability) up to 0.5 increases the overall accuracy. After that, the accuracy drops in general. This result follows from our analysis discussed in section 4.2. We achieve the highest accuracy for ensemble size (L)=8. Further increasing the ensemble size does not improve the performance. This is possible if the dataset evolves continuously, resulting in some out-dated models in a larger ensemble.

6 Conclusion

We address a more realistic problem of stream mining: training with a limited amount of labeled data. Our technique is a more practical approach to the stream classification problem since it requires a fewer amount of labeled data, saving much time and cost that would be otherwise

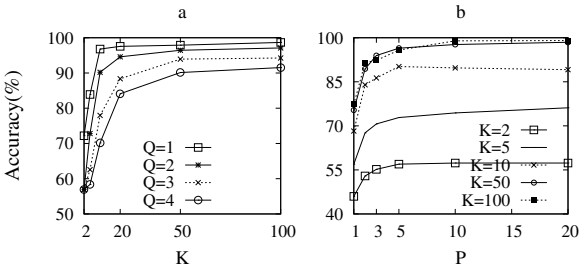

Figure 2. Sensitivity to parameters P, K, Q

required to manually label the data. Previous approaches for stream classification did not address this vital problem. We propose and implement a semi-supervised clustering based stream classification algorithm to solve this limited labeled-data problem. We tested our technique on synthetically generated dataset, and real botnet dataset, and got better classification accuracies than other stream classification techniques. In future, we would like to incorporate feature-weighting and distance-learning in the semi-supervised clustering.

References

[1] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A framework for on-demand classification of evolving data streams. *IEEE Transactions on Knowledge and Data Engineering*, 18(5):577–589, 2006.

[2] S. Basu, M. Bilenko, and R. J. Mooney. A probabilistic framework for semi-supervised clustering. In *Proc. KDD*, 2004.

[3] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society B*, 39:1–38, 1977.

[4] P. Domingos and G. Hulten. Mining high-speed data streams. In *Proc. SIGKDD*, pages 71–80, 2000.

[5] J. Gehrke, V. Ganti, R. Ramakrishnan, and W. Loh. Boat-optimistic decision tree construction. In *Proc. SIGMOD*, 1999.

[6] M. M. Masud, J. Gao, L. Khan, J. Han, and B. Thuraisingham. A practical approach to classify evolving data streams: Training with limited amount of labeled data. *Univ. of Texas at Dallas Tech. Report# UTDCS-32-08 (http://www.utdallas.edu/~mmm058000/reports/UTDCS-32-08.pdf)*, October 2008.

[7] M. Scholz and R. Klinkenberg. An ensemble classifier for drifting concepts. In *Proc. ICML/PKDD Workshop in Knowledge Discovery in Data Streams.*, 2005.

[8] K. Tumer and J. Ghosh. Error correlation and error reduction in ensemble classifiers. *Connection Science*, 8(304):385–403, 1996.

[9] K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl. Constrained k-means clustering with background knowledge. In *Proc. ICML*, pages 577–584, 2001.

[10] H. Wang, W. Fan, P. Yu, and J. Han. Mining concept-drifting data streams using ensemble classifiers. In *Proc. KDD*, 2003.

Inf Syst Front (2008) 10:33–45
DOI 10.1007/s10796-007-9054-3

A scalable multi-level feature extraction technique to detect malicious executables

Mohammad M. Masud · Latifur Khan · Bhavani Thuraisingham

Published online: 23 October 2007

Abstract We present a scalable and multi-level feature extraction technique to detect malicious executables. We propose a novel combination of three different kinds of features at different levels of abstraction. These are binary *n*-grams, assembly instruction sequences, and Dynamic Link Library (DLL) function calls; extracted from binary executables, disassembled executables, and executable headers, respectively. We also propose an efficient and scalable feature extraction technique, and apply this technique on a large corpus of real benign and malicious executables. The above mentioned features are extracted from the corpus data and a classifier is trained, which achieves high accuracy and low false positive rate in detecting malicious executables. Our approach is knowledge-based because of several reasons. First, we apply the knowledge obtained from the binary *n*-gram features to extract assembly instruction sequences using our Assembly Feature Retrieval algorithm. Second, we apply the statistical knowledge obtained during feature extraction to select the best features, and to build a classification model. Our model is compared against other feature-based approaches for malicious code detection, and found to be more efficient in terms of detection accuracy and false alarm rate.

Keywords Disassembly · Feature extraction · Malicious executable · *n*-gram analysis

M. M. Masud
Department of Computer Science,
The University of Texas at Dallas,
2700 Waterview Pkwy, #5116,
Richardson, TX 75080, USA
e-mail: mehedy@utdallas.edu

L. Khan · B. Thuraisingham (✉)
Department of Computer Science,
The University of Texas at Dallas,
Box 830688, EC 31,
Richardson, TX 75083-0688, USA
e-mail: bhavani.thuraisingham@utdallas.edu

L. Khan
e-mail: lkhan@utdallas.edu

1 Introduction

Malicious code is a great threat to computers and computer society. Numerous kinds of malicious codes wander in the wild. Some of them are mobile, such as worms, and spread through internet causing damage to millions of computers worldwide. Other kinds of malicious codes are static, such as viruses, but sometimes deadlier than its mobile counterpart. Malicious code writers usually exploit software vulnerabilities to attack host machines. A number of techniques have been devised by researchers to counter these attacks. Unfortunately, the more successful the researchers become in detecting and preventing the attacks, the more sophisticated malicious code appears in the wild. Thus, the battle between malicious code writers and researchers is virtually never-ending.

One popular technique followed by the anti-virus community to detect malicious code is "signature detection." This technique matches the executables against a unique telltale string or byte pattern called *signature*, which is used as an identifier for a particular malicious code. Although signature detection techniques are being used widely, they are not effective against *zero-day* attacks (new

malicious code), *polymorphic* attacks (different encryptions of the same binary), or *metamorphic* attacks (different code for the same functionality). So, there has been a growing need for fast, automated, and efficient detection techniques that are robust to these attacks. As a result, many automated systems (Newsome et al. 2005; Kolter and Maloof 2004; Golbeck and Hendler 2004; Newman et al. 2002) have been developed.

In this paper we describe our novel *hybrid feature retrieval* (HFR) model that can detect malicious executables efficiently. This is an extension to our previous work (Masud et al. 2007b). It extracts three different kinds of features from the executables at different levels of abstraction and combines them into one feature set, called the *hybrid feature set* (HFS). These features are used to train a classifier (e.g. *support vector machine* (SVM), *decision tree* etc.), which is applied to detect malicious executables. These features are: (a) binary *n*-gram features, (b) derived assembly features (DAFs), and (c) dynamic link library (DLL) call features. Each binary *n*-gram feature is actually a sequence of *n* consecutive bytes in a binary executable, extracted using a technique explained in Section 3.1. Binary *n*-grams reveal the distinguishing byte patterns between the benign and malicious executables. Each DAF is a sequence of assembly instructions in an executable, and corresponds to one binary *n*-gram feature. DAFs reveal the distinctive instruction usage patterns between the benign and malicious executables. They are extracted from the disassembled executables using our *assembly feature retrieval* (AFR) algorithm, explained in Section 4.2. It should be noted that DAF is different from assembly *n*-gram features mentioned in Section 3.2. Assembly *n*-gram features are not used in HFS because of our findings that DAF performs better than them. Each DLL call feature actually corresponds to a DLL function call in an executable, extracted from the executable header as explained in Section 3.3. These features reveal the distinguishing DLL call patterns between the benign and malicious executables. We show empirically that the combination of these three features is always better than any single feature in terms of classification accuracy.

Our work focuses on expanding features at different levels of abstraction, rather than using more features at a single level of abstraction. There are two main reasons behind this. First, the number of features at a given level of abstraction (e.g. binary) is overwhelmingly large. For example, in our larger dataset, we obtain 200 million binary *n*-gram features. Training with this large number of features is way beyond the capabilities of any practical classifier. That is why we limit the number of features at a given level of abstraction to an applicable range. Second, we empirically observe the benefit of adding more levels of abstraction to the combined feature set (i.e., HFS). HFS combines features at three levels of abstraction, namely, binary executables, assembly programs, and system API calls. We show that this combination has higher detection accuracy and lower false alarm rate than the features at any single level of abstraction.

Our technique is related to knowledge-management because of several reasons. First, we apply our knowledge of binary *n*-gram features to obtain DAFs. Second, we apply the knowledge obtained from the feature extraction process to select the best features. This is accomplished by extracting all possible binary *n*-grams from the training data, applying the statistical knowledge corresponding to each *n*-gram (i.e., its frequency in malicious and benign executables) to compute its *information gain* (Mitchell 1997), and selecting the best *S* of them. Finally, we apply another statistical knowledge (presence/absence of a feature in an executable) obtained from the feature-extraction process to train classifiers.

Our research contributions are as follows. First, we propose and implement our HFR model, which combines three kinds of features mentioned above. Second, we apply a novel idea to extract assembly instruction features using binary *n*-gram features, implemented with the AFR algorithm. Third, we propose and implement a scalable solution to the *n*-gram feature extraction and selection problem in general. Our solution works well with limited memory, and significantly reduces running time by applying efficient and powerful data structures and algorithms. Thus, it is scalable to large collection of executables (in the order of thousands), even with limited main memory and processor speed. Finally, we compare our results against Kolter & Maloof's results (Kolter and Maloof 2004), which uses only binary *n*-gram feature, and show that our method achieves better accuracy. We also report the performance/cost tradeoff of our method against Kolter & Maloof's method. It should be pointed out here that our main contribution is an efficient feature extraction technique, not a classification technique. We empirically prove that the combined feature set (i.e., HFS) extracted using our algorithm performs better than other individual feature sets (such as binary *n*-grams) regardless of the classifier (e.g. SVM / decision tree) used.

The rest of the paper is organized as follows: Section 2 discusses related work, Section 3 presents and explains different kinds of *n*-gram feature extraction techniques, Section 4 describes the HFR model, Section 5 discusses our experiments and analyzes results, Section 6 concludes with future research directions.

2 Related work

There have been significant research works in recent years to detect malicious executables. There are two mainstream techniques to automate the detection process: behavioral and content-based. The behavioral approach is primarily applied to detect mobile malicious code. This technique is applied to analyze network traffic characteristics such as source-destination ports/IP addresses, various packet level / flow level statistics, and application level characteristics such as email attachment type, attachment size etc. Examples of behavioral approaches include social network analysis (Golbeck and Hendler 2004; Newman et al. 2002) and statistical analysis (Schultz et al. 2001a). A data mining based behavioral approach for detecting email worms has been proposed by Masud et al. (2007a). Garg et al. (2006) apply feature-extraction technique along with machine learning for *masquerade detection*. They extract features from user behavior in GUI-based systems, such as mouse speed, number of clicks per session and so on. Then the problem is modeled as a binary classification problem, and trained and tested with SVM. Our approach is content-based, rather than behavioral.

The content-based approach analyzes the content of the executable. Some of them try to automatically generate signatures from network packet payloads. Examples are EarlyBird (Singh et al. 2003), Autograph (Kim and Karp 2004), and Polygraph (Newsome et al. 2005). In contrast, our method does not require signature generation or signature matching. Some other content-based techniques extract features from the executables and apply machine learning to detect malicious executables. Examples are Schultz et al. (2001b) and Kolter and Maloof (2004). Schultz et al. (2001b) extract DLL call information using GNU Bin-Utils (Cygnus 1999) and character strings using GNU *strings*, from the header of Windows PE executables. Also, they use byte sequences as features. We also use byte sequences and DLL call information, but we also apply disassembly and use assembly instructions as features. Besides, we extract byte patterns of various lengths (from 2 to 10 bytes), whereas they extract only 2-byte length patterns. A similar work is done by Kolter et al. (Kolter and Maloof 2004). They extract binary *n*-gram features from the binary executables and apply them to different classification methods, and report accuracy. Our model is different from (Kolter and Maloof 2004) in that we extract not only the binary *n*-grams but also assembly instruction sequences from the disassembled executables, and gather DLL call information from the program headers. We compare our model's performance only with (Kolter and Maloof 2004), since they report higher accuracy than (Schultz et al. 2001b).

3 Feature extraction using n-gram analysis

Before going into details of the process, we illustrate a code snippet in Fig. 1 from the Email-Worm "Win32.Ainjo.e," and use it as a running example throughout the paper.

Feature extraction using *n*-gram analysis involves extracting all possible *n*-grams from the given dataset (*training set*), and selecting the best *n*-grams among them. Each such *n*-gram is a feature. We extend the notion of *n*-gram from bytes to assembly instructions, and DLL function calls. That is, an *n*-gram may be either a sequence of *n* bytes, *n* assembly instructions, or *n* DLL function calls, depending on whether we are to extract features from binary executables, assembly programs, or DLL call sequences, respectively. Before extracting *n*-grams, we preprocess the binary executables by converting them to *hexdump* files and *assembly program* files, as explained shortly.

3.1 Binary *n*-gram feature

Here the granularity level is a *byte*. We apply the UNIX 'hexdump' utility to convert the binary executable files into text files, mentioned henceforth as 'hexdump files,' containing the hexadecimal numbers corresponding to each byte of the binary. This process is performed to ensure safe and easy portability of the binary executables. The feature extraction process consists of two phases: (a) feature collection, and (b) feature selection, both of which are explained in the following subsections.

```
------------------------------------------------------------
CODE SNIPPET:-

Program Entry Point = 00472E70
(Email-Worm.Win32.Ainjo.e File Offset:00000400)

 address  opcode                  assembly
--------- ------                  ------------------------
:00455000 FF21                    jmp dword[ecx]
:00455002 089000270014            or byte[eax+14002700], dl
:00455008 00761E                  add byte[esi+1E], dh
:0045500B 45                      inc ebp
:0045500C 00DE                    add dh, bl
:00455010 B4DE                    mov ah, -22

------------------------------------------------------------
DLL FUNCTION CALL INFO FROM THE HEADER

   Module : KERNEL32.DLL

Addr:00073CAE Name: LoadLibraryA
Addr:00073CBC Name: GetProcAddress
Addr:00073CCC Name: ExitProcess
------------------------------------------------------------
```

Fig. 1 Code snippet and DLL call info from the email-worm "Win32.Ainjo.e"

3.1.1 Feature collection

We collect binary *n*-grams from the 'hexdump' files. This is illustrated in example-I.

```
-------------------------------------------------
Example-I

The 4-grams corresponding to the first 6 bytes
sequence (FF2108900027) from the executable in
Figure 1 are the 4-byte sliding windows:
FF210890, 21089000, and 08900027
-------------------------------------------------
```

The basic feature collection process runs as follows. At first, we initialize a list L of *n*-grams to empty. Then we scan each hexdump file by sliding an *n*-byte window. Each such *n*-byte sequence is an *n*-gram. Each *n*-gram g is associated with two values: p_1 and n_1, denoting the total number of positive instances (i.e., malicious executables) and negative instances (i.e., benign executables), respectively, that contain g. If g is not found in L, then g is added to L, and p_1 and n_1 are updated as necessary. If g is already in L, then only p_1 and n_1 are updated. When all hexdump files have been scanned, L contains all the unique *n*-grams in the dataset along with their frequencies in the positive and negative instances. There are several implementation issues related to this basic approach. First, the total number of *n*-grams may be very large. For example, the total number of 10-g in our second dataset (see Section 5.1) is 200 million. It may not possible to store all of them in computer's main memory. To solve this problem, we store the *n*-grams in a disk file F. Second, if L is not sorted, then a linear search is required for each scanned *n*-gram to test whether it is already in L. If N is the total number of *n*-grams in the dataset, then the time for collecting all the *n*-grams would be $O(N^2)$, an impractical amount of time when N=200 million.

In order to solve the second problem, we use a data structure called Adelson Velsky Landis (AVL) tree (GoodRich and Tamassia 2006) to store the *n*-grams in memory. An AVL tree is a height-balanced binary search tree. This tree has a property that the absolute difference between the heights of the left sub-tree and the right sub-tree of any node is at most one. If this property is violated during insertion or deletion, a balancing operation is performed, and the tree regains its height-balanced property. It is guaranteed that insertions and deletions are performed in logarithmic time. So, in order to insert an *n*-gram in memory, we now need only $O(\log_2(N))$ searches. Thus, the total running time is reduced to $O(N\log_2(N))$, making the overall running time about 5 million times faster for N as large as 200 million. Our feature collection algorithm *Extract_Feature* implements these two solutions. It is illustrated in Algorithm 1.

Description of the algorithm: the *for* loop at line 3 runs for each hexdump file in the training set. The inner *while* loop at line 4 gathers all the *n*-grams of a file and adds it to the AVL tree if it is not already there. At line 8, a test is performed to see whether the tree size has exceeded the memory limit (a threshold value). If it exceeds and F is empty, then we save the contents of the tree in F (line 9). If F is not empty, then we merge the contents of the tree with F (line 10). Finally, we delete all the nodes from the tree (line 12).

Time Complexity,T = time (n − gram reading & inserting *in tree*) + time (merging with disk) = $O(B\log_2 K) + O(N)$, where B is the total size of the training data in *bytes*, K is the maximum #of nodes of the tree (i.e., threshold), and N is the total number of *n*-grams collected. Space Complexity: O (K), where K is defined as above.

Algorithm 1 The *n*-gram feature collection algorithm

```
Procedure Extract_Feature (B)
B ={ B1 , B2, … , BK} : all hexdump files
1. T ← empty tree // Initialize AVL-tree
2. F ← new file // Initialize disk file
3. for each Bi ∈ B do
4.     while not EOF(Bi) do  //while not end of file
5.         g ← next_ngram(Bi) // read next n-gram
6.         T.insert(g)  // insert into tree and/or update frequencies as necessary
7.     end while
8.     if T.size > Threshold then //save or merge
9.         if F is empty then  F ←T.inorder() //save tree data in sorted order
10.          else F ← merge(T.inorder(), F) //merge tree data with file data and
11.          save
12.        end if
13.        T ← empty tree //release memory
14.    end if
15. end for
```

3.1.2 Feature selection

If the total number of extracted features is very large, it may not possible to use all of them for training because of several reasons. First, the memory requirement may be impractical. Second, training may be too slow. Third, a classifier may become confused with a large number of features, because most of them would be noisy, redundant or irrelevant. So, we are to choose a small, relevant and useful subset of features. We choose *information gain* (IG) as the selection criterion, because it is one of the best criteria used in literature for selecting the best features.

IG can be defined as a measure of effectiveness of an attribute (i.e., feature) in classifying a training data (Mitchell 1997). If we split the training data based on the values of this attribute, then IG gives the measurement of the expected reduction in entropy after the split. The more an attribute can reduce entropy in the training data, the

better the attribute is in classifying the data. IG of an attribute A on a collection of instances I is given by Eq. 1:

$$\text{Gain}(I, A) \equiv \text{Entropy}(I) - \sum_{V \in \text{values}(A)} \frac{p_v + n_v}{p + n} \text{Entropy}(I_v) \quad (1)$$

Where

values (A) is the set of all possible values for attribute A

I_v is the subset of I where all instances have the value of $A = v$

p is the total number of positive instances in I

n is the total number of negative instances in I

p_v is the total number of positive instances in I_v,

n_v is the total number of negative instances in I_v

In our case, each attribute has only two possible values, i.e., $v \in \{0, 1\}$. If an attribute A (i.e. an n-gram) is present in an instance X, then X_A=1, otherwise it is 0. Entropy of I is computed using the following Eq. 2:

$$\text{Entropy}(I) = -\frac{p}{p+n}\log_2\left(\frac{p}{p+n}\right) - \frac{n}{p+n}\log_2\left(\frac{n}{p+n}\right) \quad (2)$$

Where I, p, and n are as defined above. Substituting Eq. 2 in Eq. 1 and letting $t = n + p$ we get,

$$\text{Gain}(I, A) \equiv -\frac{p}{t}\log_2\left(\frac{p}{t}\right) - \frac{n}{t}\log_2\left(\frac{n}{t}\right) - \sum_{v \in \{0,1\}} \frac{t_v}{t}\left(-\frac{p_v}{t_v}\log_2\left(\frac{p_v}{t_v}\right) - \frac{n_v}{t_v}\log_2\left(\frac{n_v}{t_v}\right)\right) \quad (3)$$

Now, the next problem is to select the best S features (i.e., n-grams) according to IG. One naïve approach is to sort the n-grams in non-increasing order of IG and selecting the top S of them, which requires O ($N\log_2 N$) time and O (N) main memory. But this selection can be more efficiently accomplished using a *heap* that requires O ($N\log_2 S$) time and $O(S)$ main memory. For S=500 and N=200 million, this approach is more than 3 times faster and requires 400,000 times less main memory. A heap is a balanced binary tree with the property that the root of any sub-tree contains the minimum (maximum) element in that sub-tree. We use a *min-heap* that always has the minimum value at its root. Algorithm 2 sketches the feature selection algorithm. At first, the heap is initialized to empty. Then the n-grams (along with their frequencies) are read from disk (line 2) and inserted into the heap (line 5) until the heap size becomes S. After the heap size becomes equal to S, we compare the IG of the next n-gram g against the IG of the root. If IG (*root*)=IG (g) then g is discarded (line 6) since *root* has the minimum IG. Otherwise, *root* is replaced with g (line 7). Finally, the heap property is *restored* (line 9). The process terminates when there are no more n-grams in the disk. After termination, we have the S best n-grams in the heap.

Algorithm 2 The n-gram feature selection algorithm

```
Procedure Select_Feature (F, H, p, n)
F: a disk file containing all n-grams
H: empty heap
p: total number of positive examples
n: total number of negative examples
1. while not EOF(F) do
2.   <g, p1, n1> ← next_ngram(F)  //read n-gram with frequency counts
3.   p0 = P-p1, n0 = N- n1 // #of positive and negative examples not containing g
4.   IG ← Gain(p0, n0, p1, n1, p, n) // using equation (3)
5.   if H.size() < S then H.insert(g, IG)
6.      else if IG <= H.root.IG then continue //discard lower gain n-grams
7.      else H.root ← <g, IG> //replace root
8.   end if
9.   H.restore()  //apply restore operation
10. end while
```

The insertion and restoration takes only O ($\log_2(S)$) time. So, the total time required is O ($N\log_2 S$), with only $O(S)$ main memory. We denote the best S binary features selected using IG criterion as the *Binary Feature Set* (BFS).

3.2 Assembly n-gram feature

In this case, the level of granularity is an assembly instruction. First, we disassemble all the binary files using a disassembly tool called *PEDisassem* (Windows P.E. 1998). It is used to disassemble Windows Portable Executable (P.E.) files. Besides generating the assembly instructions with opcode and address information, PEDisassem provides useful information like list of resources (e.g. cursor) used, list of DLL functions called, list of exported functions, and list of strings inside the code block and so on. In order to extract assembly n-gram features, we follow a method similar to the binary n-gram feature extraction. First we collect all possible n-grams, i.e., sequences of n consecutive assembly instructions, and select the best S of them according to IG. We mention henceforth this selected set of features as *Assembly Feature Set* (AFS). We face the same difficulties as in binary n-gram extraction, such as limited memory and slow running time, and solve them in the same way. Example-II illustrates the assembly n-gram features.

```
------------------------------------------------
Example-II

The 2-grams corresponding to the first 4
assembly instructions in Figure 1 are the
two-instruction sliding windows:

jmp dword[ecx]            ;  or byte[eax+14002700], dl
or byte[eax+14002700], dl ;  add byte[esi+1E], dh
add byte[esi+1E], dh      ;  inc ebp
------------------------------------------------
```

We adopt a standard representation of assembly instructions that has the following format: *name.param1.param2*. Name is the instruction name (e.g., *mov*), param1 is the first parameter, and param2 is the second parameter. Again, a parameter may be one of {*register*, *memory*, *constant*}. So, the second instruction above: "*or byte [eax+14002700], dl*" becomes "*or.memory.register*" in our representation.

3.3 DLL function call feature

Here the granularity level is a DLL function call. An *n*-gram of DLL function call is a sequence of *n* DLL function calls (possibly with other instructions in between two successive calls) in an executable. We extract the information about DLL function calls made by a program from the header of the disassembled file. This is illustrated in Fig. 1. In our experiments, we use only 1-grams of DLL calls, since the higher grams have poorer performance. We enumerate all the DLL function names that have been used by each of the benign and malicious executables, and select the best *S* of them using information gain. We will mention this feature set as *DLL-call feature set* (*DFS*).

4 The hybrid feature retrieval model

The HFR Model extracts and combines three different kinds of features, as illustrated in Fig. 2. HFR consists of different phases and components. The feature extraction components have already been discussed in details. Below is a brief description of the model.

4.1 Description of the model

The HFR Model consists of two phases: a training phase and a test phase. The training phase is shown in Fig. 2a, and the test phase is shown in Fig. 2b. In the training phase we extract binary *n*-gram features (BFS) and DLL call features (DFS) using the approaches explained in Sections 3.1 and 3.3, respectively. We then apply AFR algorithm (to be explained shortly) to retrieve the derived assembly features (DAFs) that represent the selected binary *n*-gram features. These three kinds of features are combined into the *hybrid feature set*, or HFS in short. Please note that DAF is different from assembly *n*-gram features (i.e., AFS).

AFS are not used in HFS because of our findings that DAF performs better than them. We compute the binary feature vector corresponding to the HFS using the technique explained in Section 4.3, and train a classifier using SVM, boosted decision tree, and other classification methods. In the test phase, we scan each test instance and compute the feature vector corresponding to the HFS. This vector is tested against the classifier. The classifier outputs the class prediction {benign, malicious} of the test file.

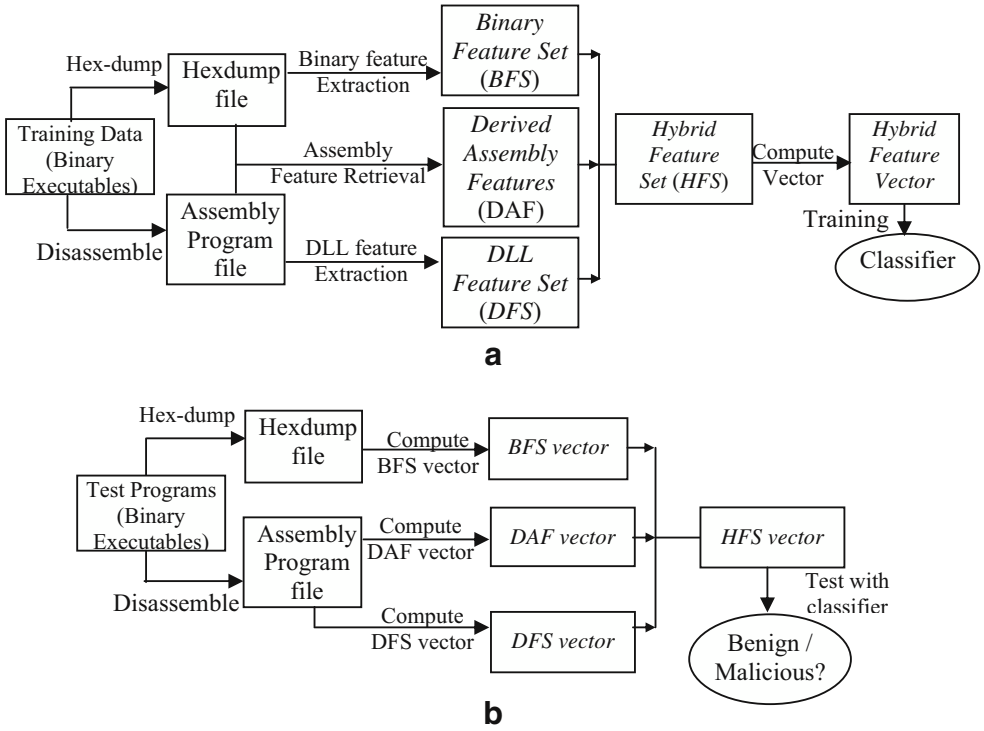

Fig. 2 The hybrid feature retrieval model, **a** training phase, **b** test phase

4.2 The assembly feature retrieval (AFR) algorithm

The AFR algorithm is used to extract assembly instruction sequences (i.e., DAFs) corresponding to the binary *n*-gram features. The main idea is to obtain the complete assembly instruction sequence of a given binary *n*-gram feature. The rationale behind using DAF is as follows. A binary *n*-gram may represent partial information, such as part(s) of one or more assembly instructions or a string inside the code block. We apply AFR algorithm to obtain the complete instruction or instruction sequence (i.e., a DAF) corresponding to the partial one. Thus DAF represents more complete information, which should be more useful in distinguishing the malicious and benign executables. However, binary *n*-grams are still required because they also contain other information like string data, or important bytes at the program header. AFR algorithm consists of several steps. In the first step, a *linear address matching* technique is applied as follows. The offset address of the *n*-gram in the hexdump file is used to find instructions at the same offset at the corresponding assembly program file. Based on the offset value, one of the three situations may occur:

1. The offset is before program entry point, so there is no corresponding assembly code for the *n*-gram. We refer to this address as *address before entry point* (ABEP).
2. There is some data, but no code at that offset. We refer to this address as DATA.
3. There is some code at that offset. We refer to this address as CODE. If this offset is in the middle of an instruction, then we take the whole instruction and consecutive instructions within *n* bytes from the instruction.

In the second step, the best CODE instance is selected among all CODE instances. We apply a heuristic to find the best sequence, called the *most distinguishing instruction sequence* (MDIS) heuristic. According to this heuristic, we choose the instruction sequence that has the highest IG. The AFR algorithm is sketched in Algorithm 3. A comprehensive example of the algorithm is illustrated in Appendix A.

Description of the algorithm: line 1 initializes the lists that would contain the assembly sequences. The *for* loop in line 2 runs for each hexdump file. Each hexdump file is scanned and *n*-grams are extracted (line 4–5). If any of these *n*-grams are in the BFS (line 6–7), then we read the instruction sequence from the corresponding assembly program file at the corresponding address (line 8–10). This sequence is added to the appropriate list (line 12). In this way, we collect all the sequences corresponding to each *n*-gram in the *BFS*. In phase II, we select the best sequence in each *n*-gram list using IG (lines 18–21). Finally, we return the best sequences, i.e., DAFs.

Algorithm 3 The assembly feature retrieval algorithm

```
Procedure Assembly_Feature_Retrival(G, A, B)
 G = {g1, g2,...,gM}: the selected n-gram features (BFS)
 A = {A1, A2, … , AL} : all Assembly files
 B = {B1, B2, … , BL} : all hexdump files
 S = size of BFS
 L = #of training files
Qi: a list containing the possible instruction sequences for gi
//phase I: sequence collection
1. for i = 1 to S do Qi ← empty  end for //initialize sequence lists
2. for each Bi ∈ B do                 //phase I: sequence collection
3.   offset ← 0                       //current offset in file
4.   while not EOF(Bi) do             //read the whole file
5.       g ← next_ngram(Bi)           //read next n-gram
6.       <index, found> ← BinarySearch(G, g) // seach g in G
7.       if found then  // found
8.          q ← an empty sequence
9.          for each instruction r in Ai with address(r)∈ [offset, offset + n]  do
10.              q ← q ∪ r
11.         end for
12.         Qindex ← Qindex ∪ q //add to the sequence
13.      end if
14.      offset = offset + 1
15.  end while
16. end for
17. V ← empty list                    //phase II: sequence selection
18. for i = 1 to S do                 //for each Qi
19.      q← t ∈ {Qi | ∀u∈ Qi IG(t) >= IG(u) //the sequence with the highest IG
20.      V ← V ∪ q
21  end for
22. return V     // DAF sequences
```

Time complexity of this algorithm is $O(nB\log_2 S)$, where B is the total size of training set in *bytes*, S is the total #of selected binary *n*-gram, and n is size of each *n*-gram in *bytes*. Space complexity is $O(SC)$, where S is defined as above and C is the average #of assembly sequences found per binary *n*-gram. The running time and memory requirements of all three algorithms are summarized in Appendix B.

4.3 Feature vector computation and classification

Each feature in a feature set (e.g., HFS, BFS) is a binary feature, meaning, its value is either 1 or 0. If the feature is present in an instance (i.e. an executable), then its value is 1, otherwise its value is 0. For each training (or testing) instance, we compute a feature vector, which is a bit vector consisting of the feature-values of the corresponding feature set. For example, if we want to compute the feature vector V_{BFS} corresponding to BFS of a particular instance I, then for each feature $f \in$ BFS we search f in I. If f is found in I, then we set $V_{\mathrm{BFS}}[f]$ (i.e., the bit corresponding to f) to 1, otherwise, we set it to 0. In this way, we set/reset each bits in the feature vector. These feature vectors are used by the classifiers for training/testing.

We apply SVM, Naïve Bayes (NB), Boosted decision tree, and other classifiers for the classification task. SVM can perform either linear or non-linear classification. The linear classifier proposed by Vladimir Vapnik creates a

hyperplane that separates the data points into two classes with the maximum-margin. A maximum-margin hyperplane is the one that splits the training examples into two subsets, such that the distance between the hyperplane and its closest data point(s) is maximized. A non-linear SVM (Boser et al. 2003) is implemented by applying kernel trick to maximum-margin hyper-planes. The feature space is transformed into a higher dimensional space, where the maximum-margin hyperplane is found. A decision tree contains attribute-tests at each internal node and a decision at each leaf node. It classifies an instance by performing attribute tests from root to a decision node. Decision tree is a rule-based classifier. Meaning, we can obtain human-readable classification rules from the tree. J48 is the implementation of C4.5 Decision Tree algorithm. C4.5 is an extension to the ID3 algorithm invented by Quinlan. A boosting technique called Adaboost combines multiple classifiers by assigning weights to each of them according to their classification performance (Freund and Schapire 1996). The algorithm starts by assigning equal weights to all training samples, and a model is obtained from this training data. Then each misclassified example's weight is increased, and another model is obtained from this new training data. This is iterated for a specified number of times. During classification, each of these models is applied on the test data, and a weighted voting is performed to determine the class of the test instance. We use the AdaBoost.M1 algorithm (Freund and Schapire 1996) on NB, and J48. We only report SVM, and Boosted J48 results because they have the best results. It should be noted that we do not have any preference of any of these two classifiers over the other. We report these accuracies in the results section (Section 5.3).

5 Experiments

We design our experiments to run on two different datasets. Each dataset has different sizes and distributions of benign and malicious executables. We generate all kinds of *n*-gram features (e.g. BFS, AFS, DFS) using the techniques explained in Section 3. Notice that the BFS corresponds to the features extracted by Kolter and Maloof's method (Kolter and Maloof 2004). We also generate the DAF and HFS using our model as explained in Section 4. We test the accuracy of each of the feature sets applying a three-fold cross validation using classifiers such as SVM, decision tree, Naïve Bayes, Bayes Net and Boosted decision tree. Among these classifiers, we obtain the best results with SVM and Boosted decision tree, reported in the results section (Section 5.3). We do not report other classifier results due to space limitations. In addition to this, we compute the average accuracy, false positive and false negative rate, and *receiver operating characteristic* (ROC) graphs (using techniques in Fawcett 2003). We also compare the running time and performance/cost trade-off between HFS and BFS.

5.1 Dataset

We have two non-disjoint datasets. The first dataset (dataset1) contains a collection of 1,435 executables, 597 of which are benign and 838 are malicious. The second dataset (dataset2) contains 2,452 executables, having 1,370 benign and 1,082 malicious executables. So, the distribution of dataset1 is benign=41.6%, malicious=58.4%, and that of dataset2 is benign=55.9%, malicious=44.1%. This distribution was chosen intentionally to evaluate the performance of the feature sets in different scenarios. We collect the benign executables from different Windows XP, and Windows 2000 machines, and collect the malicious executables from (VX-Heavens 2006), which contains a large collection of malicious executables. The benign executables contain various applications found at the Windows installation folder (e.g. "C:\Windows"), as well as other executables in the default program installation directory (e.g., "C:\Program Files"). Malicious executables contain Viruses, Worms, Trojans, and Back-doors. We select only the Win32 Portable Executables (P.E.) in both the cases. We would like to experiment with the ELF executables in future.

5.2 Experimental setup

Our implementation is developed in Java with JDK 1.5. We use the libSVM library (LIBSVM 2006) for running SVM, and Weka ML toolbox (WEKA 2006) for running Boosted decision tree and other classifiers. For SVM, we run C-SVC with a Polynomial kernel; using gamma=0.1, and epsilon=1.0E−12. For Boosted decision tree we run 10 iterations of the AdaBoost algorithm on the C4.5 decision tree algorithm,

Table 1 Classification accuracy (%) of SVM on different feature sets

n	Dataset1			Dataset2		
	HFS	BFS	AFS	HFS	BFS	AFS
1	93.4	63.0	88.4	92.1	59.4	88.6
2	96.8	94.1	88.1	96.3	92.1	87.9
4	96.3	95.6	90.9	97.4	92.8	89.4
6	97.4	95.5	87.2	96.9	93.0	86.7
8	96.9	95.1	87.7	97.2	93.4	85.1
10	97.0	95.7	73.7	97.3	92.8	75.8
Avg	96.30	89.83	86.00	96.20	87.25	85.58
Avg[a]	96.88	95.20	85.52	97.02	92.82	84.98

[a] Average accuracy excluding 1-gram

called J48. We set the parameter S (# of selected features) to 500, since it is the best value found in our experiments. Most of our experiments are run on two machines: a Sun Solaris machine with 4 GB main memory and 2 GHz clock speed, and a LINUX machine with 2 GB main memory and 1.8 GHz clock speed. The reported running times are based on the latter machine. The disassembly and hex-dump are done only once for all machine executables and the resulting files are stored. We then run our experiments on the stored files.

5.3 Results

In this sub-section, we first report and analyze the results obtained by running SVM on the dataset. Later, we show the accuracies of Boosted J48. Since the results from Boosted J48 are almost the same as SVM, we do not report the analyses based on Boosted J48.

Accuracy Table 1 shows the accuracy of SVM on different feature-sets. The columns headed by HFS, BFS, and AFS represent the accuracies of the Hybrid Feature Set (our method), Binary Feature Set (Kolter and Maloof's feature set) and Assembly Feature Set, respectively. Note that the AFS is different from the DAF (i.e., derived assembly features) that has been used in the HFS (see Section 4.1 for details). Table 1 reports that the classification accuracy of HFS is always better than other models, on both datasets. It is interesting to note that the accuracies for 1-gram BFS are very low in both datasets. This is because 1-gram is only a 1-byte long pattern, having only 256 different possibilities. Thus, this pattern is not useful at all in distinguishing the malicious executables from the normal, and may not be used in a practical application. So, we exclude the 1-gram accuracies while computing the average accuracies (i.e., the last row).

Dataset1 Here the best accuracy of the hybrid model is for n=6, which is 97.4, and is the highest among all feature sets. On average, the accuracy of HFS is 1.68% higher than that of BFS, and 11.36% higher than that of AFS. Accuracies of AFS are always the lowest. One possible reason behind this poor performance is that AFS considers only the CODE (see Section 4.2) part of the executables. So, AFS misses any distinguishing pattern carried by the ABEP or DATA parts, and as a result, the extracted features have poorer performance. Moreover, the accuracy of AFS greatly deteriorates for n=10. This is because longer sequences of instructions are rarer in either class of executables (malicious/benign), so these sequences have less distinguishing power. On the other hand, BFS considers all parts of the executable, achieving higher accuracy. Finally, HFS considers DLL calls, as well as BFS and DAF. So, HFS has better performance than BFS.

Dataset2 Here the differences between the accuracies of HFS and BFS are greater than that of dataset1. The average accuracy of HFS is 4.2% higher than that of BFS. Accuracies of AFS are again the lowest. It is interesting to note that HFS has an improved performance over BFS (and AFS) in dataset2. Two important conclusions may be derived from this observation. First, dataset2 is much larger than dataset1, having more diverse set of examples. Here HFS performs better than dataset1, whereas BFS performs worse than dataset1. This implies that HFS is more robust than BFS in a diverse and larger set of instances. Thus, HFS is more applicable than BFS in a large, diverse corpus of executables. Second, dataset2 has more benign executables than malicious, whereas dataset1 has less benign executables. This distribution of dataset2 is more likely in a real world, where benign executables outnumber malicious executables. This implies that HFS is likely to perform better than BFS in a real-world scenario, having larger number of benign executables in the dataset.

Statistical significance test We also perform a pair-wise two-tailed t-test on the HFS and BFS accuracies to test whether the differences between their accuracies are statistically significant. We exclude 1-gram accuracies from this test for the reason explained above. The result of the t-test is summarized in Table 2. The t-value shown in this table is the value of t obtained from the accuracies. There are (5+5−2) *degrees of freedom*, since we have five observations in each group, and there are two groups (i.e., HFS and BFS). *Probability* denotes the probability of rejecting the NULL hypothesis (that there is no difference between HFS and BFS accuracies), while p-value denotes the probability of accepting the NULL hypothesis. For dataset1, the probability is 99.65%, and for dataset2, it is 100.0%. Thus, we conclude that the average accuracy of HFS is significantly higher than that of BFS.

DLL call feature Here we report the accuracies of the DLL function call features (DFS). The 1-gram accuracies are: 92.8% for dataset1 and 91.9% for dataset2. The accuracies for higher grams are less than 75%, so we do not report them. The reason behind this poor performance is possibly that there are no distinguishing call-sequences that can identify the executables as malicious or benign.

Table 2 Pair-wise two-tailed t-test results comparing HFS and BFS accuracies

	DataSet1	DataSet2
t-value	8.9	14.6
Degrees of freedom	8	8
Probability	0.9965	1.00
p-value	0.0035	0.0000

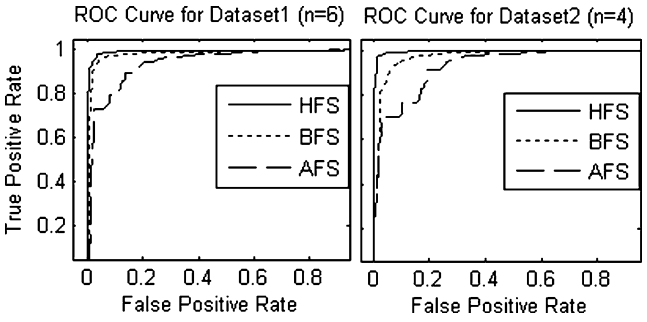

Fig. 3 ROC curves for different feature sets in dataset1 (*left*), and dataset2 (*right*)

ROC curves ROC curves plot the true positive rate against the false positive rates of a classifier. Figure 3 shows ROC curves of dataset1 for n=6 and dataset2 for n=4 based on SVM testing. ROC curves for other values of n have similar trends, except for n=1, where AFS performs better than BFS. It is evident from the curves that HFS is always dominant (i.e. has larger area under the curve) over the other two and it is more dominant in dataset2. Table 3 reports the area under the curve (AUC) for the ROC curves of each of the features sets. A higher value of AUC indicates a higher probability that a classifier will predict correctly. Table 3 shows that the AUC for HFS is the highest, and it improves (relative to other two) in dataset2. This also supports our hypothesis that our model will perform better in a more likely real-world scenario, where benign executables occur more frequently.

False positive and false negative Table 4 reports the false positive and false negative rates (in percentage) for each feature set based on SVM output. The last row reports the average. Again, we exclude the 1-gram values from the average. Here we see that in dataset1, the average false positive rate of HFS is 4.9%, which is the lowest. In dataset2, this rate is even lower (3.2%). False positive rate is a measure of false alarm rate. Thus, our model has the lowest false alarm rate. We also observe that this rate decreases as we increase the number of benign examples. This is because the classifier gets more familiar with benign executables and misclassifies fewer of them as malicious. We believe that a large collection of training set with a larger portion of benign executables would eventually diminish false positive rate towards zero. The false negative rate is also the lowest for HFS as reported in Table 4.

Table 3 Area under the ROC curve on different feature sets

n	Dataset1			Dataset2		
	HFS	BFS	AFS	HFS	BFS	AFS
1	0.9767	0.7023	0.9467	0.9666	0.7250	0.9489
2	0.9883	0.9782	0.9403	0.9919	0.9720	0.9373
4	0.9928	0.9825	0.9651	0.9948	0.9708	0.9515
6	0.9949	0.9831	0.9421	0.9951	0.9733	0.9358
8	0.9946	0.9766	0.9398	0.9956	0.9760	0.9254
10	0.9929	0.9777	0.8663	0.9967	0.9700	0.8736
Avg	0.9900	0.9334	0.9334	0.9901	0.9312	0.9288
Avg[a]	0.9927	0.9796	0.9307	0.9948	0.9724	0.9247

[a] Average value excluding 1-gram

Table 4 False positive and false negative rates on different feature sets

n	Dataset1			Dataset2		
	HFS	BFS	AFS	HFS	BFS	AFS
1	8.0/5.6	77.7/7.9	12.4/11.1	7.5/8.3	65.0/9.8	12.8/9.6
2	5.3/1.7	6.0/5.7	22.8/4.2	3.4/4.1	5.6/10.6	15.1/8.3
4	4.9/2.9	6.4/3.0	16.4/3.8	2.5/2.2	7.4/6.9	12.6/8.1
6	3.5/2.0	5.7/3.7	24.5/4.5	3.2/2.9	6.1/8.1	17.8/7.6
8	4.9/1.9	6.0/4.1	26.3/2.3	3.1/2.3	6.0/7.5	19.9/8.6
10	5.5/1.2	5.2/3.6	43.9/1.7	3.4/1.9	6.3/8.4	30.4/16.4
Avg	5.4/2.6	17.8/4.7	24.4/3.3	3.9/3.6	16.1/8.9	18.1/9.8
Avg[a]	4.9/2.0	5.8/4.1	26.8/1.7	3.2/2.7	6.3/8.1	19.2/17.8

[a] Average value excluding 1-gram

Running Time We compare in Table 5 the running times (feature extraction, training, testing) of different kind of features (HFS, BFS, AFS) for different values of n. Feature extraction time for HFS and AFS includes the disassembly time, which is 465 s (in total) for dataset1, and 865 s (in total) for dataset2. Training time is the sum of feature extraction time, feature-vector computation time, and SVM training time. Testing time is the sum of disassembly time (except BFS) feature-vector computation time, and SVM classification time. Training and testing times based on Boosted J48 have almost similar characteristics, so we do not report them. Table 5 also reports the cost factor as a ratio of time required for HFS relative to BFS. The column *cost factor* shows this comparison. The average feature-extraction times are computed by excluding the 1- and 2-gram, since these grams are unlikely to be used in practical applications. The boldface cells in the table are of particular interest to us. From the table we see that the running times for HFS training and testing on dataset1 are 1.17 and 4.87 times higher than those of BFS, respectively. For dataset2, these numbers are 1.08 and 4.5, respectively. The average throughput for HFS is found to be 0.6 MB/sec (in both datasets), which may be considered as near real-time performance. Finally, we summarize the cost/performance trade-off in Table 6. The column *Performance improvement* reports the accuracy improvement of HFS over BFS. The cost factors are shown in the next two columns. If we drop the disassembly time from testing time (considering that

Table 5 Running times (in seconds)

	n	Dataset1				Dataset2			
		HFS	BFS	AFS	Cost factor[a]	HFS	BFS	AFS	Cost factor[a]
Feature extraction	1	498.41	135.94	553.2	3.67	841.67	166.87	908.42	5.04
	2	751.93	367.46	610.85	2.05	1,157.5	443.99	949.7	2.61
	4	1,582.21	1,189.65	739.51	1.33	3,820.7	3,103.14	1,194.4	1.23
	6	2,267.94	1,877.6	894.26	1.21	8,010.24	7,291.4	1,519.56	1.1
	8	2,971.9	2,572.26	1,035.06	1.16	11,736.	11,011.67	1,189.01	1.07
	10	3,618.31	3,223.21	807.85	1.12	15,594.	14,858.68	2,957	1.05
	Avg[b]	2,610.09	2,215.68	869.17	1.18	9,790.6	9,066.22	1,714.99	1.08
Training	Avg[c]	2,654.68	2,258.86	910.68	**1.18**	9,857.85	9,134.36	1,782.8	**1.08**
Testing	Avg[c]	195.25	40.09	194.9	**4.87**	377.89	83.91	348.35	**4.5**
Testing/MB MB		**1.74**	**0.36**	1.74	4.87	**1.57**	**0.35**	1.45	4.5
Throughput(MB/s))		**0.6**	**2.8**	0.6	–	**0.64**	**2.86**	0.69	–

[a] Ratio of time required for HFS to time required for BFS
[b] Average feature extraction times excluding 1- and 2-gram
[c] Average training/testing times excluding 1- and 2-gram

disassembly is done offline), then the testing cost factor diminishes to 1.0 for both dataset. It is evident from Table 6 that the performance/cost trade-off is better for dataset2 than dataset1. Again, we may infer that our model is likely to perform better in a larger and more realistic dataset. The main bottleneck of our system is disassembly cost. The testing cost factor is higher because here larger proportion of time is used up in disassembly. We believe that this factor may be greatly reduced by optimizing the disassembler, and considering that disassembly can be done offline.

Training & Testing with Boosted J48 We also train and test with this classifier and report the classification accuracies for different features and different values of *n* in Table 7. The second last row (*Avg*) of Table 7 is the average of 2- to 10-g accuracies. Again, for consistency, we exclude 1-gram from the average. We also include the average accuracies of SVM (from last row of Table 1) in the last row of Table 7 for ease of comparison. We would like to point out some important observations regarding this comparison. First, the average accuracies of SVM and Boosted J48 are almost the same, being within 0.4% of each other (for HFS). There is no clear winner between these two classifiers. So, we may use any of these classifiers for our model. Second, accuracies of HFS are again the best among all three. Besides, HFS has 1.84% and 3.6% better accuracies than BFS in dataset1 and dataset2, respectively. This result also justifies our claim that HFS is a better feature set than BFS, irrespective of the classifier used.

Table 6 Performance/cost trade-off between HFS and BFS

	Performance improvement (%) (HFS–BFS) /BFS	Training cost factor (HFS/ BFS)	Testing cost factor (HFS/ BFS)
Dataset1	1.73	1.17	4.87
Dataset2	4.52	1.08	4.5

6 Conclusion

Our HFR model is a novel idea in malicious code detection. It extracts useful features from disassembled executables using the information obtained from binary executables. It then combines the assembly features with other features like DLL function calls and binary *n*-gram features. We have addressed a number of difficult implementation issues and provided efficient, scalable and practical solutions. The difficulties that we face during implementation are related to memory limitations and long running times. By using

Table 7 Classification accuracy (%) of boosted J48 on different feature sets

N	Dataset1			Dataset2		
	HFS	BFS	AFS	HFS	BFS	AFS
1	93.9	64.1	91.3	93.5	58.8	90.2
2	96.4	93.2	89.4	97.1	92.7	85.1
4	96.3	95.4	92.1	97.2	93.6	87.5
6	96.3	95.3	87.8	97.6	93.6	85.4
8	96.7	94.1	89.1	97.6	94.3	83.7
10	96.6	95.1	77.1	97.8	95.1	82.6
Avg[a] (Boosted J48)	96.46	94.62	87.1	97.46	93.86	84.86
Avg[b](SVM)	96.88	95.20	85.52	97.02	92.82	84.98

[a] Average accuracy excluding 1-gram
[b] Average accuracy for SVM (from Table 1)

efficient data structures, algorithms and disk I/O, we are able to implement a fast, scalable and robust system for malicious code detection. We run our experiments on two datasets with different class distribution, and show that a more realistic distribution improves the performance of our model.

Our model also has a few limitations. First, it does not directly handle obfuscated DLL calls or encrypted/packed binaries. There are techniques available for detecting obfuscated DLL calls in the binary (Lakhotia et al. 2005), and to unpack the packed binaries automatically (Royal et al. 2006). We may apply these tools for de-obfuscation/decryption and use their output to our model. Although this is not implemented yet, we look forward to integrate these tools with our model in our future versions. Second, the current implementation is an offline detection mechanism. Meaning, it cannot be directly deployed on a network to detect malicious code. However, it can detect malicious codes in near real time.

We address these issues in our future work, and vow to solve these problems. We also propose several modifications to our model. For example, we would like to combine our features with run-time characteristics of the executables. Besides, we propose building a feature-database that would store all the features and be updated incrementally. This would save a large amount of training time and memory.

Table 8 Assembly code sequence for binary 4-g "00005068"

Sequence #	Op-code	Assembly code	Information gain
1	E8B702**0000**	call 00401818	0.5
	50	push eax	
	6828234000	push 00402328	
2	0FB6800D02**0000**	movzx eax,byte [eax+20]	0.1
	50	push eax	
	68CC000000	push 000000CC	
3	8B805C04**0000**	mov eax, dword [eax+45]	0.2
	50	push eax	
	6801040000	push 00000401	
29	8D801001**0000**	lea eax, dword [eax+110]	0.7
	50	push eax	
	6807504000	push 00405007	
50	25FFFF**0000**	and eax, 0000FFFF	0.3
	50	push eax	
	68E8164100	push 004116E8	
90	25FFFF**0000**	and eax, 0000FFFF	0.4
	50	push eax	
	68600E4100	push 00410E60	

Table 9 Time and space complexities of different algorithms

Algorithm	Time complexity	Space complexity
Feature collection	$O(B\log_2 K)+O(N)$	$O(K)$
Feature selection	$O(N\log_2 S)$	$O(S)$
Assembly feature retrieval	$O(nB\log_2 S)$	$O(SC)$
Total (worst case)	$O(nB\log_2 K)$	$O(SC)$

Acknowledgment The work reported in this paper is supported by AFOSR under contract FA9550-06-1-0045 and by the Texas Enterprise Funds. We thank Dr. Robert Herklotz of AFOSR and Prof. Robert Helms, Dean of the School of Engineering at the University of Texas at Dallas for funding this research.

Appendix A

Here we illustrate an example run of the AFR algorithm. The algorithm scans through each hexdump file, sliding a window of n bytes and checking the n-gram against the binary feature set (BFS). If a match is found, then we collect the corresponding (same offset address) assembly instruction sequence in the assembly program file. In this way, we collect all possible instruction sequences of all the features in BFS. Later, we select the best sequence using information gain. *Example-III*: Table 8 shows an example of the collection of assembly sequences and their IG values corresponding to the n-gram "00005068." Note that this n-gram has 90 occurrences (in all hexdump files). We have shown only 5 of them for brevity. The bolded portion of the op-code in Table 8 represents the n-gram. According to the *Most Distinguishing Instruction Sequence* (MDIS) heuristic, we find that sequence #29 attains the highest information gain, which is selected as the DAF of the n-gram. In this way, we select one DAF per binary n-gram, and return all DAFs.

Appendix B

Here we summarize the time and space complexities of our algorithms in Table 9.

B is the total size of training set in *bytes*, C is the average #of assembly sequences found per binary n-gram, K is the maximum #of nodes of the AVL tree (i.e., threshold), N is the total number of n-grams collected, n is size of each n-gram in *bytes,* and S is the total number of selected n-grams. The worst case assumption: $B > N$ and $SC > K$

References

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (2003). A training algorithm for optimal margin classifiers. In D. Haussler (Ed.), *5th annual ACM workshop on COLT* (pp. 144–152). New York: ACM Press.

Cygnus (1999). *GNU Binutils Cygwin*. Retrieved from http://sourceware.cygnus.com/cygwin.

Fawcett, T. (2003). *ROC Graphs: Notes and practical considerations for researchers*. Tech Report HPL-2003-4, HP Laboratories. Retrieved May 26, 2006, from http://www.hpl.hp.com/personal/TomFawcett/papers/ROC101.pdf.

Freund, Y., & Schapire, R. (1996). Experiments with a new boosting algorithm. In *Proc. of the thirteenth international conference on machine learning* (pp. 148–156). San Mateo, CA: Morgan Kaufmann.

Garg, A., Rahalkar, R., Upadhyaya, S., & Kwiat, K. (2006). Profiling users in GUI based systems for masquerade detection. In *Proc. of the 7th IEEE information assurance workshop (IAWorkshop 2006)* (pp. 48–54).

Golbeck, J., & Hendler, J. (2004). *Reputation network analysis for email filtering*. In CEAS.

GoodRich, M. T., & Tamassia, R. (2006). *Data structures and algorithms in Java* (4th ed.). New York: Wiley.

LIBSVM. (2006). *A library for support vector machine.* Retrieved June 1, 2006 from http://www.csie.ntu.edu.tw/~cjlin/libsvm/.

Kim, H. A., & Karp, B. (2004). Autograph: Toward automated, distributed worm signature detection. In *Proc. of the 13th Usenix security symposium (Security 2004)* (pp. 271–286).

Kolter, J. Z., & Maloof, M. A. (2004). Learning to detect malicious executables in the wild. In *Proc. of the tenth ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 470–478).

Lakhotia, A., Kumar, E. U., & Venable, M. (2005). A method for detecting obfuscated calls in malicious binaries. *IEEE Transactions on Software Engineering, 31*(11), 955–968.

Masud, M. M., Khan, L., & Thuraisingham, B. (2007a). Feature based techniques for auto-detection of novel email worms. In *Proc. of the eleventh Pacific-Asia conference on knowledge discovery and data mining (PAKDD'07)* (pp. 205–216). LNAI 4426/2007.

Masud, M. M., Khan, L., & Thuraisingham, B. (2007b). A hybrid model to detect malicious executables. In *Proc. of the IEEE international conference on communication (ICC'07)* (pp. 1443–1448).

Mitchell, T. (1997). *Machine learning*. New York: McGraw-Hill.

Newman, M. E. J., Forrest, S., & Balthrop, J. (2002). Email networks and the spread of computer viruses. *Physical Review, 66*(3), 035101.

Newsome, J., Karp, B., & Song, D. (2005). Polygraph: Automatically generating signatures for polymorphic worms. In *Proc. of the IEEE symposium on security and privacy* (pp. 226–241).

Royal, P., Halpin, M., Dagon, D., Edmonds, R., & Lee, W. (2006). PolyUnpack: Automating the hidden-code extraction of unpack-executing malware. In *Proc. of 22nd annual computer security applications conference (ACSAC'06)* (pp. 289–300).

Schultz, M., Eskin, E., & Zadok, E. (2001a). MEF Malicious email filter, a UNIX mail filter that detects malicious windows executables. In *Proc. of the USENIX annual technical conference—FREENIX track* (pp. 245–252).

Schultz, M., Eskin, E., Zadok, E., & Stolfo, S. (2001b). Data mining methods for detection of new malicious executables. In *Proc. of the IEEE symposium on security and privacy* (pp. 178–184).

Singh, S., Estan, C., Varghese, G., & Savage, S. (2003). *The earlyBird system for real-time detection of unknown worms*. Technical report—cs2003-0761, UCSD.

VX-Heavens. (2006). Retrieved May 6, 2006 from http://vx.netlux.org/.

WEKA. (2006). Retrieved Aug 1, 2006 from http://www.cs.waikato.ac.nz/ml/weka/.

Windows P.E. Disassembler. (1998). Retrieved June 05, 2006 from http://www.geocities.com/~sangcho/index.html.

Mr. Mohammad Mehedy Masud is a Ph.D. student at the department of Computer Science at the University of Texas at Dallas (UTD) since August, 2005. He received his undergraduate degree in Computer Science and Engineering (CSE) from Bangladesh University of Engineering and Technology (BUET) in 2001. Before joining UTD, he was a lecturer at the department of CSE, BUET, from 2001 to 2004. He is currently an Assistant Professor (on leave) at the same department. He is a member of the IEEE Computer Society, and the Association for Computing Machinery (ACM). His current areas of research are data mining, intrusion detection, and network security. He has already published 15 conference papers and journal articles and more papers are currently under review. He is also an award-winning programmer at the ACM-International Collegiate Programming Contests (ICPC) World Finals-1999, held in Eindhoven, The Netherlands.

Dr. Latifur R. Khan is currently an Associate Professor in the Computer Science department at the University of Texas at Dallas (UTD), where he has taught and conducted research since September 2000. He received his Ph.D. and M.S. degrees in Computer Science from the University of Southern California, in August of 2000, and December of 1996 respectively. He obtained his B.Sc. degree in Computer Science and Engineering from Bangladesh University of Engineering and Technology, Dhaka, Bangladesh in November of 1993. Dr. Khan is the director of the UTD Data Mining/Database Laboratory. Dr. Khan's research areas cover data mining, multimedia information management, and semantic web and database systems. He has served as a committee member in numerous prestigious conferences, symposiums and workshops including the ACM SIGKDD Conference on Knowledge Discovery and Data Mining. Dr. Khan currently serves on the editorial board of North Holland's Computer Standards and Interface Journal, Elsevier Publishing. Dr. Khan has published over 80 papers in prestigious journals and conferences.

Dr. Bhavani Thuraisingham is a Professor of Computer Science and the Director of Cyber Security Research Center at the University of Texas at Dallas (UTD). Prior to joining UTD, she was a program director for 3 years at the National Science Foundation (NSF) in Arlington, VA. She has also worked for the Computer Industry in Mpls, MN for over 5 years and has served as an adjunct professor of computer science and member of the graduate faculty at the University of Minnesota and later taught at Boston University. Dr. Thuraisingham's research interests are in the area of Information Security and data management. She has published over 300 research papers including over 60 journals articles and is the inventor of three patents. She serves on the editorial board of numerous journals including ACM Transactions on Information and Systems Security and IEEE Transactions on Dependable and Secure Computing.

A Novel Technique to Defeat Data Mining-Based Malware Detection Models

Kevin Hamlen Vishwath Mohan Mohammad M Masud
Latifur Khan Bhavani Thuraisingham

University of Texas at Dallas, Richardson, TX 75083

Abstract

We propose a novel obfuscation technique to defeat data mining based malware detection models. Assuming that the model can be extracted from the anti-malware software, we show how patterns can be inserted and removed from the malware so that it can be recognized as a benign executable by the malware detector. Experiments with real malware justifies the effectiveness of our approach.

1 Introduction

Traditional malware detectors are "signature-based". This technique matches the executables against a unique telltale string, or "signature", that can identify the malware. However, these techniques fail when a new malware arrives whose signature is unknown. As a result, data mining techniques for malware detection have been devised [4, 7, 5] that are capable of detecting malware whose signatures are unknown.

A signature-based malware detector can be defeated by a malware if the malware writer can extract the model from the malware detector. For example, assume that the malware detection model uses the signature $\sigma(x)$ to identify malware x. If the writer of x can extract the model, this signature $\sigma(x)$ will be exposed to him. Then the malware can be obfuscated by removing the signature $\sigma(x)$ from it. Now, this obfuscated malware cannot be detected by the malware detector. It has been shown in the past that it is possible to automatically extract models from a malware detector [2].

A similar obfuscation technique can be applied to the data-mining based malware detectors if the detection model can be extracted. However, this obfuscation task would be more challenging, since the data-mining based models are more complex. Rather than searching a signature in the malware, these models search the presence and absence of a set of patterns in the malware. Thus, rather than removing a single signature from the malware, the obfuscation attempt may involve a series of pattern additions and deletions in the

malware. Our current work proposes a solution to this challenging job.

Assuming that we have the data mining model for malware detection, we provide obfuscation techniques that can successfully evade detection. This involves analyzing the structure of the model, understanding its working principle, and modify the malware accordingly. We propose two obfuscation techniques. The first one is "pattern insertion", which involves inserting a pattern into the malware so that it is recognized as a benign executable by the model. The second one is "pattern mutation" that removes some patterns from the malware to evade detection. We have successfully applied one of these techniques for obfuscating a real malware, against a data mining based malware detector.

The rest of the paper is organized as follows. Section 2 discusses related works, section 3 describes the overview of our approach, section 4 describes a data-mining based malware detection model, section 5 describes how to defeat this model, section 6 discusses experiments and evaluation of our technique and section 7 concludes with directions to future works.

2 Related work

3 Overview

Our proposed technique is illustrated in figure 1. It is called "SMalware", which stands for Smart Malware. SMalware consists two main modules. The "model-extraction" module and the "analysis and obfuscation" module. The model-extraction extracts the malware detection model from the anti-virus software.

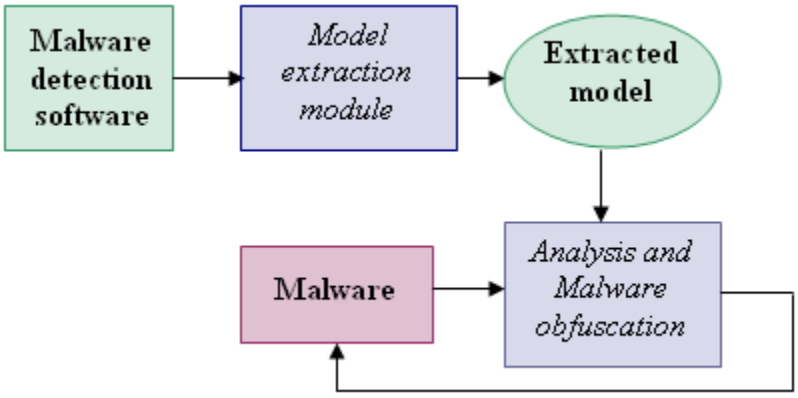

Figure 1: The SMalware life-cycle

Then the analysis and obfuscation module analyzes the extracted model makes necessary obfuscation to the malware code so that the model can no longer detect it. Although in our current implementation the malware code itself

is a separate entity from the two modules, in future we would like to integrate everything into one single executable, so that the malware can automatically update itself to avoid detection.

4 A data mining based malware detection model

A data mining based malware detector first trains itself with known instances of malicious and benign executables. Once trained, it can predict the nature (malicious,benign) of unknown executables by testing them against the model. The basic idea is illustrated in figure 2

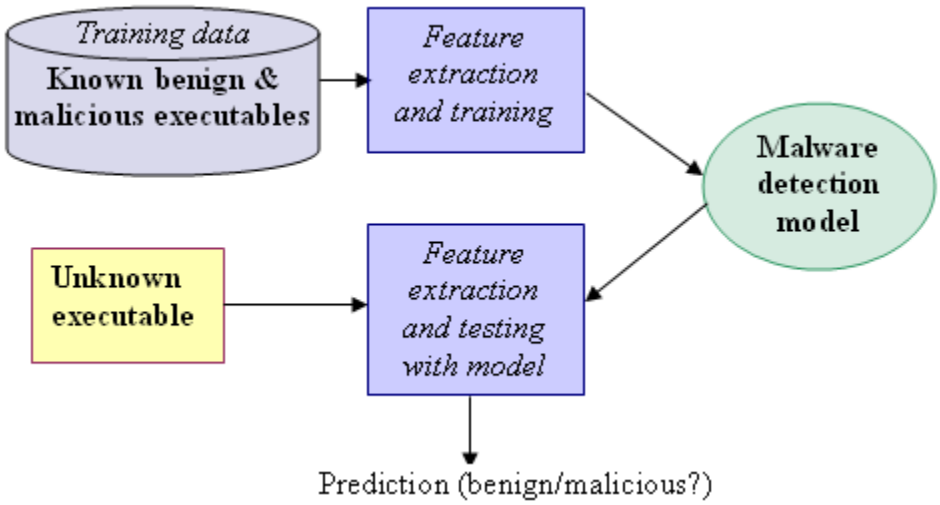

Figure 2: A framework of data mining based malware detector

The correctness of prediction of the model depends on the given training data and the learning algorithm (e.g. support vector machine, decision tree, naive bayes etc.). Several data mining based malware detectors have been proposed in the past [4, 5, 7]. The main advantage of these models over the traditional signature-based models is that the data mining based models are more robust to changes in the malware. Signature based models fail when a malware appears with an unknown signature. In other words, these models only "remembers" the known signatures. Thus, they can detect only those malware that they have seen already. On the other hand, data mining based models try to "generalize" the discriminating features among benign and malicious executables. Thus they are capable of detecting malware that were not known at the time of training. Thus, it is more challenging to defeat a malware detector based on data mining.

Next, we briefly describe one of our previous works [5] in data mining based malware detection. It consists of three main steps:

1. Feature extraction, feature selection, and feature-vector computation from the training data.

2. Training a classification model using the computed feature-vector.

3. Testing executables with the trained model.

4.1 Feature extraction

We extract three different kinds of features from the training instances (i.e., executables). These are as follows:

1. **Binary n-gram features:** In order to extract these features, we consider each executable as a string of bytes, and extract all possible n-grams from the executables. n is varied from 1 to 10.

2. **Assembly n-gram features:** In this case, we disassemble each executable and obtain an assembly language program. Then we extract the n-grams of assembly instructions.

3. **Dynamic link library (DLL) call features:** In this case, we extract the library calls from the executables and use them as features.

In order to keep things simple, we use only the binary n-gram features in our current work to show how this model can be defeated by SMalware. However, SMalware can be extended to defeat a model that uses all the three kinds of features explained above. Next, we describe how the binary features are extracted.

Binary n-gram feature extraction: First, we apply the UNIX hexdump utility to convert the binary executable files into text files, mentioned henceforth as "hexdump files", containing the hexadecimal numbers corresponding to each byte of the binary. This process is performed to ensure safe and easy portability of the binary executables. The feature extraction process consists of two phases: (a) feature collection, and (b) feature selection.

The basic feature collection process runs as follows. Let the set of hexdump training files be $\mathcal{H} = \{h_1, ..., h_b\}$. At first, we initialize a set L of n-grams to empty. Then we scan each hexdump file h_i by sliding an n-byte window. Each such n-byte sequence is an n-gram. Each n-gram g is associated with two values: p_1 and n_1, denoting the total number of positive instances (i.e., malicious executables) and negative instances (i.e., benign executables), respectively, that contain g. If $g \notin L$, then it is added to L, and p_1 is initialized to 1 and n_1 is initialized to 0 if h_i is positive and vice versa. If $g \in L$, then p_1 (n_1) is incremented if h_i is positive (negative). When all hexdump files have been scanned, L contains all the unique n-grams in the dataset along with their frequencies in the positive and negative instances.

There are several implementation issues related to this basic approach. First, the total number of n-grams may be very large. For example, the total number

of 10-g in our dataset is 200 million. It may not possible to store all of them in computers main memory. To solve this problem, we store the n-grams in a disk file F. Second, if L is not sorted, then a linear search is required for each scanned n-gram to test whether it is already in L. If N is the total number of n-grams in the dataset, then the time for collecting all the n-grams would be $O(N^2)$, an impractical amount of time when N=200 million. In order to solve the second problem, we use a data structure calledAdelson Velsky Landis (AVL) tree [3] to store the n-grams in memory. An AVL tree is a height-balanced binary search tree. This tree has a property that the absolute difference between the heights of the left sub-tree and the right sub-tree of any node is at most one. If this property is violated during insertion or deletion, a balancing operation is performed, and the tree regains its height-balanced property. It is guaranteed that insertions and deletions are performed in logarithmic time. So, in order to insert an n-gram in memory, we now need only $O(log_2(N))$ searches. Thus, the total running time is reduced to $O(Nlog_2(N))$, making the overall running time about 5 million times faster when N as large as 200 million. Our feature collection algorithm implements these two solutions.

Feature selection If the total number of extracted features is very large, it may not possible to use all of them for training because of several reasons. First, the memory requirement may be impractical. Second, training may be too slow. Third, a classifier may become confused with a large number of features, because most of them would be noisy, redundant or irrelevant. So, we are to choose a small, relevant and useful subset of features. We choose information gain (IG) as the selection criterion, because it is one of the best criteria used in literature for selecting the best features. IG can be defined as a measure of effectiveness of an attribute (i.e., feature) in classifying a training data [6]. If we split the training data based on the values of this attribute, then IG gives the measurement of the expected reduction in entropy after the split. The more an attribute can reduce entropy in the training data, the better the attribute is in classifying the data.

Now, the next problem is to select the best S features (i.e., n-grams) according to IG. One nave approach is to sort the n-grams in non-increasing order of IG and select the top S of them, which requires $O(Nlog_2N)$ time and $O(N)$ main memory. But this selection can be more efficiently accomplished using a heap that requires $O(Nlog_2S)$ time and $O(S)$ main memory. For S=500 and N=200 million, this approach is more than 3 times faster and requires 400,000 times less main memory. A heap is a balanced binary tree with the property that the root of any sub-tree contains the minimum (maximum) element in that sub-tree. First we build a min-heap of size S. The min-heap contains the minimum-IG n-gram at its root. Then each n-gram g is compared with the n-gram at the root r. If $IG(g) \leq IG(r)$ then we discard g. Otherwise, r is replaced with g, and the heap is restored.

Feature vector computation Suppose the set of feature selected in the above step is $\mathcal{F} = \{f_1, ..., f_S\}$. For each hexdump file h_i, we build a binary feature vector $h_i(\mathcal{F}) = \{h_i(f_1), ..., h_i(f_S)\}$, where $h_i(f_j) = 1$ if h_i contains feature f_j, or 0, otherwise. The training algorithm of a classifier is supplied with a tuple $(h_i(\mathcal{F}), l(h_i))$, for each training instance h_i, where $h_i(\mathcal{F})$ is the feature vector and $l(h_i)$ is the class label of the instance h_i (i.e., positive or negative).

4.2 Training

We apply SVM, Nave Bayes (NB), and decision tree (J48), classifiers for the classification task. SVM can perform either linear or non-linear classification. The linear classifier proposed by Vladimir Vapnik creates a hyperplane that separates the data points into two classes with the maximum-margin. A maximum-margin hyperplane is the one that splits the training examples into two subsets, such that the distance between the hyperplane and its closest data point(s) is maximized. A non-linear SVM [1] is implemented by applying kernel trick to maximum-margin hyper-planes. The feature space is transformed into a higher dimensional space, where the maximum-margin hyperplane is found. A decision tree contains attribute-tests at each internal node and a decision at each leaf node. It classifies an instance by performing attribute tests from root to a decision node. Decision tree is a rule-based classifier. Meaning, we can obtain humanreadable classification rules from the tree. J48 is the implementation of C4.5 Decision Tree algorithm. C4.5 is an extension to the ID3 algorithm invented by Quinlan. In order to train a classifier, we provide the feature vectors along with the class labels of each training instance, that we have computed in the previous step.

4.3 Testing

Once a classification model is trained, we can assess its performance by testing it against an instance (i.e., executable) that has not been seen by the model at training time. In order to test an executable h, we first compute the feature vector $h(\mathcal{F})$ corresponding to the executable, in the same way explained above. When this feature vector is provided to the classification model, the model outputs (predicts) a class label $l(h)$ for the instance . If we know the true class label of h, then we can compare the prediction with the true label, and check the correctness of the learned model.

In the next section, we describe a technique that can be applied to defeat a malware detection model like the one explained above.

5 Obfuscation techniques

Now we describe a malware obfuscation technique that can defeat a data mining based malware detector, which is built using a decision tree classifier. We choose

decision tree so that the SMalware technique can be easily understood. However, this technique can be generalized for any data mining based malware detector.

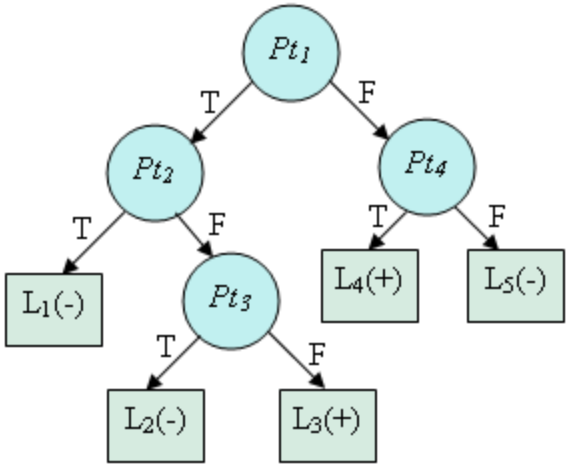

Figure 3: An example of a decision-tree based malware detector

Figure 3 shows a simple decision tree model for malware detection. Each internal node in the tree corresponds to an attribute test. For example, the root has the attribute test pt_1. A test instance x (i.e., executable) is first tested against the root. If x has the attribute pt_1, then the left branch ('T'rue) is followed by x, otherwise, the right branch ('F'alse) is followed. The leaf nodes contains the final decision, denoted by '-' (benign) or '+' (malicious). For example, the leaf node L_1 corresponds to a '-' decision. Here we consider that each attribute corresponds to a particular pattern (i.e., feature) in the executable.

According to the model in figure 3, if a malware x has pattern pt_1 and pt_2, then it will be detected as benign or, if it does not have neither pt_1 or pt_4, then it will be detected as benign. In order to evade detection, the malware must be detected as benign by the model. This can be done by inserting the patterns pt_1 and pt_2 into the malware, or removing both pt_1 and pt_4 from the malware.

5.1 Overview

Malware detectors based on static data-mining attempt to learn correlations between the syntax of untrusted binaries and the (malicious or benign) behavior that those binaries exhibit when executed. This learning process is necessarily unsound because most definitions of "malicious behavior" are Turing-undecidable . Thus, every purely static algorithm for malware detection is vulnerable to false positives, false negatives, or both. Our obfuscator exploits this weakness by discovering false negatives in the model inferred by a static malware detector.

We discover false negatives using one of two techniques -

- Pattern Insertion and
- Pattern Mutation

Of the two, pattern insertion requires significantly less effort and is the first vulnerability we look to exploit. Some static models cannot be exploited using pattern insertion, and in such cases we show that a form of pattern mutation can be used to obfuscate the malicious nature of the malware. In the following examples, we assume that the decision tree model has been constructed using the binary n-gram features. Thus, each pattern pt_i is a binary n-gram, that is, a string of n consecutive bytes in the binary executable.

5.2 Pattern insertion

This technique analyzes the model and adds a pattern to the malware so that the malware is diagnosed as "benign" by the model.

Methodology Each path from the root to a leaf node in a decision tree corresponds to a rule, which is a conjunction of conditions. For example, in figure 3, the leaf node L_2 corresponds to the following rule (R_2):

$$R_2 : pt_1 \wedge \neg pt_2 \wedge pt_3 \Rightarrow benign \tag{1}$$

which says that if a test instance has pattern pt_1 and does not have pattern pt_2, and has pattern pt_3, then it is benign. Here the negative literal $\neg pt_2$ indicates that this pattern must not be present in the test instance. In this attack, the obfuscator examines the decision tree and tries to identify a path from the root to a leaf node that satisfies the following conditions -

1. All the negative literals in the rule corresponding to the path from root to the leaf node are satisfied.
2. There is at least one positive literal in the rule that is not satisfied.
3. The leaf node is classified as benign.

For example, with reference to figure 3, suppose the malware x does not have pt_2. Then the following leaf nodes satisfy the above conditions: L_1, and L_2, both of which are negative (benign) decision nodes. L_1 is satisfied because the rule corresponding to L_1 does not have any negative literal. Meaning, all we have to do is to insert both pt_1 and pt_2 in the malware. L_2 is satisfied because the negative literal $\neg pt_2$ is true (since x does not have pt_2). The presence of such a path is a necessary and sufficient condition for a pattern insertion attack.

Let the positive literals in the rule be pt_1, pt_2,...,pt_n. Let the byte string that is formed from the concatenation of these patterns be called a "feature-string" fs.

$$fs = pt_1|pt_2|...|pt_n$$

where the symbol '—' stands for the string concatenation operation. A trivial implementation of the pattern insertion attack is to append the byte string fs to the end of the malware executable x.

Justification of insertion The extra bytes appended will not affect the functionality of the malware in any way because the presence of these bytes will not be reflected in the meta-data found in the header of the executable. As such, these extra bytes will never be used by the malware executable in its lifetime.

5.3 Pattern Mutation

If no path can be found that satisfies the conditions for a pattern insertion attack, it is still possible to obfuscate the malware binary using a pattern mutation attack. In this case, we first identify the leaf node where the malware x ends up, and generate the rule corresponding to the leaf node. Note that there must be one or more positive literals in the rule. Because, otherwise, the pattern insertion technique can be applied. We then list the set of positive literals $P_L = \{pt_1, ..., pt_n\}$. Next, we disassemble the malware binary and identify the assembly level instructions that correspond to each of the patterns $pt_i \in P_L$.

At this point there are multiple ways to proceed. The simplest form of mutation involves adding nop instructions in the middle of the group of assembly instructions corresponding to each pattern. This effectively changes the byte pattern of the reassembled binary, thus masking the patterns from the malware detector.

A more sophisticated strategy would involve identifying the sequence of assembly instructions $I_{pt_i} = \{i_1, ..., i_k\}$ that correspond to each pattern $pt_i \in P_L$ and replacing I_{pt_i} with a different set of functionally similar instructions, so that the modified binary does not contain pt_i.

Although we have not implemented pattern mutation in our obfuscater, we are confident that the pattern mutation attacks we have described can be used to successfully obfuscate a malware binary in those cases where the model is not vulnerable to a pattern insertion exploit.

6 Experiments

We design our experiments to run on two different datasets. Each dataset has different sizes and distributions of benign and malicious executables. We generate the binary n-gram features using the techniques explained in Section 4. Then we build classifiers using decision tree.

6.1 Dataset

We have two non-disjoint datasets. The first dataset (dataset1) contains a collection of 1,435 executables, 597 of which are benign and 838 are malicious. The second dataset (dataset2) contains 2,452 executables, having 1,370 benign and

1,082 malicious executables. So, the distribution of dataset1 is benign=41.6%, malicious=58.4%, and that of dataset2 is benign=55.9%, malicious=44.1%. This distribution was chosen intentionally to evaluate the performance of the feature sets in different scenarios. We collect the benign executables from different Windows XP, and Windows 2000 machines, and collect the malicious executables from (http://vx.netlux.org), which contains a large collection of malicious executables. The benign executables contain various applications found at the Windows installation folder (e.g. "C:/Windows"), as well as other executables in the default program installation directory (e.g., "C:/Program Files"). Malicious executables contain Viruses, Worms, Trojans, and Back-doors. We select only the Win32 Portable Executables (P.E.) in both the cases. We would like to experiment with the ELF executables in future.

6.2 Experimental setup

Our implementation is developed in Java with JDK 1.5. We use Weka ML toolbox (http://www.cs.waikato.ac.nz/ml/weka/.) for training decision tree classifier (the C4.5 algorithm).

6.3 Evaluation

In order to evaluate our technique on malware obfuscation, we randomly pick a malware called "Win32.Navidad.a", which is an email worm. Our malware detection model M sucessfully detected this as a malware. In order to defeat the model, the malware has been obfuscated with the following steps:

1. Generate the binary feature vector corresponding to the malware x using our technique described in section 4.3. Let the feature vector be $\mathcal{F}(x) = \{f_1(x), ..., f_n(x)\}$, where $f_i(x)$ is either zero or 1 depending on whether the feature (i.e., n-gram) f_i is absent or present in x.

2. Analyze the decision tree model M and identify a leaf L_i that satisfies the conditions for "pattern insertion" attack.

3. Identify the patterns (i.e., n-grams) that need to be inserted.

4. Insert the patterns into the malware using a hexadecimal editor.

Once the obfuscation is done, the the model M detects the obfuscated malware as "benign". It has been also tested that the obfuscated malware still has the identical functionality as the original malware.

7 Conclusion

References

[1] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *D. Haussler (Ed.), 5th annual ACM workshop on COLT, New York: ACM Press*, pages 144–152, 2003.

[2] M. Christodorescu and S. Jha. Testing malware detectors. In *Proc. of the ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA04)*, pages 34–44, July 2004.

[3] M. T. GoodRich and R. Tamassia. Data structures and algorithms in java (4th ed.). *New York: Wiley*, 2006.

[4] J. Z. Kolter and M. A. Maloof. Learning to detect malicious executables in the wild. In *Proc. of the tenth ACM SIGKDD international conference on knowledge discovery and data mining*, pages 470–478, 2004.

[5] M. M. Masud, L. Khan, and B. Thuraisingham. A scalable multi-level feature extraction technique to detect malicious executables. *Information System Frontiers*, 10(1):33–35, March 2008.

[6] T. Mitchell. Machine learning. *New York: McGraw-Hill*, 1997.

[7] M. Schultz, E. Eskin, E. Zadok, and S. Stolfo. Data mining methods for detection of new malicious executables. In *Proc. of the IEEE symposium on security and privacy*, pages 178–184, 2001.

Part III-B

Handling Untrustworthy Partners: Offensive Operations

Proactive, Automated Signature Reversal for Offensive Operations

Kevin Hamlen, Latifur Khan, Mehedy Masud, Bhavani Thuraisingham

Abstract

We propose a novel technique for malware development in which malware proactively learns and adapts to malware signature updates fully automatically in the wild, in order to avoid detection by signature-based protection systems. The technique serves as a general malware propagation mechanism to carry out sustained attacks against adversaries with signature-based malware protection, even in the presence of signature updates that may be specifically intended to defeat the attack. In addition to its usefulness in offensive operations, we believe that the research will identify important weaknesses and poor design decisions in existing anti-malware products that leave users vulnerable to such attacks, and suggest methods of mitigating or closing those vulnerabilities.

1. Introduction

Mobile code has become a ubiquitous component of almost all modern computing architectures, both networked and non-networked. Examples typical of networked architectures include web browsers that download and interpret web scripting code, automatic patching facilities that download and apply software updates, and email clients that download mobile code via email attachments. Non-networked architectures take a more manual approach to mobile code, such as when a user attaches a handheld device that uploads a new device driver or transfers other binary files to the system.

A practical reality is that most mobile code systems are vulnerable to malicious code attacks of some form. This is because most useful security properties are provably undecidable; thus, there is no sound and complete static analysis that can reliably distinguish policy-violating code from policy-satisfying code [1]. As a result, protection systems for mobile code architectures must rely upon unsound or incomplete enforcement strategies. A canonical example is that of anti-malware scanning technology, such as that found in commercial products like Norton Antivirus and McAfee VirusScan, which distinguish malware from benign software using *signature-matching*. Each signature encodes a set of bit-patterns specific to known malware. When untrusted software matches a signature in the detector's database, the detector identifies it as possible malware and takes steps to disable it. Signature-matching can take place at download-time, at load-time, or by examining the program's memory image at runtime [2].

Signature-matching is susceptible to both false-negatives and false-positives. To keep false-positives to a minimum, signatures must not include bit-patterns found in benign software lest that software be rejected by the protection system. To minimize false-negatives, the signature database must be frequently updated by human experts (or automated machine learning systems—see below) as new malware is identified. This makes signature-based malware detection ineffective against zero-day attacks since in those cases no signature has yet been generated. However, once the malware has been identified, signature-based approaches have proven to be an effective and efficient means of disinfecting compromised systems and for preventing reinfection by the same malware or its variants.

Carrying out a sustained attack on a system protected by signature-matching technology thus requires malware that can survive a signature update deployed in response to the attack. While there are numerous polymorphic worms that randomly self-modify in an attempt to defeat signature-matching (e.g., Storm, Melissa, MyDoom, etc.), prior work has demonstrated that each can be modeled by a suitably expressive signature (c.f., [3]). The fundamental weakness of these polymorphic worms is that they each employ a static self-obfuscation algorithm, leaving signature-generators free to develop a suitably expressive signature that identifies all possible outputs of that static algorithm. Once a single instance of the malware is identified and analyzed, it is therefore only a matter of time before an adequate signature-matching defense is implemented and deployed to defeat all variants.

1.1. Objectives

We propose to study a novel alterative approach to malware development that we term *proactive signature reversal*. In our approach the malware chooses its obfuscation strategy based on the contents of a rival signature database. The obfuscation is derived directly from the signature information so as to avoid detection by all signatures in the database. It then applies the obfuscation to itself, actively adapting to ongoing signature updates. Such malware therefore propagates in three steps: (1) *signature acquisition,* (2) *obfuscation derivation,* and (3) *self-obfuscation,* each of which is described in greater detail in the following sections.

Reliably detecting proactive signature-reversing malware is far more difficult than detecting conventional polymorphic malware because any signature developed in response to the attack has the effect of introducing new malware variants that did not exist (and were not possible) when the signature was devised. These new variants circumvent the new signature database with high probability. Thus, the malware adapts in the wild to new detection strategies, paralleling and resisting the efforts of human experts and expert systems to adapt signature databases to new malware threats.

The above goals give rise to the following set of research objectives:

- Discover and evaluate machine learning algorithms that can learn an adversary's signature database (or an adequate approximation) in the wild, using only information likely to be available to malware.

- Discover methods of automatically deriving semantics-preserving obfuscation functions from signatures.
- Apply techniques from language-based security, recursion theory, and cryptography to develop formal proofs of irreversibility, soundness, and transparency for the obfuscation functions and binary transformation algorithms we derive.
- Implement an automated, proactive, signature-reversing malware binary (with a benign payload) and test it against existing signature-based anti-malware products using UTD's secure SAIAL lab.
- Recommend defenses that could be used to protect against proactive signature-reversing attacks.

2. Technical Approach

Developing a proactive signature-reversing algorithm can be divided into three sub-problems: (1) acquiring the adversary's signature database or a suitable approximation in a fully automated, reliable manner, (2) deriving a semantics-preserving obfuscation function from a signature database, and (3) successfully applying the obfuscation function dynamically. Each of these problems is discussed in more detail below.

2.1. Signature Acquisition

In order for proactive signature reversal to be practical, malware must have some means of acquiring or approximating an adversary's signature database fully automatically on the fly. We plan to study several methods that leverage our prior work on automated malware classifiers to accomplish this.

In order to effectively interoperate with other system software, most antivirus products support an API via which operating systems and applications can initiate virus scans of specific files or processes and learn the results. For example, antivirus products compatible with Internet Explorer support Microsoft's `IOfficeAntivirus` COM interface [4], which exposes this functionality. Such APIs allow the underlying signature database to be queried by malware.[1] We believe that such a query interface is sufficient to learn relevant details of the underlying signature model [5], given a suitable machine learning algorithm.

Given a sufficient amount of instances (i.e., benign and malicious executables), a machine learning algorithm can learn to distinguish a malware from a benign executable by generalizing their common and contradictory characteristics. Our previous work [6] shows that automated classifiers, built using a collection of malicious and benign executables as training examples, can achieve very high prediction accuracies, and low false positives. If such a machine learning algorithm is embedded within a malware, the malware can collect samples (i.e., executables) from the host machine and query the

[1] To conceal this behavior from the user, the malware must of course take appropriate steps to prevent the antivirus product from displaying a graphical interface to the user in response to the query.

antivirus to know whether a sample is benign or malicious. In this way, the malware can build a classifier from the scratch, or enrich its existing classifier (if it already had one). As proven by [6], this classifier will have almost the same prediction accuracy as the expert-generated classifier (i.e., the antivirus). In other words, the knowledge of malicious "signatures" acquired by the automated classifier approximates the knowledge embedded into the expert-generated classifier. Our work could examine various different classification algorithms to discover which ones are most effective at approximating signature databases used by various antivirus products.

Some antivirus products, such as open-source products [7], distribute signature updates to the public in a standard format, or provide free detector engines that can identify (but not necessarily disinfect) known malware. In these cases proactively signature-reversing malware can obtain updated signature databases (or the equivalent) by directly downloading it from the same source that supplies updates to the installed antivirus product. Even when the victim's signature database is not freely distributed in this way, publicly available databases are useful as approximations of the victim's database since they are very likely to provide similar classification behavior on many inputs.

2.2. Obfuscation Derivation

Once an adequate model of the signature database has been acquired by the malware, it must be reformulated as an obfuscation function that transforms code that matches the signatures into semantically equivalent code that does not. The details of this process depend on the exact signature approximation model used. Our ongoing research on decision tree-based classifiers has demonstrated that salient features can be added and removed from binaries to defeat these classifiers using standard binary rewriting strategies such as payload encryption, instruction insertion, basic block reordering, and register re-allocation. [8].

It should be noted that besides decision tree, any rule-based classifiers, such as Ripper, decision-stump, decision table etc. can be applied as an obfuscation function. However, the possibility of using other kinds of classifiers, such as Support Vector Machine (SVM), bayes net etc. needs to be investigated, which can be considered as a future work.

Building upon this preliminary work, we plan to apply language-based security techniques such as type-preserving compilation and cryptographic one-way hash functions to obtain irreversible, provably semantics-preserving obfuscation functions from signatures.

2.3. Self-Obfuscation

The self-obfuscation operation itself involves applying the obfuscation function derived above to the malware binary itself and to copies produced during the course of malware propagation. When signature-matching occurs prior to runtime, the obfuscation function can simply be applied to the file image of the malware binary; however, defeating

runtime signature-matching requires a more sophisticated binary-rewriting strategy that obfuscates the runtime memory image of the process.

3. Conclusion

Proactive signature-reversing is a powerful but as yet unrealized form of malware propagation. By automating and reversing the processes currently used by signature-based malware defense technologies, such malware adapts dynamically to signature updates, rendering the updates ineffective. Investigating and evaluating the practicality of this approach is therefore an important step not only for effective offensive operational capability but also for protecting against reciprocal attacks.

4. References

[1] Hamlen, K. W., Morrisett, G., and Schneider F. B., *Computability Classes for Enforcement Mechanisms.* ACM Trans. on Prog. Lang. and Sys. (TOPLAS), 2006.

[2] Martignoni, L., Mihai, C., and Jha, S. *OmniUnpack: Fast, Generic, and Safe Unpacking of Malware*. In Proc. of the Annual Comp. Security Applications Conf. (ACSAC), 2007.

[3] Newsome, J., Karp, B., and Song, D. X. *Polygraph: Automatically Generating Signatures for Polymorphic Worms*. In Proc. of IEE Sym. on Security and Privacy (S&P), 2005.

[4] Microsoft Software Developer's Network Library. *IOfficeAntiVirus Interface.* http://msdn.microsoft.com/en-us/library/ms537369(VS.85).aspx

[5] Christodorescu, M. and Jha, S. *Testing Malware Detectors.* In Proc. of Int. Sym. on Software Testing and Analysis (ISSTA), 2004.

[6] Masud, M. M., Khan, L. & Thuraisingham, B. *A Scalable Multi-level Feature Extraction Technique to Detect Malicious Executables*. Information System Frontiers 10(1): 33-45, 2008.

[7] Clam Antivirus. http://www.clamav.net

[8] Hamlen, K. W., Mohan, V., Masud, M. M., Khan, L., and Thuraisingham, B. *Malware Obfuscation by Signature Reversal.* Submitted for publication, 2008.