

AN INVESTIGATION INTO IMMUNE-BASED INTRUSION
DETECTION

A Thesis Presented in Partial Fulfillment of the Requirments for the
Degree of Master of Science
with a
Major in Computer Science
in the
College of Graduate Studies
University of Idaho

by

John M. Hall

December 2003

Major Professor: Deborah A. Frincke, Ph.D.

**AUTHORIZATION TO SUBMIT
THESIS**

This thesis of John M. Hall, submitted for the degree of Master of Science with a major in Computer Science and titled “An Investigation into Immune-Based Intrusion Detection,” has been reviewed in final form. Permission, as indicated by the signatures and dates given below, is now granted to submit final copies to the College of Graduate Studies for approval.

Major Professor

_____ Date _____
Deborah A. Frincke

Committee
Members

_____ Date _____
Terence Soule

_____ Date _____
Carolyn Bohach

Department
Administrator

_____ Date _____
Bob Hiromoto

Dicipline’s
College Dean

_____ Date _____
David E. Thompson

Final Approval and Acceptance by the College of Graduate Studies

_____ Date _____
Margrit von Braun

ABSTRACT

This research evaluates and implements immune-based intrusion detection techniques in a research intrusion detection system. In this process, we introduce a generalized architecture and two prototype implementations. We also introduce the dynamic affinity, greedy detector set selection, and dynamic maturation time algorithms to improve the efficiency and applicability of immune-based techniques. We compare the first prototype to a simple misuse-based intrusion detection system and the second prototype to an advanced probabilistic method. In each case, we find the tested systems to be comparable. We also perform an evaluation of the dynamic affinity and greedy detector set selection algorithms. We find that both improve the quality of detector sets, but that the greedy detector set selection algorithm is inefficient for creating large detector sets. In more general terms, we show that immune techniques can be very successful in intrusion detection systems.

ACKNOWLEDGMENTS

This work was done under a Fellowship from the Hewlett-Packard Company.

CONTENTS

Title Page	i
Authorization To Submit Thesis	ii
Abstract	iii
Acknowledgments	iv
Contents	v
List of Figures	viii
List of Tables	ix
1 Introduction	1
2 Background	3
2.1 Computer Security	3
2.2 Intrusion Detection	3
2.2.1 Basic Concepts	4
2.2.2 History and Systems	6
2.3 Immunology	6
2.3.1 Human Immunology	7
2.3.2 Computer Immunology	9
2.3.3 Immunology-Based Intrusion Detection Systems	10
3 An Architecture for Intrusion Detection Modeled After the...	12
3.1 Introduction	12
3.2 Architecture	14

3.3	Implementation	15
3.4	Assessment	16
3.5	Future Work	18
3.6	Conclusion	20
4	An Evaluation of Dynamic Affinity	21
4.1	Introduction	21
4.2	Experiment	23
4.3	Results	26
4.4	Conclusion	30
4.5	Future Work	33
5	An Empirical Comparison of Probabilistic and Immunological...	34
5.1	Introduction	34
5.2	Best Sequence n-gram Modeling	35
5.2.1	N-gram Modeling	36
5.2.2	Parameters of an n-gram Model	38
5.2.3	N-gram Applications with Arbitrary Dependencies	40
5.3	Immunological Modeling	42
5.3.1	Negative Selection	42
5.3.2	Dynamic Affinity	45
5.3.3	Parameters of Dynamic Affinity	46
5.4	Experiment	46
5.5	Conclusion	51
6	immuneDM	52
6.1	Architecture	52
6.1.1	System Components	52

6.1.2	Data Objects	53
6.1.3	Installation into Hummer	55
6.2	File Formats	56
6.2.1	Configuration File	56
6.3	Background	57
6.4	Module Specifics	57
6.4.1	Parameters	57
6.4.2	Commands	59
6.5	Future Work	59
7	Discussion	60
8	Conclusion	64
	Bibliography	65
	Appendix	70

LIST OF FIGURES

3.1	Overview of ISNIDS	15
4.1	Coverage of Static Threshold vs. Dynamic Affinity Detector Sets . . .	26
4.2	Coverage of Standard Selection vs. Greedy Selection Detector Sets . .	27
4.3	False Positives of Static Thresholds vs. Dynamic Affinity	28
4.4	False Positives of Standard Selection vs. Greedy Selection	29
4.5	False Negatives of Static Thresholds vs. Dynamic Affinity	29
4.6	False Negatives of Standard Selection vs. Greedy Selection	30
4.7	Running Times of Static Thresholds and Dynamic Affinity	31
5.1	Perplexity With Training Size	48
5.2	Immunological False Positives	49
5.3	Comparison of best sequence n-Gram and immunological Models . . .	50
6.1	Architecture of immuneDM	52

LIST OF TABLES

6.1	immuneDM Attribute Codes and Ranges	58
A.1	Example Genes	70
A.2	Example Detector	70

CHAPTER 1

INTRODUCTION

This thesis investigates the use of human immunological concepts in the field of computer intrusion detection. In particular, this thesis documents work done to build an immune-based intrusion detection system. This research includes an investigation into a negative selection algorithm known as dynamic affinity.

Two immune-based intrusion detection systems have been built for this research. The first is a prototype known as the Immune System Intrusion Detection System (ISNIDS). This prototype explored immune concepts including negative selection, clonal selection, gene library evolution, and approximate binding. This system introduces the algorithm now known as dynamic affinity. This prototype is compared against a simple misuse-based intrusion detection system. This prototype confirms that an immune-based intrusion detection system shows promise.

The second system introduces a generic gene format. This format allows easily adapting the system to the available data sources. This system allows a more rigorous definition of events space which is necessary to evaluate the coverage of a detector set. This system evaluates the use of both the normal negative selection algorithm and the dynamic affinity algorithm. The system also evaluates the use of a greedy detector set selection algorithm. These evaluations show that dynamic affinity outperforms the standard negative selection algorithm. The greedy detector set selection algorithm is also shown to have some benefits.

This system is further evaluated by comparing it to another anomaly-based system that uses a probabilistic method. The immunological system compared uses the dynamic affinity and standard detector set selection algorithms. The comparison is done using an identical set of recorded events. This comparison shows

the surprising result that the false negative rates of each system are roughly equivalent for reasonable false positive rates.

This research also integrates this system into the HUMMER [18] architecture. The resulting decision module is known as the immuneDM. It generates events directly from network packet headers.

The rest of this document is organized as follows: Chapter 2 gives a basic background of intrusion detection and immunology. Chapter 3 contains a paper describing the ISNIDS prototype mentioned above. This paper was presented at CCCT 2003 this summer. Chapter 4 contains a paper that evaluates the effectiveness of the dynamic affinity algorithm. Chapter 5 contains a paper submitted to the IEEE Symposium on Privacy and Security that experimentally compares the effectiveness of an immune-based intrusion detection system and the effectiveness of a statistical detection method. Chapter 6 discusses and documents the integration of the immuneDM mentioned above into HUMMER [18]. Chapter 7 discusses what was learned throughout this work. Chapter 8 summarizes and suggests areas of future research.

CHAPTER 2

BACKGROUND

This chapter describes the concepts used throughout this paper. It provides a brief history to computer security and intrusion detection research. It also gives a high-level description of immunological basics and mechanisms.

2.1 Computer Security

Computer security has long been recognized as a serious issue. The problem of computer security was first formulated in a general fashion by Ware in [37]. Since these beginnings, computer security has continued to receive attention due to the rapid increase in security incidents [1].

For the purpose of this work, a security violation will be defined as any activity that is not explicitly permitted in a security policy. This security policy may or may not be formally defined. It should be noted that the security of a system and its usability are tradeoffs. Increasing the security of a system will tend to decrease its usefulness. As a practical matter, this makes it impossible to guarantee the security of a useful system. Intrusion detection improves this tradeoff by allowing activities to continue, but by trying to identify those activities which may lead to security violations.

2.2 Intrusion Detection

Intrusion detection attempts to identify activities that correspond to security violations. This recognition of security violations allows a system administrator to respond. The response to this violation may come in many forms. The knowledge of attacks gained may be used to improve the security of future systems. The knowledge may also be used to minimize the impact of the attack, for example by activating countermeasures.

2.2.1 Basic Concepts

Intrusion detection was first introduced by Anderson in [2]. This discussion is drawn primarily from this source. While some of the terminology used has since changed, the basic ideas remain the same. This discussion uses the terms currently in use by the research community. Prior to this point, identifying security violations was done by manually analyzing audit events recorded by a system. This audit data not explicitly generated for the use of identifying intrusions, but was the only data available. At the time, it was recognized that it was becoming infeasible to identify security violations using manual methods. As a result, intrusion detection was introduced to automate this process. Anderson introduced two fundamental methods of doing this, misuse-based and anomaly-based.

Misused-based, sometimes known as signature-based, detection methods identify violations by identifying those activities that are prohibited. Signatures are generated by looking for patterns in the audit data that are generated when the prohibited activities occur. Depending on the system and the activity, these signatures may be fairly straight forward, or they may be fairly complex. These signatures are typically chosen such that they have a low likelihood of matching audit events created from legitimate activities. This same process is sometimes used to detect activities that typically accompany security violations. For example, many intrusion detection systems have a threshold value for the number of failed logins. If the number of failed logins exceeds the threshold, the system generates an alert.

Anomaly-based detection methods use the opposite principle. Instead of trying to encode all abnormal behaviors, they build a model of normal behavior. This is typically done by taking some amount of audit data and assuming that it represents normal behavior. Audit data are then compared to this model. The system generates alerts for those events that fall outside some range of normal behavior.

Anderson introduced a simple model that found the mean and standard deviation of the attributes being considered. Those events that were outside some factor of the standard deviation were considered abnormal.

Both misuse-based and anomaly based systems have benefits and drawbacks. Misuse-based systems are typically fast and have low false positive rates, but they can only detect activities that have been previously considered. Anomaly-based systems can identify activities that have not been previously considered, but they typically require training and must update their model as the system profile changes. Typically, hybrid systems are used where misuse-based methods detect known attacks and anomaly-based methods detect unknown attacks.

It should now be clear that systems based on either method may generate an alert when there is no security violation or may fail to generate an alert when there is a security violation. The first case is known as a false positive, the system generated an alert when it should not have. The second case is known as a false negative, the system failed to generate an alert when it should have. Both false positives and false negatives represent a failure of the system. A system may also generate true positives and true negatives when it correctly identifies an attack or the lack of an attack correctly, but these values are not used as frequently to identify the effectiveness of a system. All systems may provide a tradeoff between false positives and false negatives. By decreasing the sensitivity of a system, fewer alerts are generated. This decreases the false positive rate, but increases the false negative rate. By increasing the sensitivity of a system, more alerts are generated. This decreases the false negative rate, increases the false positive rate. Historically, low false negatives were considered more important than low false positive rates. However, the rate at which audit data is produced has continued to increase. As a

result, even modest false positive rates can generate alerts more quickly than they can be investigated.

2.2.2 History and Systems

Since the introduction of intrusion detection in [2], there have been many research and commercial intrusion detection systems. The framework put forth in [12] spurred much of the formal study of intrusion detection. In order to put the current research in perspective, this section will discuss a small subset of systems. DIDS [36] introduced distributed collection of data. In this system, each host on the network runs a common manager program. This allows a centralized manager to monitor all hosts on the network as well as the network traffic itself. The primary downside to this approach is that the centralized manager becomes a performance bottleneck. EMERALD [35] extended this idea with a flexible hierarchical system that spans domains. In this system, data collection and analysis is done at every node in the system. However, this model assumes that all nodes in the system are maintained by the same group. The HUMMER [18] system introduces cooperative intrusion detection. This model introduces trust relationships between entities. This allows domains to collaborate to detect widespread intrusions while not fully trusting each other.

2.3 Immunology

The human immune system provides many of the same functions to the human body as an intrusion detection system provides to a computer system. Forrest et. al. first explored the idea that these immune concepts could be applied to computer security [16]. In order to understand how these concepts can be applied in this field, we first must have a basic understanding of how the human immune system works.

2.3.1 Human Immunology

The primary function of the human immune system may be viewed as differentiating between those things that belong in the body and those that do not. The human immune system distinguishes between self and non-self antigen. The process of detecting and removing non-self involves both innate and adaptive immunity. The innate components are nonspecific and unchanging with repeated exposure to antigen. Adaptive immunity is specific and includes memory that allows the immune system to respond more quickly the second time an antigen is encountered.

There are three basic components of the active immune response, T-cells, B-cells, and antibodies, which are similar to detectors in computer systems. Individual T-cells and B-cells have antigen receptors that are specific for one antigen epitope. The sum of all T and B-cells provides a repertoire of antigen receptors that can recognize more than 10^{16} antigens [22]. Both T-cells and B-cells are created in the bone marrow and acquire antigen receptors through a random recombination of gene segments. B-cells remain there to mature while T-cells mature in the Thymus. B-cells recognize free antigen and T-cells recognize antigen associated with self (MHC). During the maturation process, T-cells that react too strongly to self or not strongly enough are destroyed. It is believed that 90% to 95% of the immature T-cells are destroyed before maturing [4].

Antibodies are created by plasma cells that develop from activated B-cells. Antibodies are divalent meaning that they have two domains that can each interact with antigen. The structure of an antibody is 'Y' shaped. The biological function of the antibody is determined by the constant region at the N-terminus. The specificity of the antibody is determined by the antigen-combining site, a variable region at the C-terminus. The antigen-combining site on each branch binds to

molecules using relatively weak forces. As a result, the detector must match a molecule closely to cause the antibody to bind to the antigen. However, the detector only matches a small region of the antigen known as epitope. Different antigens that have identical epitopes may be matched by the same antibody. This process is known as approximate binding. Approximate binding and mutation during cell replication allow an antibody repertoire of an estimated $10^6 - 10^7$ proteins [4] to match an estimated 10^{16} distinct patterns [22].

When a pathogen invades the body, some are engulfed by antigen presenting cells such as B-cells and macrophages. The pathogen is digested and presented on the surface of the cell. When a helper T-cell with a matching antigen receptor interacts with this cell, it will recognize that there is a threat and will release cytokines to alert other components of the immune system. When these cytokines are received by a cytotoxic T-cell, it will be activated and will kill or lyse cells that present proteins that match the cytotoxic T-cell's antigen receptor. Likewise, when B-cells specific for antigen detect cytokines from helper T-cells, they become activated and divide. In addition, some of these B-cells become plasma cells that create and release antibodies of the same specificity. As the B-cells clonally expand, they mutate. As the B-cells are competing to better match the antigens, the ones with better antigen receptors will be more successful. This process is known as clonal selection and drives the immune system to improve the receptors for a particular antigen.

The human immune system incorporates an integrated response mechanism. This response could be dangerous if initiated against the wrong target. The above process is complex because it includes many checks and balances in order to minimize this possibility. In general, the biological response triggered by antibody is determined by the constant N-terminus of the antibody. There are 5 different

constant regions (isotypes) of antibodies. For more information, see [4]. Simply binding an antibody to an antigen may restrict it sufficiently by precipitating or agglutinating it to render it harmless. Antibodies may also initiate more active responses by triggering an enzymatic cascade known as complement. Part of this process, for example, enhances macrophages and neutrophils phagocytosis to kill pathogens that are tagged by particular antibodies. Other antibodies trigger the release of histamine.

Memory is an important hallmark of the response. The clones of B-cells and T-cells that were successful in removing an antigen become memory cells. The life-span of these cells is greatly increased. If the same pathogen attacks again, the secondary response initiated from these memory cells is significantly faster than the initial response. This secondary response is the concept behind vaccines.

The human immune system does have several weaknesses. Allergies are cases where the immune system overreacts to substances that are relatively benign. If a pathogen is very similar to self, such as some cancer cells, the negative selection process may not be able to generate appropriate immune cells. However, in general, the immune model elegantly and efficiently determines the difference between self and non-self. This model is so successful that its application in other fields is warranted.

2.3.2 Computer Immunology

Immunological concepts have been successfully applied to many computer applications [11]. There are several issues to consider when applying these concepts to computer systems. One of the primary issues is the efficiency in a computer system. In the human body, both detector creation and detection is a massively distributed operation. Current computer systems are much more limited. From a computer security point of view, the speed of the attack is also quite different. In a

computer system the probability of success of an attack increases rapidly over a period of hours while it is undetected [7]. Computer systems are also much more susceptible to system changes than the human body. Users can change behaviors, new programs can be installed, and new computers can be added to the network.

2.3.3 Immunology-Based Intrusion Detection Systems

The work on applying immunological concepts to intrusion detection systems was pioneered by Forrest et. al. in [16]. Forrest with other researchers applied the concepts to processes on an actual system in [15]. This work attempts to gain an understanding of a program's normal behavior by looking at system call sequences. They identify the self pattern for a process by recording all sequences of system calls of a given length that were normally executed. This database forms the self profile of the given process. Future sequences that do not appear in the database represent abnormal behavior.

Kim and Bentley introduced the Artificial Immune Model in [27]. They proposed applying the same basic principles to network traffic. Their initial experiments were promising, but they had difficulty making an efficient solution. They found that no short detectors could be generated in a reasonable amount of time using just negative selection [30]. Their solution was to *improve* the quality of their initial detectors by using a form of clonal selection. Their approach was to select those detectors that were successful and encourage those genes to be used in future detectors.

Around the same time, Dasgupta and González introduced a technique to evaluate how far an example of abnormal behavior deviated from normal. Their approach allows creating bounds at given ranges from normal behavior. This work is most interesting for their discussion of incorporating time into a model. Basically,

this is the simple notion that whether an event is normal depends on when it happens.

There are two ways to incorporate the time element into an immunological model. First, the event sequence as a whole may be considered as one antigen. In this model, the goal of immunology is to look for abnormal patterns in this sequence. The second option is to group related events together and look for abnormal sessions. This research focuses on the second of these methods.

CHAPTER 3

AN ARCHITECTURE FOR INTRUSION DETECTION MODELED AFTER THE HUMAN IMMUNE SYSTEM¹

We propose a novel architecture for an immunological network intrusion detection system, Immune System Network Intrusion Detection System (ISNIDS), suitable for inclusion in a broader-based multi-enterprise misuse management system. This paper will discuss the architecture, prototype, testing, and lessons learned from ISNIDS, as well as outlining the strategy for integration with a distributed/collaborative misuse management system.

This paper compares the prototype with a similar rule based system in both live and isolated conditions. The live testing was geared toward evaluating the number of false alarms generated under normal conditions. The isolated testing was geared toward evaluating the number of attacks missed under attack conditions. Each detection scheme detected six of the eight implemented attacks. ISNIDS missed one of two masquerading attacks and one password guessing attack. The rule-based system missed both masquerading attacks. As expected, this indicates that the two types of systems could effectively augment each other. The immune-based IDS offers considerable promise as traditional detection methods also have difficulty recognizing masquerading type attacks.

3.1 Introduction

Immunology based IDS apply human immune system concepts to network security [14]. The human immune system is very good at repelling a wide range of attacks in a dynamic hostile environment. Immunology is based on determining whether events belong to self or to non-self. This makes immunology approaches

¹This chapter consists of work that was presented at CCCT2003. The secondary author is Dr. Deborah A. Frincke. This work is referenced as [20].

anomaly based. This means that immunological approaches try to look for events that demonstrate the system is operating unusually.

ISNIDS implements immune detection in a hierarchical manner such as used in Hummer [18]. This hierarchy is formed by assigning each group of IDS a manager. In this system, the manager is responsible for creating detectors for the subordinates. The immune detectors used in ISNIDS are formed primarily through negative selection. Negative selection is the process in which random detectors are compared to normal events. In the typical negative selection process, the detectors which match a normal event are destroyed. The detectors that remain after the negative selection process will only match events that are not normal.

Unfortunately, this process makes creating detectors a computationally infusible operation. ISNIDS improves this process by dynamically reducing the sensitivity of detectors to guarantee that the detectors do not match the normal traffic. We call this process *dynamic approximate binding*². ISNIDS separates the collection of events, the grouping of events, the analysis of events, and the response to events.

In this preliminary stage, we wished to determine whether this is a feasible prototype with four attacks in mind. The goal was to detect and respond to these attacks automatically, effectively, and quickly using an immunological approach. These general attacks were doorknob rattling, password guessing, masquerading, and illegal data access attempts. Our results indicated that this approach shows considerable promise. The prototype detected 6 of 8 implemented attacks.

Because of the preliminary nature of this prototype, several issues were ignored to simplify development. These limitations would seriously hinder the use of this intrusion detection system in a production environment. The prototype was not designed for efficiency, robustness, security, or configuration.

²This process is now known as *dynamic affinity*.

3.2 Architecture

ISNIDS was designed as two systems, a primary IDS and a secondary IDS. These components communicate across the network. The primary IDS is centralized and is responsible for creating detectors. This is done using negative selection. In order to adapt to new attacks, the primary IDS also evolves its gene library using clonal selection, a process through which components of successful detectors are recombined using the evolutionary process to make new detectors. The secondary IDS is distributed and is responsible for data gathering, data reduction, detection, and response. It also forwards successful detections to the primary IDS. This architecture is similar to the artificial immune model proposed by Kim and Bentley [27]. Their model offered significant promise. This system is intended to show the results and practical concerns of implementing such a system in a large environment.

In terms of the human immune system, the centralized primary IDS represents the thymus and to some degree the bone marrow. The decentralized secondary IDS represents the mobile portions of the immune system. In particular this IDS represents the flow of immune system detectors throughout the body.

The architecture of ISNIDS is shown in Figure 3.1. The secondary IDS consists of four components, the sensors, the packager, the detector, and the response. The primary IDS consists of only an analysis component. The sensors collect audit information and convert it to a common event format. The packager performs data reduction by grouping the events into sessions. The analysis component uses these sessions to create detectors. The detector component matches current sessions to its detectors. Finally, the response component automatically responds to attacks. Ideally, once the secondary IDS had a set of detectors, it could continue to function even if the primary IDS failed.

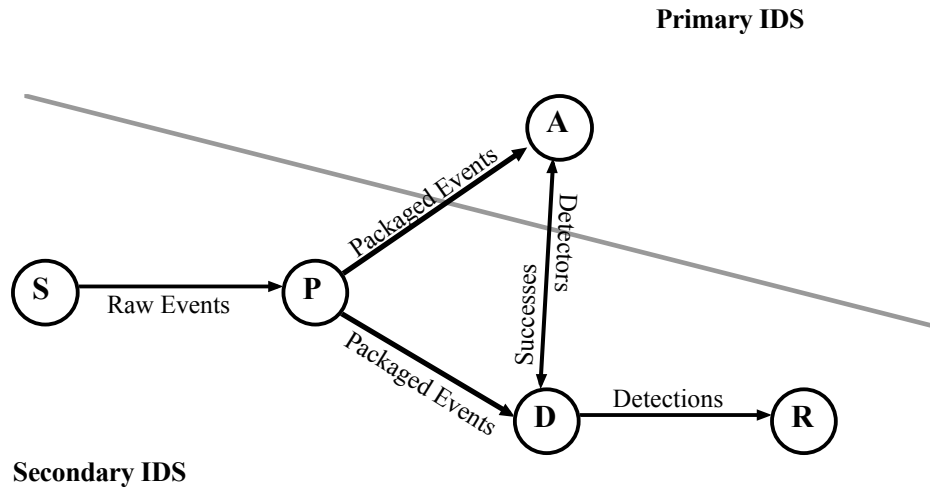


Figure 3.1: Overview of ISNIDS

3.3 Implementation

Detailed information of the overall ISNIDS system can be found in [19]. However, the implementation of several components warrant additional discussion. These are the packager component, the detection component, and the response component.

The packager component was originally missing from the architecture. Early experiments indicated that raw network events were insufficient for the IDS to effectively distinguish between attacks and normal behavior. The packaging method used should be investigated further. The current implementation groups events based on the associated user, the associated network address, and the event time.

Another interesting implementation detail is the detection method used by detectors to match event sessions. One of the largest problems facing the implementation of an immunological based IDS is negative selection [30]. In the human body, this is a massively distributed process. Kim and Bently found that generating a detector by applying only negative selection on network data matching

four or more consecutive attributes was a computationally infeasible operation [30]. They failed to generate a single detector with four genes that could survive negative selection after running their experiment for 24 hours. ISNIDS solves this by using *dynamic approximate binding*. In this process, immature detectors are created to match all events. Every time the detector matches a valid session, the binding constraints of that detector are tightened until it no longer matches. Using this method, a detector will only be discarded if it exactly matches all the values in a session.

The prototype implementation of the response component is also interesting. The response dispatcher will choose the response with the least impact. The impact of responses will change over time based on prior responses. The firewall response for example has several levels based on the current state of the firewall. The responses will also remember previous actions. For example, if the trace response has recently traced an IP address, then it will not consider itself a valid response if the same address is detected again. The system shutdown response is configured to have the most impact.

3.4 Assessment

The prototype was evaluated to determine the viability of this architecture. In order to facilitate this process, a rule-based detection scheme of similar complexity was also built. We tested the prototype and a similar rule-based system under both normal and attack conditions. The normal condition data was generated over a two and a half week period. Over this period all audit data was saved to files. These files were later sent to both systems. We generated two attacks based on each threat considered. The audit data collected over the duration of these attacks was also saved and later fed into the detection components. Our attacks were, a probe,

password guessing, masquerading, and attempts to locate private data. The audit data from each attack was sent to each system independently.

The immune detector detected 400 attacks out of the 2618 identified sessions in this data [19]. At over 20 detections per day, this seems unwieldy. However, almost all of these are doorknob rattling and worm attacks which all fall into our initial definition of attacks. There was one detection of particular interest. The events in this session were packaged incorrectly, merging legitimate activity with a doorknob rattling attack. The detector correctly identified the doorknob rattling portion as an attack. However, this may be a danger if an automated response may affect the legitimate activity. To summarize, we believe that ISNIDS generated only one false positive when run on 17 days of log events.

The rule-based detector detected only 2 attacks in this data. One of these appears to be a legitimate attack, the other appears to be the same incorrectly packaged event that ISNIDS detected. To summarize, the rule-based detector also made only one false positive when run on 17 days of data. The rule-based system did not detect the doorknob-rattling attacks simply because we did not write a rule with these attacks in mind. This was also acceptable behavior.

Both the immune-based and rule-based detectors performed exceptionally well in this test when evaluated with respect to number of false positives. They both incorrectly identified the same session as an attack.

The next step of the test plan was to feed attack data to both detection systems. For each of the four threats identified, a local and a remote attack was generated. Almost all of the detections occurred in less than a second after the packaging component released the session. The packaging component was designed to take at least 20 seconds. In most cases, this detection time seems reasonable. Both detection schemes detected six attacks and missed two attacks. Based on these

tests, this gives each detector a 25% false negative rate. This indicates that each detection scheme offers fairly good protection. It is interesting the the attacks missed by the rule-based system were the masquerading attacks.

3.5 Future Work

The response mechanism currently implemented in ISNIDS is not as good an immune response as it could be. In an immune system context, the current mechanism implements the two standard types of response: weakening the intruder and strengthening self. It also implements global and local responses. However, the system does not classify them by these attributes. Such a classification would aid the system level immune system view. Another weakness of the current implementation is that choosing an appropriate attack does not take into account whether the attack is still in progress. The current mechanism also does not allow cooperative responses between IDS.

The negative selection algorithm used was very effective at creating detectors. With early versions of the gene library, about 7 detectors were destroyed for every 1000 that matured. Later versions of the gene library lowered that to less than 1 for every 4000. That being said, using binary masks with counting values did not seem to work as well as it does for other values. The problem is that most of the sessions contain small values for these. As a result, the mask size tended to be quickly reduced. Perhaps a range type gene would handle these values better.

Building sessions worked well. However, there may be ways to do better. ISNIDS condensed 15,071 log entries to 3394 events to 2618 sessions in the original training data. Increasing the packaging delay did not drastically reduce the number of sessions. Allowing longer delays between events in the same sessions had a much larger effect on the number of sessions. This implies that there are a large number of small sessions generated.

Gene selection is a difficult problem. The current prototype makes it very difficult to add or remove genes. Simplifying this process is necessary before we can easily compare the effectiveness of different genes. A common representation for genes would be desirable. This would also seem to make the system more like the human immune system in that the human immune system represents every detector as a string of the same underlying protein structure.

Training is important. Originally, only about 2000 sessions were used to create the detectors. This was too few. This original detector set matched almost two-thirds of other sessions. Some less than optimal solutions including requiring multiple detector matches were tried. This was not viewed as a good long term solution because it uniformly increased the probability that a malicious event would be missed. The current training set consists of around 4000 sessions. Along with other changes made, the detection rate is much more reasonable. This does have the side effect of training the system to expect more anomalous behavior as normal.

Diversity in the gene library is essential. The original gene library was built from the training data. This resulted in detectors that matched almost no anomalies. Based on this library, only 2 of the false negative tests above passed. The initial generation of the library was not optimal. We suspect that more randomness is needed to effectively detect novel attacks.

Integrating this tool into the Hummer architecture [18] consists of two steps. First, ISNIDS must have a tool written to integrate its output to the Hummer system. This step may involve porting portions of the system to Java as the current prototype was written in C++. Second, ISNIDS must be integrated with the Hummer communication structure. This will allow the hierarchy needed by ISNIDS to update dynamically as Hummer manager/subordinate relationships change. This step may require extending the existing Hummer architecture to allow tool-initiated

communications. To allow efficient early prototyping, we simplified certain aspects of ISNIDS. In particular, we have not included key aspects of production IDS such as bandwidth scalability, speed of processing, intrusion/fault tolerance of the system, and configurability. Some of these may be the subject of future research while others are more appropriate topics of study when ISNIDS is situated in the Hummer architecture.

3.6 Conclusion

We built a prototype of an intrusion detection system that used human immune system concepts to detect attacks. This prototype shows considerable promise, especially at detecting unexpected attacks. The immune system model is often better at noticing patterns than a rule-based system. However, the immune-based system may miss some obvious attacks and raise alerts when exposed to rare but permissible activities. Our tests showed that the immune model performed comparably well to a rule-based prototype system of similar complexity. With this prototype we have shown that reasonably efficient intrusion detection may be performed using human immune system concepts. However, we would further recommend combining both detection methods to maximize the effectiveness of an IDS.

CHAPTER 4

AN EVALUATION OF DYNAMIC AFFINITY¹

The process of *Dynamic Affinity*, previously known as Dynamic Approximate Binding allows performing negative selection at a reasonable complexity within a computer system. This algorithm is primarily targeted at intrusion detection systems which utilize negative selection and other mechanisms that mimic the human immune system. This paper examines the effectiveness of this algorithm in generating small detector sets that cover a substantial portion of the search space. It also investigates using a greedy algorithm which adds detectors into the detector set which cover the largest additional region of the search space. We find that Dynamic Affinity improves the efficiency of detector generation, and that both Dynamic Affinity and greedy selection improve the quality of the generated detector sets. We also show that the detector sets created with the Dynamic Affinity algorithm have a higher false positive rate, but a lower false negative rate.

4.1 Introduction

Intrusion detection systems (IDS) are designed to detect attacks on computers systems. The human immune system performs a similar function within the body by discriminating between self and non-self. Using this immunological model within a computer system has shown considerable promise, especially at detecting novel attacks [15].

In the human immune system, the process of generating detectors relies heavily on a process known as *negative selection*. This process destroys those detectors which match self or normal behavior. In the human body, it is estimated that 90% to 95% of the unique detectors generated are destroyed during the negative selection

¹This chapter contains a draft work. This work has not yet been submitted. The secondary author of this work is Dr. Deborah A. Frincke.

process [4]. In the human body, this overhead is acceptable because the process is highly distributed. Unfortunately, it appears that a computer system matching a reasonable number of sequential events may have even larger destruction rates. In fact, one such experiment was unable to generate any short detectors using 24 hours of fast CPU time [30].

In order to overcome this limitation, ISNIDS introduced the concept of dynamic approximate binding [20]. We have renamed this process *dynamic affinity*. Dynamic affinity utilizes and extends the standard concept of approximate binding. Approximate binding is the process where a given detector is considered to match an antigen if the distance between the two is less than some threshold. Dynamic affinity extends this process by dynamically adjusting the threshold of a detector when it matches self or normal behavior during the negative selection process. As a result, instead of destroying the detector, the detector is adjusted to match fewer events. In particular, this new detector will no longer match the self or normal behavior presented to it. We suspect that the resulting detectors will form a tighter bound on the training data.

This research seeks to evaluate the dynamic affinity algorithm. In particular, it compares it to the standard static threshold algorithm. We find that the dynamic affinity algorithm is more efficient at generating detectors and that the detectors generated cover a greater portion of the event space. However, we also found that the detector sets created with the dynamic affinity algorithm will cause more false positives, but also fewer false negatives.

Immunology offers the ability to create a small number of detectors that match a large portion of the search space. Using smaller detector sets is of interest in a number of applications. A majority of the processing time utilized by a typical software IDS involves matching events against a detector set. By decreasing the

needed detector set size, the amount of time used to perform this matching may also be decreased. Minimal hardware IDS are another application of growing interest. These are small devices that can be connected anywhere on a network. To keep the cost of these devices to a minimum, they typically have minimal resources. As a result, these devices can only hold a small number of detectors.

Because of the importance of minimizing detector set size, this research also investigates the use of a greedy selection algorithm. We find that the greedy selection algorithm increases the size of the region spanned by the detector set. We were unable to show that it reduces the number of false positives when using the dynamic affinity algorithm. However, we do show that our greedy selection algorithm is slow for larger detector sets.

4.2 Experiment

This research seeks to determine the effectiveness of dynamic affinity. In order to do this, dynamic affinity is compared with the standard static threshold algorithm. In addition, for each of these two algorithms, a greedy selection algorithm is compared to the standard selection algorithm. The four resulting test cases are evaluated using detector set sizes of 5, 10, and 15 detectors. In order to get statistically significant results, each of these 12 tests are repeated for 25 trials using different random number seeds. The criterion measurements for each test include the total possible search space covered by each detector set and the number of detectors destroyed through the negative selection process. The processing time needed to create each detector set as well as an evaluation of the number of false positives and false negatives as determined by the detector set on some additional event data is also compared. These experiments evaluate whether the dynamic affinity algorithm is better than the static threshold algorithm, and whether the greedy selection is better than the standard selection method. Only those results

that have less than a 1% chance of occurring incorrectly due to random variation are considered statistically significant.

All trials use an immature detector set of size 4000. The number of self events used during the negative selection process may vary, but each detector must undergo at least 2000 negative selection iterations to mature. These values were chosen from past experience that suggests these values as minimum for an effective immunology based IDS [19]. The negative selection process stops as soon as enough detectors exist for the test. Tests using greedy selection create 10 detectors for each detector that is used. The test data is actual packaged system sessions as would typically be seen by an IDS. These sessions are generated from authentication events, firewall events, and web access and error events. The packaging process combines events which have the same user and occurred within the same 10 minutes, the same source address and occurred within the same 2 minutes, or the same network port and the same IP packet time-to-live and occurred within the same 30 seconds. Although the event data is recombined offline, the times associated with each event are used to assure a similar ordering of events as would have been achieved with live testing. An additional 1000 sessions are used to evaluate each detector set for false positives.

In order to evaluate the number of false negatives generated by each detector set, 10 simulated attacks were conducted. They were chosen because they were easy to implement and some events from these attacks would be available in the sources viewed by the system. The attacks implemented were:

1. Nmap scan from an external location
2. Nmap scan from an internal location
3. Internal telnet password guessing
4. Internal privilege escalation attempt

5. External http proxy attempt
6. Http password guessing
7. Http hunt for files
8. Http exploit attempt
9. Http put attempt
10. Http DoS attack

The system uses a generic range model to represent each attribute. In this model, each attribute value is converted to an integer format. Each event is treated as a single point in this multidimensional space. Sessions are combined by combining the values for each attribute independently. Detectors may accept a continuous range of values for each attribute. In addition, a detector may match a nonexistent attribute. The static threshold based detectors are created so this range varies uniformly between 30% and 130% of the search space where any values over 100% match all values. In addition, 80% of the detector attributes match nonexistent attributes. Static thresholds allows wrapping of values from the highest allowed value of an attribute to the lowest allowed value. Dynamic affinity detectors initially match all range values as well as nonexistent attributes. Each attribute also has an associated value created uniformly from the allowed attribute values. When a session is matched during negative selection, the attributes are looked at in random order. Attributes that are nonempty in the session are looked at first. This is done first to reduce the chance that detectors are reduced simply by requiring attributes that are rarely in the event data. The range of the chosen attribute is reduced to the largest continuous region that includes the attribute value, but that does not include any values from the session. If no such range is possible, that attribute is

not changed. This process continues until a range is changed, an attributed is modified to not match nonexistent attributes, or the detector is destroyed.

Each event, session, and detector, contains a common representation of a portion of the search space. The search space used in these experiments contains 13 dimensions. These are: user, source address, destination address, IP protocols, IP packet time-to-live, UDP and TCP ports, logon count, logoff count, logon failure count, web results, web protocol failure count, web file error count, and web access error count. The total search space created by these attributes contains $1.467e47$ individual points. The large number of dimensions and large search space make determining exact detector set coverage a computationally expensive operation. This operation is particularly expensive in the greedy tests as the coverage is recalculated for each potential detector every time a detector is added to the set.

4.3 Results

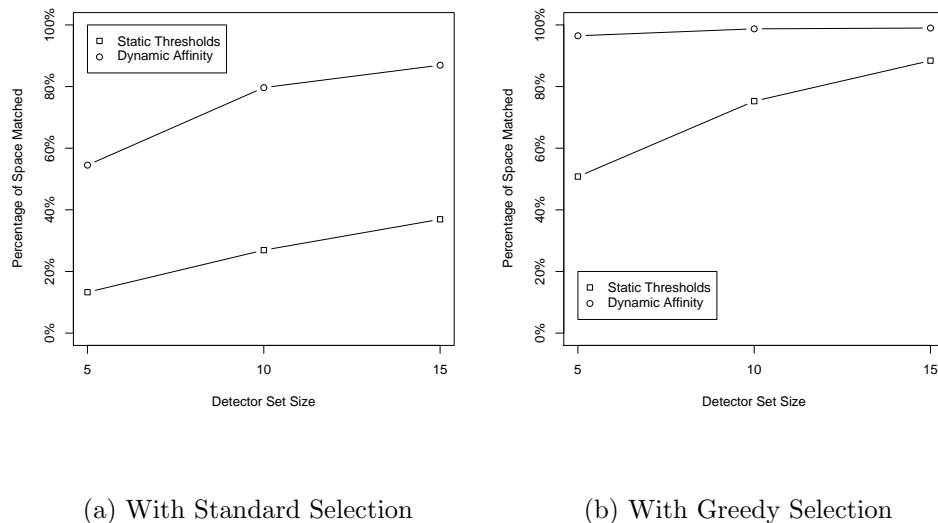


Figure 4.1: Coverage of Static Threshold vs. Dynamic Affinity Detector Sets

Figure 4.1 shows the results comparing the static threshold and the dynamic

affinity algorithms in terms of search space covered. At each measured point, the dynamic affinity algorithm produced a detector set that matched a greater percentage of the search space than the detector set produced by the static threshold algorithm. In each case, this relationship is statistically significant to at least our 1% requirement.

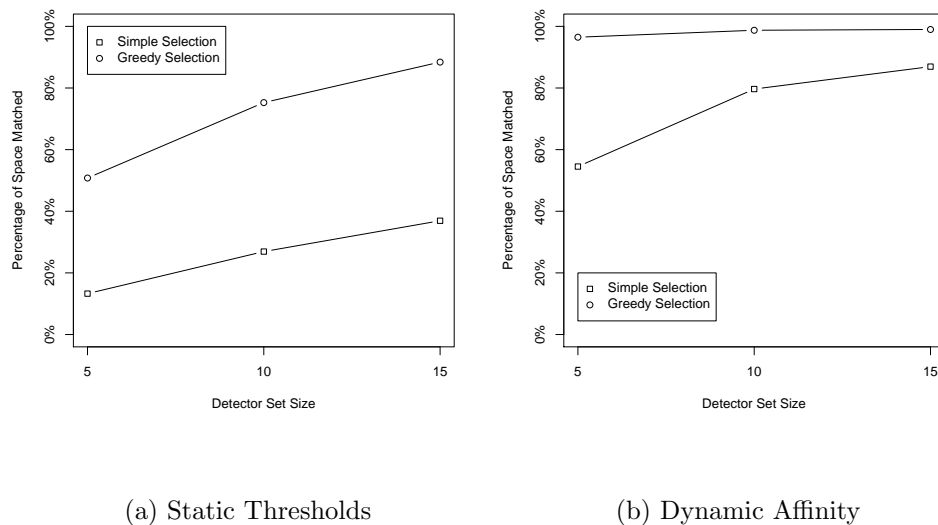


Figure 4.2: Coverage of Standard Selection vs. Greedy Selection Detector Sets

Figure 4.2 shows the same four plots. However, in this figure, the plots have been arranged to compare standard and greedy selection methods. At each measured point, the greedy selection method produced a detector set that matched a greater percentage of the search space. Again, this relationship is statistically significant to at least our 1% requirement.

The number of detectors destroyed during negative selection is also dependent on the algorithm used. Using static thresholds caused an average of 998 detectors to be destroyed per run. Since the immature population size was 4000, approximately one in four of the immature detectors never mature. The dynamic affinity algorithm

performed better. Dynamic affinity did not destroy any immature detectors throughout this entire experiment.

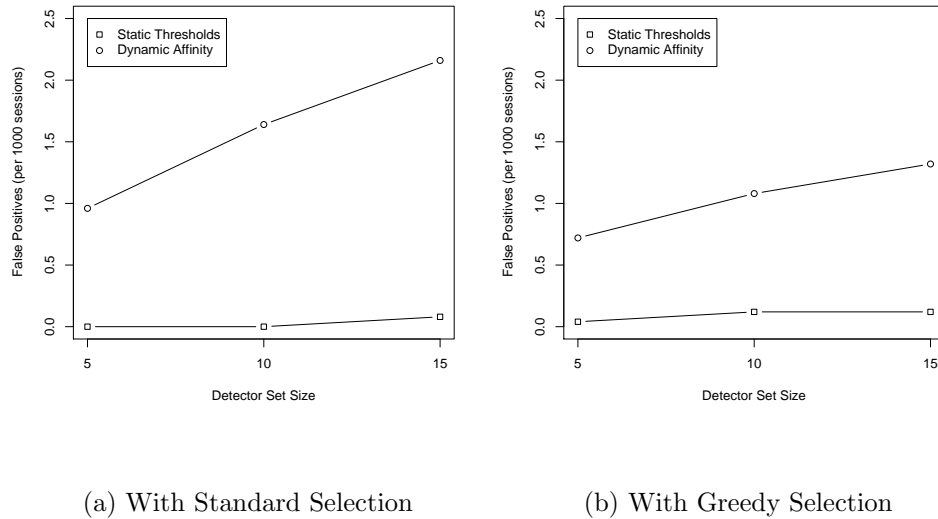


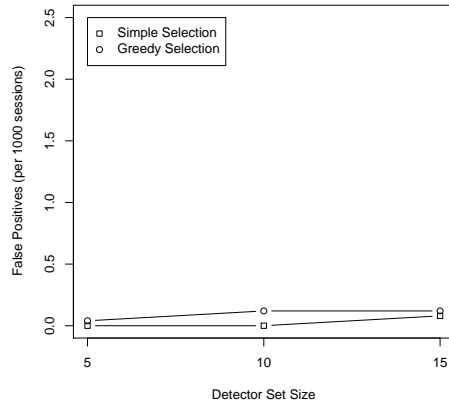
Figure 4.3: False Positives of Static Thresholds vs. Dynamic Affinity

Figure 4.3 shows the results comparing the static threshold algorithm and the dynamic affinity algorithm. Both with and without greedy selection, the dynamic affinity algorithm generated more false positives. This relationship is statistically significant to at least our 1% requirement.

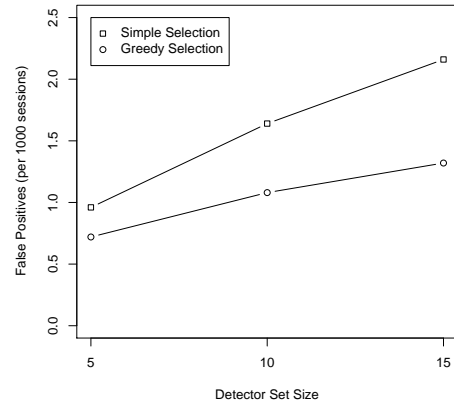
Figure 4.4 shows the same four plots from figure 4.3. However, they are rearranged to compare false positives based on the selection method. None of the differences shown on this plot were shown to be statistically significant.

Figure 4.5 shows the false negative rates comparing the static threshold algorithm and the dynamic affinity algorithm. Both with and without greedy selection the dynamic affinity algorithm generated fewer false negatives. This relationship is statistically significant to at least our 1% requirement.

Figure 4.6 shows the same four plots from figure 4.5. However, they are

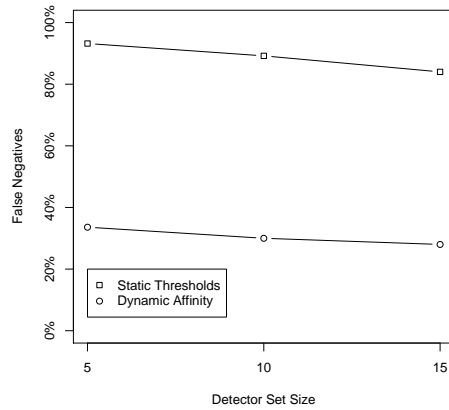


(a) Static Thresholds

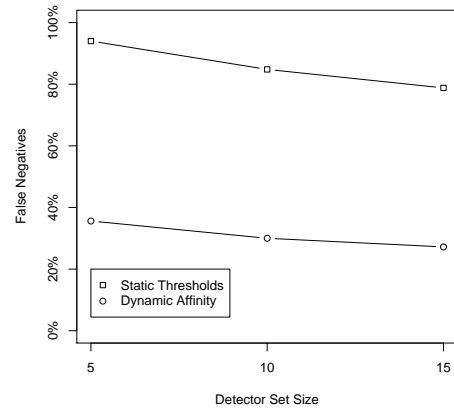


(b) Dynamic Affinity

Figure 4.4: False Positives of Standard Selection vs. Greedy Selection



(a) With Standard Selection



(b) With Greedy Selection

Figure 4.5: False Negatives of Static Thresholds vs. Dynamic Affinity

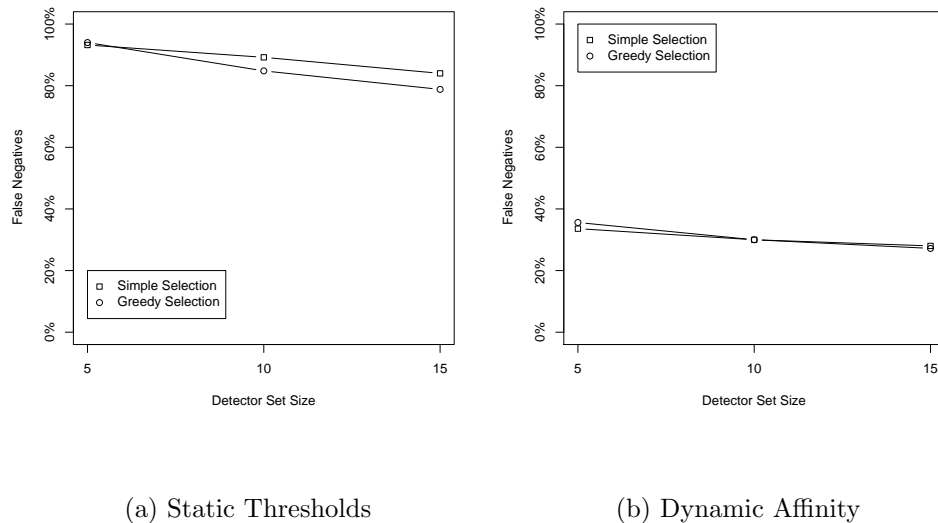


Figure 4.6: False Negatives of Standard Selection vs. Greedy Selection

rearranged to compare the false negatives based on the selection method. It appears that there is no statistical difference between the two selection methods.

The other interesting characteristic of these trials was the running time of each. Most tests ran between 30s and 35s per trial. The only exceptions were the tests using static threshold and greedy selection with 10 and 15 detectors. The average trial for each of these tests took 46s and 527s respectively. Figure 4.7 shows the running times of the static threshold algorithm and the running times of the dynamic affinity algorithm. For reference, the dynamic affinity plots include the results of informal tests using more than 15 detectors. It should be noted that a fair portion of this time, especially on the slower tests, was spent calculating the coverage of the detector sets.

4.4 Conclusion

Dynamic affinity works well. It effectively reduces the number of detectors destroyed during negative selection. It also allows detectors to match greater regions

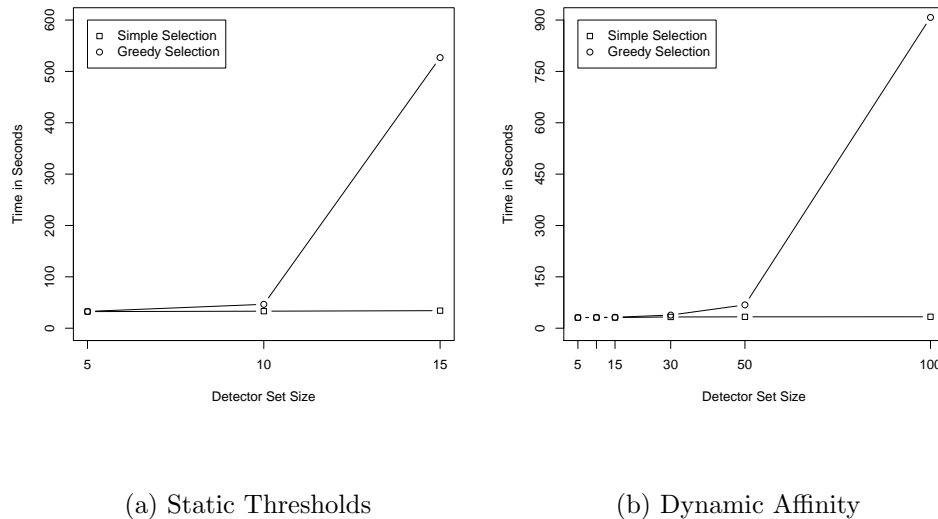


Figure 4.7: Running Times of Static Thresholds and Dynamic Affinity

of the anomaly space. This implies that the resulting detectors give a tighter bound on the training sessions. This closer bound to the training data is apparently the cause for both the higher false positive and lower false negative rates. Particularly when combined with greedy selection, the detector sets created using the dynamic affinity algorithm seem well suited to low footprint software IDS systems and low resource hardware IDS.

This research did not investigate the tradeoff between created detector size and the likelihood of being destroyed due to negative selection allowed by the static threshold algorithm. Using larger random detectors would, to a point, allow covering more of the search space. Unfortunately, the resulting detector set would be unlikely to perform well anywhere near training points. The algorithm used in this paper allowed the detector sizes to fluctuate randomly. This algorithm seemed to work fairly well, but could probably be tuned to perform better either in terms of

negative selection or in terms of difficulty to generate mature detectors. However, it is unlikely that it would reduce both.

The final algorithm used to measure the size of the space covered by a detector set maintained a canonical enumeration of all independent 13 dimensional blocks. This algorithm merged adjacent blocks whenever possible. While it worked better than its predecessors, this algorithm did not work very well when measuring a detector set consisting of static threshold based detectors. The algorithm was only able to handle about 20 detectors in a reasonable time. However, this algorithm was more efficient on dynamic affinity based detector sets. We suspect that this is due to the fact that these detectors tend to overlap at more common boundaries. Despite this, the exponential time of this algorithm seems clear from figure 4.7. Since many space calculations are necessary for the greedy algorithm, it becomes prohibitive for large detector sets unless a faster method of evaluating coverage is used. An approximation algorithm may make greedy selection feasible for larger detector sets.

This paper addresses the temporal nature of the training data only through the packager. Many research IDS systems view detectors as a sequence of events of the same format [15]. In this type of system, a match would be made by a sliding window of the input data against the detector set. Dynamic affinity could also be used in this sort of time based immunology algorithm. Each time an immature session matched an event from the training data, dynamic affinity could reduce the possibilities for one element of the detector or increase the detector length.

We identify two problems with using the above algorithms in an IDS. First, dynamic affinity creates detectors that neighbor the training data closely. This results in higher false positive rates as high as 1 out of every 450 sessions. We believe this is caused by forming a tighter bound on the training data. This appears to be confirmed by the lower false negative rate. Despite this, there may still be

ways to reduce the false positive rate. One possibility would be to increase the maturation time. Another solution to this problem would be to increase the affinity of the training data. This would have the advantage of being able to precisely control the deviation allowed from normal behavior, but would be likely to increase the false negative rates. The second problem is a little more difficult to address. Dynamic affinity combined with greedy selection may allow an attacker who can predict patterns in or access the training data to more easily predict what behavior would be considered normal. This could be done by looking through the training data for the small regions between training data points. However, it is believed that these smaller areas would be likely to overlap with legitimate activity. The proper solution would be to increase the size of the detector sets. It should be noted that this sort of attack is little different from the situations in other anomaly based IDS where attackers may try to attack the system with activities that are similar to normal behavior.

4.5 Future Work

As promising as these results are for the use of dynamic affinity, they do not address its effectiveness in an actual IDS. The next step to these experiments is to evaluate this algorithm in a running IDS. The results will need to consider both false positives and false negatives. This IDS will be integrated into the Hummer architecture [18]. The resulting system will be a completely functional digital immune system.

CHAPTER 5

**AN EMPIRICAL COMPARISON OF PROBABILISTIC AND
IMMUNOLOGICAL METHODS FOR ANOMALY-BASED
INTRUSION DETECTION¹**

This work describes two anomaly-based intrusion detection systems and compares them against common training and test data. A new system is introduced, based on a probabilistic method we call best sequence n-gram modeling. The second system is based on an immunological method that uses an alternative to the negative selection algorithm called dynamic affinity. The results of both systems are then evaluated by observing the relationship between the false positive and false negative rates. Surprisingly, despite the fundamentally different approaches used by the two systems, their performance was very similar.

5.1 Introduction

Intrusion detection is the field of computer security that involves identifying attacks on computer systems. In anomaly-based intrusion detection, computer activity is monitored and compared to historical norms. Anything new or different is automatically considered suspect.

By their nature, anomaly-based intrusion detection systems (IDS) involve data-mining and modeling to establish the baseline of *normal activity* against which future activity can be compared. Various techniques such as decision tree modeling [33], and Bayesian classification [32], and fuzzy logic [34] have been used.

In our work, we introduce a classification system based on an estimate of the

¹This chapter contains a draft submitted to the IEEE Symposium on Privacy and Security. The lead author of this work is Timothy Meekoff. The other authors are Dr. Deborah A. Frincke, Robert B. Heckendorn, and me. The most relevant parts to this research are those discussing the immunological method and those comparing this method to the best sequence n-gram modeling. The entire paper is left intact to adequately describe the comparison made.

probability distribution of events. To estimate the probability distribution, we have generalized n-gram modeling to form what we call *best sequence n-gram modeling*. This allows us to generate an accurate estimate of the probability distribution using maximum likelihood, even when the relationships among the data are unknown and the amount of training data is limited. We use the model to raise an alarm when the probability estimate of an observed event falls below a threshold.

A second approach used in intrusion detection is based on the way that biological immune systems distinguish between what is part of the body and what is not [16]. These immune-based anomaly detection systems are characterized by detectors that mimic the behavior of antibodies. We describe an algorithm called dynamic affinity, which produces an immune-based model that has a higher probability of covering anomalous events.

To compare the performance of the two systems, we examine the tradeoff made between the rate of false positives and negatives made by the two systems. By showing the rate of false negatives as a function of the rate of false positives, we are able to directly compare these two very different systems. Despite the strongly differing methodologies, we were surprised to find that our two systems produce roughly comparable results when trained and tested on our sample data. While the performance of the two algorithms presented has not been compared against previously published systems, we believe the algorithms themselves are promising and provide advantages over previous approaches. We are currently in the process of more fully evaluating their performance.

5.2 Best Sequence n-gram Modeling

Our first technique for detecting anomalous activity is based on building a probability model of normal activity. Given an event X , the model predicts the probability that it will occur as part of normal activity, $P(X)$. To detect anomalous

behavior, we choose a value τ which serves as a threshold. Any event whose probability is less than τ is treated as an anomaly.

N-gram modeling has been used in many fields as a tool for estimating the probability distribution of vectors in multivariate vector spaces. We describe the derivation of n-gram modeling and then describe a generalization that we call *best sequence n-gram modeling*. Unlike simple n-gram modeling, best sequence n-gram modeling is applicable to problems where we do not have *a priori* knowledge of the dependencies in the multivariate system.

5.2.1 N-gram Modeling

Our derivation of N-gram Modeling is drawn from [6]. Consider a discrete multivariate system in which observable events X are constructed from m elements x_1, x_2, \dots, x_m . Each x_i can take on any one of γ values. The joint probability of an event vector taking on a specific set of values is written as:

$$P(X = V) = P(x_1 = v_1, x_2 = v_2, \dots, x_m = v_m) \quad (5.1)$$

where the probability of vector, X , having a value, V , is the joint probability of the elements of the vector, x_i , having particular values, v_i . For simplicity we let $x_i = v_i$ be implied:

$$P(X) = P(x_1, x_2, \dots, x_m) \quad (5.2)$$

The joint probability distribution in Equation (5.2) has γ^m parameters to be estimated, where γ is the size of the range of each element x_i and m is the length of

the multivariate vector. γ^m grows extremely large and is not practical to estimate except for very small domains.

N-gram modeling begins by making the observation that Equation (5.2) is equivalent to:

$$\begin{aligned}
 P(X) &= P(x_1, x_2, \dots, x_m) \\
 &= P(x_1)P(x_2, \dots, x_m|x_1) \\
 &= P(x_1)P(x_2|x_1)P(x_3|x_2, x_1) \dots \\
 &\quad P(x_m|x_{m-1}, \dots, x_1)
 \end{aligned} \tag{5.3}$$

Equation (5.3) has just as many parameters to estimate as Equation (5.2). To make the estimation problem more tractable, we make an independence assumption that each factor in Equation (5.3) is the conditional probability of an element given only the previous $n - 1$ elements. In the general case of n , the equation is as follows:

$$\begin{aligned}
 P(X) &\approx P(x_1)P(x_2|x_1) \dots P(x_m|x_{m-1}, \dots, x_{m-n}) \\
 &\approx P(x_1)P(x_2|x_1) \dots P(x_n|x_{n-1}, \dots, x_1) \\
 &\quad \prod_{i=n+1}^m P(x_i|x_{i-1}, x_{i-2}, \dots, x_{i-n})
 \end{aligned} \tag{5.4}$$

We refer to n as the *order* of the model and the sequence $\langle x_{i-1}, x_{i-2}, \dots, x_{i-n} \rangle$ is the *context* of x_i . The independence assumption here is commonly called the Markov assumption and it forms the basis for Markov modeling and n-gram modeling. In practice, n-gram modeling has been limited to domains where there is an obvious underlying sequence to the elements of the multivariate system. The

presence of an underlying sequence allows one to apply the Markov assumption. In a later section we will show how we can find an appropriate sequence for domains in which we do not have an underlying sequence *a priori*.

The time to compute $P(X)$ given an n -gram model is of order $\mathcal{O}(nm)$ where n is the order of the model and m is the length of the multivariate-vector.

An n -gram model must store estimates of the values of $P(x_i|x_{i-1}, \dots, x_{i-n})$ for each x_i in every context $\langle x_{i-1}, \dots, x_{i-n} \rangle$. The amount of data required is of order $\mathcal{O}(\gamma^n)$ where γ is the size of the set of possible values for x_i . n is usually smaller than m and thus γ^n is much smaller than γ^m ; therefore, n -gram modeling allows for far fewer parameters than the full joint distribution.

5.2.2 Parameters of an n -gram Model

There are $\mathcal{O}(\gamma^n)$ parameters to be estimated in an n -gram model. Using maximum likelihood, we could estimate the parameters using this formula:

$$\hat{P}_{MLE}(x_i|x_{i-1}, \dots, x_{i-n}) = \frac{C(x_i, x_{i-1}, \dots, x_{i-n})}{C(x_{i-1}, \dots, x_{i-n})} \quad (5.5)$$

$\hat{P}_{MLE}(x_i|x_{i-1}, \dots, x_{i-n})$ denotes the maximum likelihood estimate of $P(x_i|x_{i-1}, \dots, x_{i-n})$. $C(X)$ denotes the number of times that X appeared in our training set T . $C(x_{i-1}, \dots, x_{i-n})$ can be small or even zero when n and γ are large unless the training set is extremely large. When this happens, Equation (5.5) is inaccurate or possibly undefined. To avoid this problem, n -gram models often use linear interpolation with models of lower order [3].

Using the notation $\hat{P}_n(X)$ to denote an order n estimation of probability for the multivariate X , we define the probability estimate of order n , $n > 1$, as follows:

$$\begin{aligned} \widehat{P}_n(x_i|x_{i-1}, \dots, x_{i-n+1}, x_{i-n}) &= \lambda_n \widehat{P}_{MLE}(x_i|x_{i-1}, \dots, x_{i-n+1}, x_{i-n}) + \\ &\quad (1 - \lambda_n) \widehat{P}_{n-1}(x_i|x_{i-1}, \dots, x_{i-n+1}) \end{aligned} \quad (5.6)$$

We want λ_n to be nearly 1 when $\widehat{P}_{MLE}(X)$ is reliable and nearly 0 when $\widehat{P}_{MLE}(X)$ is unreliable. In practice this is accomplished by making λ_n a function of $C(x_{i-1}, \dots, x_{i-n})$. Note that Equation (5.6) is a recursive definition which is anchored such that for \widehat{P}_1 , $\lambda_1 = 1$.

One simple estimator of λ_n is given in [8]. In this derivation, when $C(x_{i-1}, \dots, x_{i-n}) = 0$, $\lambda_n = 0$. When $C(x_{i-1}, \dots, x_{i-n}) > 0$, λ_n is given by the following:

$$\lambda_n = \frac{C(x_{i-1}, \dots, x_{i-n})}{C(x_{i-1}, \dots, x_{i-n}) + kD(x_{i-1}, \dots, x_{i-n})} \quad (5.7)$$

The value of k is an empirically derived constant, chosen such that it maximizes the likelihood of an additional set of training data. $D(x_{i-1}, \dots, x_{i-n})$ is the diversity of x_i in the context $\langle x_{i-1}, \dots, x_{i-n} \rangle$ and is related to the set Y defined as:

$$Y(x_{i-1}, \dots, x_{i-n}) = \{y | C(y, x_{i-1}, \dots, x_{i-n}) > 0\} \quad (5.8)$$

$Y(x_{i-1}, \dots, x_{i-n})$ is then the set of values of x_i that actually occurred in the training set, T , in the context $\langle x_{i-1}, \dots, x_{i-n} \rangle$. Diversity is defined as the size of the set Y :

$$D(x_{i-1}, \dots, x_{i-n}) = |Y(x_{i-1}, \dots, x_{i-n})| \quad (5.9)$$

When diversity is low and $C(x_{i-1}, \dots, x_{i-n})$ is large, then the value of λ_n given in Equation (5.7) is nearly 1. However, when the diversity is nearly equal to $C(x_{i-1}, \dots, x_{i-n})$ then the value of λ_n decreases, and depending on k , may be very close to zero.

5.2.3 N-gram Applications with Arbitrary Dependencies

N-gram modeling has been primarily used in domains where there is an obvious sequence to the multivariate system such that each element is closely related to its nearest neighbors and less related to its more distant neighbors. An example of such a domain is speech recognition, where n-gram modeling has been used to estimate the probability distributions of phoneme and word sequences [3]. N-gram modeling can be directly applied in these cases since the time sequence of the phoneme events is taken as the n-gram sequence of the multivariate system.

In our problem, the relationships between the attributes of events in the event space are unknown. Therefore, we do not have an obvious sequence available for the elements and cannot use the Markov assumption effectively. Our solution is to generalize n-gram modeling to form *best sequence n-gram modeling*.

We begin by assuming that there is a one-to-one function $s : \{1..m\} \rightarrow \{1..m\}$. The derivation for n-gram modeling can be expressed in terms of the sequence given by s :

$$P_s(X) = P(x_{s(1)})P(x_{s(2)}|x_{s(1)}) \dots P(x_{s(m)}|x_{s(m-1)}, \dots, x_{s(m-n)}) \quad (5.10)$$

The problem for n-gram modeling of a multivariate vector space can be reduced to finding the best sequence function s . To do that, we construct a figure of merit function $Q(s, T)$.

$$Q(s, T) = \sum_{X \in T} \log P_s(X) \quad (5.11)$$

where the value $Q(s, T)$ is the log likelihood of the training set given our n-gram model and the function s . Thus, the process of finding the best n-gram model for a dataset is the process of finding the set of n-gram model parameters and s^* that maximizes the log likelihood of our training set.

$$s^* = \operatorname{argmax}_s Q(s, T) \quad (5.12)$$

There are $m!$ possible functions s , which map $\{1 \dots m\} \rightarrow \{1 \dots m\}$. For any but the most trivial problems, it is impractical to search exhaustively. The technique we use is essentially a hill-climbing algorithm. An initial mapping function s_0 is chosen randomly and the corresponding log likelihood of the training set $Q(s_0)$ is calculated.

In a loop, s_0 becomes s_1 becomes s_2 etc. All $\binom{m}{2}$ possible single-position swap operations are tested to create $\binom{m}{2}$ new candidates s_j^k . The candidate s_j^k that has the largest figure of merit $Q(s_j^k, T)$ is taken and becomes s_{j+1} . The operation is repeated until no substitution improves the value of Q . At this point the algorithm has computed a function s and an n-gram model that maximizes the likelihood of seeing the training set given s .

There are potentially γ^n parameters in an n-gram model, but our training set T cannot have more than $\mathcal{O}(|T|m)$ parameters in it, so the time complexity for estimating the parameters of the model is $\mathcal{O}(|T|m)$. The complexity of calculating the merit function Q is $\mathcal{O}(|T|mn)$.

In each step of the algorithm, $\binom{m}{2}$ different permutations are tried and a model built and tested for each one. Assuming that the algorithm requires r repetitions to converge to locally optimal sequence, then the time complexity is $\mathcal{O}(rm^2(m-1)(1+mn)|T|)$.

In practice, we have found that our algorithm converges very quickly to a particular sequence, meaning that r is usually very small and is often very close in value to m , but very seldom larger. We do note, however, that our algorithm converges to a local optima that may or may not be the global optimum.

5.3 Immunological Modeling

The second of our two methods uses immunological concepts. Immunological algorithms have been used to find multivariate relationships in many fields [9]. Forrest et al. introduced immunological concepts in the field of intrusion detection [16]. The principle algorithm used in this process is negative selection. This section gives a mathematical background to negative selection. It then explains an alternative to this algorithm known as dynamic affinity.

5.3.1 Negative Selection

Negative selection is an algorithm seen in the human immune system. This algorithm finds complex multi-variable relationships using a fairly small detector set. In the human body, immature immune detectors are created to match random molecular patterns. The process of negative selection destroys those detectors whose patterns match molecules normally found in the body. Since those detectors which match self have been destroyed, anything matched by a mature detector must be non-self.

There are two basic types of immunological algorithms in use in intrusion detection systems. The first uses a sliding window to look for patterns of events over time as described in [15]. The second method, used in this paper, groups

related events together and then uses negative selection to form regions outside of the training events [10, 20].

To use the negative selection algorithm we choose a fixed number M of uniformly distributed random detectors. Each detector is said to *cover* a continuous region of the event space; and all detectors cover regions of the same size. We refer to these randomly generated detectors as immature detectors. We then check the detectors against the training set and discard any detector that covers an event from the training data. Those that remain are called mature detectors and form the set used to detect anomalies.

We need to quantify the ability of a set of detectors to cover anomalous events. To do so, we estimate the probability that an event that is not in the training set is covered by at least one of the detectors in the detector set. The remainder of this section gives the derivation of the formula to estimate this probability.

Consider one dimension of the event space, referred to as S . S is a finite, discrete space that has a one-to-one mapping to the set of whole numbers from 1 to $|S|$. The events in S are ordered and we refer to this ordering in terms of *left* and *right*. However, the space is considered to wrap such that the rightmost event is to the left of the leftmost event. The distance between any two events is one plus the size of the smallest possible set of events separating the two events. A set of immature detectors D is generated, where each detector $d \in D$ covers a subset of S . We define C to be the number of events covered by each detector d . There are $|S|$ possible detectors. The probability that an event $e \in S$ is covered by d is:

$$P_{cover}(d, e) = \frac{C}{|S|} \quad (5.13)$$

The probability that e is covered by at least one of the M detectors in D is:

$$P_{cover}(D, e) = 1 - \left(1 - \frac{C}{|S|}\right)^M \quad (5.14)$$

The expected number of events in S covered by D is $|S|P_{cover}(D, e)$. However, this coverage calculation does not take into account that the immature detectors which cover training events are destroyed. The mature detectors D that remain after negative selection bound training events into regions.

Between two regions of training set events there may be a region in which no training events occur. We refer to such a region as a *gap*. We evaluate the probability that a detector $d \in D$ will exist that covers an event e_g in a gap of size G . Clearly, if $G < C$ then the probability is 0, since no detector can fit.

The events in the gap which have a distance of at least C from either side of the gap follow Equation (5.14). The probability that an event e_g is covered by at least one detector $d \in D$ where r_g is the distance from e_g to the closest training event and $r_g \leq C$ is given by:

$$P_{gapcover}(D, e_g) = 1 - \left(1 - \frac{r_g}{|S|}\right)^M \quad \text{where } r_g \leq C \quad (5.15)$$

Since S wraps, the region between the leftmost and rightmost training event forms a single gap. The probability of an event e_g in a gap being covered by a detector set D is given by the equation:

$$P_{gapcover}(D, e_g) = \begin{cases} 0 & \text{if } G < C, \\ 1 - \left(1 - \frac{r_g}{|S|}\right)^M & \text{if } r_g \leq C, \\ 1 - \left(1 - \frac{C}{|S|}\right)^M & \text{otherwise.} \end{cases} \quad (5.16)$$

5.3.2 Dynamic Affinity

Dynamic affinity [21], formerly known as dynamic approximate binding [20], is an alternative to negative selection. Unlike in negative selection, the immature detectors generated for dynamic affinity do not cover a constant number of events. Instead they are initially generated to cover all events in the event space. Each detector is defined to include a randomly chosen event called the interior point. Instead of destroying a detector when it covers a training event, the size of the detector is reduced such that it includes only events between its interior point and the nearest training event in each direction. If the interior point of a detector is in the training set then the detector must be destroyed. This permits tighter bounds on the training data than the standard negative selection algorithm.

A training event is exactly bordered by a detector as long as there exists a detector with an interior point within the gap between that event and the neighboring training event. For a gap of size G , a total event space of size $|S|$ and an initial number of immature detectors M , the probability of the gap containing a detector is:

$$P_{gapcover}(D, e) = 1 - \left(1 - \frac{G}{|S|}\right)^M \quad (5.17)$$

In all cases, the probability given by Equation (5.17) is larger than the probability given by negative selection in Equation (5.16). This shows that a detector set created with dynamic affinity is more likely to detect anomalous events than one created with negative selection.

5.3.3 Parameters of Dynamic Affinity

The false positive and false negative rates generated by a detector set created with dynamic affinity can be adjusted by changing the detector set size. In general, gaps in the training data are assumed to correspond to anomalous activities. We may also assume that larger gaps have a higher probability of containing anomalous events.

For each gap in the training set, the probability that a detector covers an event in the gap is given in Equation (5.17). The probability of a gap being covered increases as the size of the gap increases. This corresponds with an increased probability that the gap represents anomalous activities. As the detector set size M increases, the detectors cover events from more gaps in the training events, but the probability of the gaps representing anomalous behavior decreases. This shows the tradeoff between false positives and false negatives. As M increases, false negatives decrease, but false positives increase. As M decreases, false positives decrease, but false negatives increase.

The training time is also dependent on the detector set size. Given a training set T , the dynamic affinity algorithm has complexity $\mathcal{O}(M|T|)$.

5.4 Experiment

Our experimental data was collected by recording events from a live server. Each event vector contains 13 attributes, which are listed below:

1. User
2. Source Address
3. Destination Address
4. Protocols

5. Packet Time To Live
6. Network Ports
7. Number of Logons
8. Number of Logoffs
9. Number of Logon Failures
10. Web Results
11. Number of Web Protocol Errors
12. Number of Web File Errors
13. Number of Web Access Errors

The raw events were correlated and grouped into sessions as discussed in [21]. Models from both methods were built using various sized training sets to show that each converges. In the comparisons between the models we used the largest training set size, 2000.

To measure false positives, we used a set of 1000 test events that were not used in training, but were drawn from the same source. To measure false negatives (and conversely true positives) we used a set of 10 attacks:

1. Nmap scan from an external location
2. Nmap scan from an internal location
3. Internal telnet password guessing
4. Internal privilege escalation attempt

5. External http proxy attempt
6. Http password guessing
7. Http hunt for files
8. Http exploit attempt
9. Http illegal put attempt
10. Http DoS attack

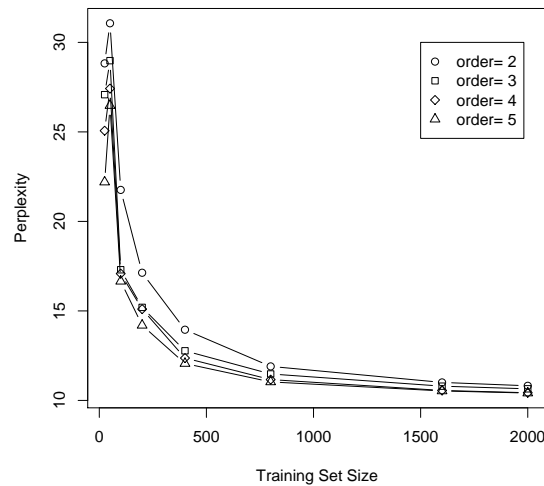


Figure 5.1: Perplexity With Training Size

Figure (5.1) shows the plot of the measured perplexity of the n-gram model against our test data as a function of training set size. Perplexity is a measure of the difficulty of predicting random data and is defined in [6] as:

$$Perplexity(T) = P(T)^{-\frac{1}{|T|}} \quad (5.18)$$

Perplexity is the inverse of the geometric average of the probability of the members of the training set. Perplexity directly corresponds to the difficulty of predicting events in the space. Perplexity grows larger as the problem of predicting becomes harder. Figure (5.1) shows that the measured perplexity of our test data converges as the training set size increases. This shows that training our model on the training set allows it to estimate the probability distribution of the test set.

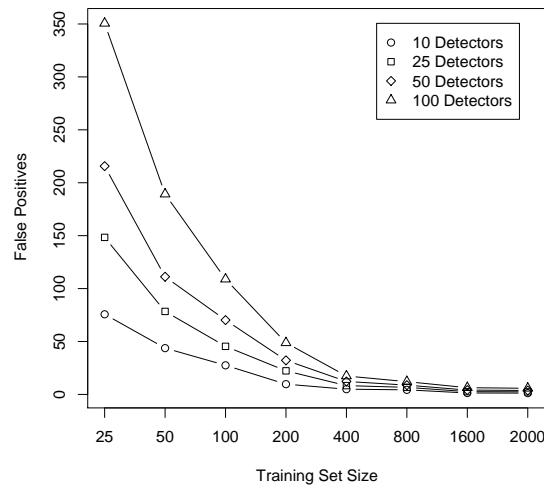


Figure 5.2: Immunological False Positives

Figure (5.2) shows the number of false positives generated by the immunological model. The shape of the graph shows that our immunological system is converging with larger sets of training data.

Both our systems, and all anomaly-based IDS systems in general, must make the trade-off between false negatives and false positives. In the case of the n-gram model, the threshold τ represents this tradeoff. In the case of the immune-based system, the detector set size M represents the tradeoff.

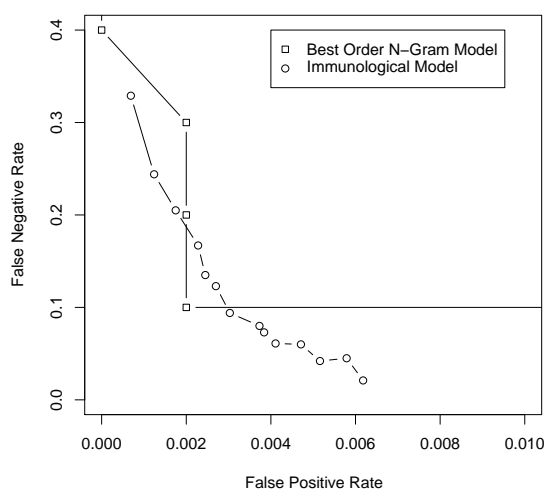


Figure 5.3: Comparison of best sequence n-Gram and immunological Models

Figure (5.3) shows the relationship of false positives and false negatives for both systems. The closer the graphs approach the origin the better the IDS system is. For a given rate of false positives, the model with the lowest false negative rate is the one which is more likely to detect real attacks. By comparing the performance of two IDS systems in this way, we are able to show the tradeoff between false positives and false negatives in a way that is independent of how the tradeoff is made. This is a general technique that can be used to compare any anomaly-based IDS systems.

We observe in Figure (5.3) that our two anomaly detection methodologies are very similar in the region to the left of 0.004 false positives. To the right they diverge. That point corresponds to 4 false positives per 1000 events. Given the many thousands of events that typically happen per day on a system, it is unlikely that a system administrator would tolerate a higher false positive rate. Therefore, it is the left of graph in Figure (5.3) that interests us. In that region, the performance of the two systems is similar.

One issue that we would like to pursue in future work is to examine the comparative performance of the two methodologies as the perplexity of the modeled system increases. As we can see in Figure (5.1), our training and test data have a relatively low perplexity, even though the theoretical event space is enormous.

5.5 Conclusion

In this paper we examined two anomaly-based IDS systems. First we introduced best sequence n-gram modeling as a tool to estimate the probability distribution of events. Its advantage is that it allows us to estimate the probability distribution even when the training set is limited and the relationships among the elements are unknown. We use the model in an IDS by estimating the probability of events and raising an alarm when the probability is below a threshold.

We also described an IDS based on immune modeling that uses the dynamic affinity algorithm. We showed that a model using dynamic affinity is more likely to cover anomalous events than a system that uses the standard negative selection algorithm.

In our experiment we demonstrated the rate of false negatives expressed as a function of the rate of false positives that our two systems produce over a variety of tuning parameters. By using this technique we were able to objectively compare their performance. Given the strong difference in methodology, the fact that they perform similarly is very promising. We are now in the process of further evaluating the performance of our systems and comparing them to other systems.

CHAPTER 6

IMMUNEDM¹

The immuneDM is an experimental decision manager for the HUMMER [18] IDS system. The concepts used in the immuneDM are leveraged heavily from the ISNIDS system developed at the University of Idaho and discussed in chapter 3. The immuneDM uses human immunological methods to mature detectors and then uses these detectors for detection. This maturation process depends heavily on the Dynamic Affinity algorithm discussed in chapter 4.

In the HUMMER architecture, decision managers perform analysis. Previous decision managers used misuse-based detection. The immuneDM provides anomaly-based detection to HUMMER. This improves HUMMER's ability to detect new attacks. It also gives a HUMMER system probabilistic detection. This means that an attack that is undetected by one system may be detected by another system even when the two systems have the same configuration.

6.1 Architecture

6.1.1 System Components

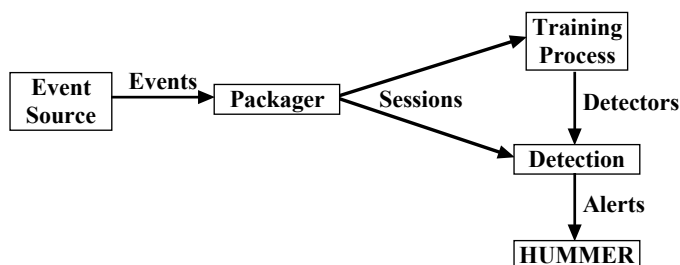


Figure 6.1: Architecture of immuneDM

¹This chapter is modified from work submitted to DARPA.

The immuneDM consists of 5 logical components as shown in figure 6.1. The first creates events. Currently this is done directly from tcpdump data. This data is read in by a separate thread and events are formed which are added to the event queue. The second component reads events from the event queue and combines related events into sessions. This task and the next two components are also performed by a separate thread. The third component matures detectors. This component applies negative selection to the immature detectors using the dynamic affinity algorithm. The fourth component performs detection. This component looks for matches between the mature detectors and the sessions generated. Any matches found denote a change of behavior. The alerts are generated into the alert queue. The fifth component communicates with the HUMMER system. This component runs on the main thread. This component listens for messages from the hummer system and responds appropriately. When this component is sent a heartbeat message, it will respond with an alert if there are any available in the alert queue.

6.1.2 Data Objects

The immuneDM relies heavily on the CImmuneSpace object. This object represents a region of the possible immune space covered by the attributes being considered. The only constraint imposed by this object is that the region covered combines the attributes independently. This class is not dependent on the actual attributes used. Attributes may be freely added, but will not have an effect until an event generator begins to add the new attribute to events. Attributes may also be freely removed. Any nonexistent attributes added by an event generators will be ignored. The values of the removed attributes in existing detectors will also be lost. At this time, the immune space consists of 22 attributes. These attributes are listed below. The immuneDM uses these 22 attributes because they are easy to determine

from packet dumps. Using these attributes defines a space which theoretically contains $3.627 * 10^{103}$ individual events.

1. Packet Count
2. Total Data Length
3. Fragmented Packet Count
4. Source IP Address
5. Destination IP Address
6. IP Packet Time-to-Live
7. IP Protocols
8. TCP Packet Count
9. Source TCP Ports
10. Destination TCP Ports
11. TCP Window Size
12. TCP Urgent Flag Count
13. TCP Ack Flag Count
14. TCP Push Flag Count
15. TCP Reset Flag Count
16. TCP Syn Flag Count
17. TCP Finish Flag Count

18. UDP Packet Count
19. Source UDP Port
20. Destination UDP Port
21. ICMP Packet Count
22. ICMP Packet Type

The CEvent, CSession, and CDetector classes all contain a CImmuneSpace. CEvents are primitive events. Typically, each gene attribute in the immune space consists of at most one element. These primitive events are combined by the packager or session generator into CSession objects. CSessions simply combine related events by merging the values for each attribute independently. CDetectors also consist of some portion of the immune space. In addition, detectors allow for matching empty or nonexistent attributes. A detector matches a session if every attribute in the session overlaps the corresponding attribute in the detector.

6.1.3 Installation into Hummer

Since the immuneDM follows the DM protocols, it may be added through the hummer DMSUserInterface. The command name should be “java” and the parameters should be “-cp ../lib/dms/ immuneDM”. An optional configuration file may follow these parameters. Once the immuneDM starts, it will take considerable time to mature a detector set. Until at least one detector has matured, the immuneDM will not generate any alerts. The actual time until the immuneDM has matured a complete detector set will depend on the volume of network traffic.

The immuneDM typically reports to the adDM. The adDM will automatically link to the immuneDM when it starts. This allows the output of the immuneDM to be limited to alerts. The adDM generates appropriate messages to the belief and

response servers. The immuneDM enables Heartbeats that cause HUMMER to routinely pull for new alerts. The immuneDM also executes `tcpdump`. This call is made using the “`sudo`” command. In order for this to work, either the immuneDM tool must be run as root, or in a test environment; the administrator password must be empty.

6.2 File Formats

6.2.1 Configuration File

The configuration file contains the immature detector set as well as the mature detector set. Detectors may appear in this file in any order, but they are written with the immature detectors first. This file is automatically written out regularly to record updates to these detector sets. The procedure used to write this file minimizes the likelihood that the file is corrupted. Each line in the configuration file contains a detector. The first space delimited word of each line determines the contents of the rest of the line. Detectors from the immature detector set are preceded with the keyword, “IMMATURE:”. This is followed by an integer which represents the number of sessions processed during the negative selection process since the last overlap with the training data. Detectors from the mature detector set are preceded with the keyword, “DETECTOR:”. The remainder of both lines is the actual detector data. The detector data is made up of one or more labeled ranges of data. Each label is a three character code that corresponds to an attribute. The attributes currently supported as well as the code and ranges of each are listed in table 6.2.1. Each character code is followed by a number in parentheses. This value is used primarily by immature detectors by the dynamic affinity algorithm. This is followed by a colon (‘:’) character. The range of values matched by the detector follows this. Detectors may match multiple discrete ranges of values for each attribute. However, the dynamic affinity algorithm only allows detectors

with a single continuous range of values. Ranges of values are separated with comma character (','),. Ranges may consist of a single value, or of a range of values specified with a minus ('-') character. These ranges must be in increasing numeric order. In addition, a range may contain the single value 'X'. This specifies that the detector will also match a session which does not contain the given attribute.

6.3 Background

As mentioned through this document, the immuneDM uses principles of the human immune system to detect attacks using anomaly based methods. This is done by generating detectors that do not match normal behaviors. These detectors are created primarily by a process called negative selection. More specifically, the dynamic affinity algorithm is used. Immature detectors are first created which match all possible events. Each attribute in each detector is also given a random point within the space of allowed values for the attribute. These immature detectors are then submitted to the dynamic affinity algorithm. This algorithm takes sessions from the training data and compares them with each immature detector. Those detectors which match are reduced in scope until they no longer match. This process is continued until an immature detector survives a predetermined number of sessions without needing to be adjusted. At this point, the detector matures. This matured detector is added to the detector set. Current sessions are also matched against the detector set. All matches indicate behavior that was not present in the training set. These sessions indicate suspicious behavior.

6.4 Module Specifics

6.4.1 Parameters

The immuneDM allows one command line argument. This is the file to use for the configuration file. If no file is specified, the default of "immune.conf" is used.

Code	Range	Description
PKT	$[0 - 2^{16})$	Packet Count
LEN	$[0 - 2^{32})$	Total Data Length
FRG	$[0 - 2^{16})$	Fragmented Packet Count
SRC	$[0 - 2^{32})$	Source IP Address
DST	$[0 - 2^{32})$	Destination IP Address
TTL	$[0 - 2^8)$	IP Packet Time-to-Live
PRO	$[0 - 2^8)$	IP Protocols
TCT	$[0 - 2^{16})$	TCP Packet Count
STP	$[0 - 2^{16})$	Source TCP Ports
DTP	$[0 - 2^{16})$	Destination TCP Ports
WIN	$[0 - 2^{16})$	TCP Window Size
URG	$[0 - 2^{16})$	TCP Urgent Flag Count
ACK	$[0 - 2^{16})$	TCP Ack Flag Count
PSH	$[0 - 2^{16})$	TCP Push Flag Count
RST	$[0 - 2^{16})$	TCP Reset Flag Count
SYN	$[0 - 2^{16})$	TCP Syn Flag Count
FIN	$[0 - 2^{16})$	TCP Finish Flag Count
UCT	$[0 - 2^{16})$	UDP Packet Count
SUP	$[0 - 2^{16})$	Source UDP Port
DUP	$[0 - 2^{16})$	Destination UDP Port
ICT	$[0 - 2^{16})$	ICMP Packet Count
ITP	$[0 - 2^8)$	ICMP Packet Type

Table 6.1: immuneDM Attribute Codes and Ranges

6.4.2 Commands

The immuneDM supports two runtime commands. First it supports report to commands. This command are specified with a text line containing “ReportTo”, the IP address of the hummer service to report to, and the name of the DM to report to. Each of these should be separated by whitespace. All other commands, including the empty line heartbeat command are treated as heartbeat messages. The immuneDM will either return an empty line message, or it will return an alert message packaged to the appropriate IP and DM.

6.5 Future Work

The immuneDM is currently experimental. This is primarily due to the fact that the immuneDM has not been fully tested. Ideally, this testing process would use a publicly available standardized data set such as from the HoneyNet project. In addition, several features have been omitted. These include more detailed failure reporting, using the proper HUMMER communication channels for communicating with sub-tools, including events from multiple data sources, dealing with non-ethernet tcpdump data, and allowing configuration of many internal constants.

CHAPTER 7

DISCUSSION

The primary goal of this research is to evaluate an immune-based detection method for inclusion in the HUMMER [18] intrusion detection system. In order to do this, we introduced a generic immune architecture and built and tested two prototype systems based on this architecture. The second prototype was successfully integrated with the HUMMER system as the immuneDM.

The basic architecture introduced consists of 4 primary components: sensors, packager, analysis, and detection. The sensor components creates raw events. The packager component groups related events into sessions. The analysis component generates new detectors. The detection component identifies anomalous sessions. One of the key features of this architecture is that the analysis and detection components receive exactly the same set of session data. In the original architecture [20], these components were split across a primary and a secondary intrusion detection system. On later systems, the same behavior was accomplished on a single system. Detectors generated in this fashion are similar to B-cells in the human body. However, analysis done on a manager that is receiving sessions from subordinates may be used in the future to generate a secondary detector that is more akin to a T-cell. There are three primary reasons why this was not done in the immuneDM. First, transmitting these events would have created a large amount of additional network traffic. The second, related issue is that the immuneDM monitors network traffic directly, such behavior could start what is known as a broadcast storm. For example, if one event was generated on a host it would be sent to the supervisor. This would result in an event being generated on every other host running the immuneDM on the local network. This would cause every host to also send an event to the supervisor. This effect can be reduced by adding delays, but

without ignoring events, it cannot be eliminated. The third reason is that it would decrease the survivability of the system in the event that the manager system was compromised.

Several other related methods were introduced in this research. The dynamic affinity algorithm was introduced to compensate for the limited computation available in computer systems. This algorithm improves the efficiency of the standard negative selection algorithm. It also creates tighter bounds on the set of training events, which improves detector set coverage. It also appears that this method better models the assumption that larger gaps in the training events corresponds to a higher likelihood of the gap representing anomalous activity. This research also introduced a generic gene format. This format simplifies the mapping of sessions onto a simple event space. In the immune system this is similar to the fact that all detectors consist of proteins. The research also introduces greedy detector set selection algorithm. This allows maximizing the coverage of small detector sets. In addition, a dynamic maturation time algorithm is introduced. This algorithm is designed to increase the maturation time when using complex training data and decrease the maturation time when using simpler training data. This algorithm simply resets the maturation counter for an immature detector whenever a training event overlaps the detector. A similar modification for the standard negative selection algorithm is significantly more complex.

The prototype systems were compared with several other systems. The first prototype was compared with a simple rule-based detection method. The effectiveness of both systems appeared similar. The second system used theoretical methods to evaluate the dynamic affinity algorithm and the greedy detector set selection algorithm. These experiments show that dynamic affinity increases the efficiency of detector creation and improves the quality of the generated detector

sets. These experiments also show that the greedy detector set selection algorithm improves the quality of the generated detector sets, but requires exponential time in terms of detector set size. With large detector set sizes, the greedy detector set selection algorithm also created unacceptable false positive rates. However, this research showed that this algorithm works exceptionally well for small detector sets. Small detector sets are important for low-footprint software and hardware intrusion detection systems.

The immuneDM uses the dynamic affinity algorithm with standard detection set selection. Greedy detector set selection was not used primarily because of the computational overhead. By using a larger detector set, the coverage of the event space is more uniform. The current version of the immuneDM uses only a single data sensor. This sensor reads raw packet headers from the network. The immuneDM uses the dynamic maturation time algorithm to allow the same configuration to be effective in many different network environments.

This research has supported many of the claims made of immune-based systems. Immune-based detection allows good detection characteristics with small detector sets. This is especially true using the dynamic affinity and greedy detector set algorithms. Immune-based detector sets have low size requirements. In addition, detector sets can be created with reasonable time complexity when using the proper algorithms. The benefits of probabilistic detection are seen most clearly in the evaluation phases of this research. By running multiple trials using different random number seeds the effectiveness of the algorithms used may be evaluated more precisely than can be done with other systems.

This research has also introduced some new claims. This research showed that dynamic affinity made it easier to tune the system by adjusting the detector set size. With the standard negative selection algorithm it is difficult to tune the

coverage of the initial random detectors without greatly increasing the number destroyed during maturation. Packaging the events before presenting them to both the detection and analysis simplified the detection scheme. Presenting the same data to both components in particular appeared to work well. This packaging process also made it easier to compare the immune-based system to other systems. This research also defined a standard event space. The size and complexity of this space clearly shows why intrusion detection is such a difficult problem.

CHAPTER 8

CONCLUSION

This research successfully evaluates immune-based intrusion detection techniques for a research intrusion detection system. The introduced architecture is implemented in two systems. These two systems are tested with other systems and found to compare favorably. Several algorithms for improving the effectiveness and efficiency of immune concepts are introduced. In particular the dynamic affinity and greedy detector set selection algorithms are shown to improve detector set quality. This research shows that immune-based techniques can be quite useful in intrusion detection systems, but there are still many open practical and research questions as discussed below.

This research has integrated these concepts into a decision manager for the HUMMER [18] system. However, this immuneDM is currently a prototype system. In order to assess the practical abilities of this system in a production environment, additional sensors must be added and extensive testing on data from complex network environments must be done. Ideally, this testing could be done on a standardized data set. This would allow effectively comparing of the results with those from other systems.

There are also many research areas still to be explored. These include evaluating maturation time algorithms, evaluating the use of a manager-based T-cell, and dealing with behavioral changes such as moving between networks. One other interesting research area is finding the best possible false positive to false negative tradeoff curve that is possible for a given data set with no heuristic knowledge. It is interesting that the n-gram model was so comparable to the immunological model. Perhaps both are approaching this ideal tradeoff.

BIBLIOGRAPHY

- [1] J. Allen, A. Christie, W. Fithen, J. McHugh, J. Pickel, and E. Stoner. State of the practice of intrusion detection technologies. Technical Report CMU/SEI-99TR-028, Carnegie Mellon Software Engineering Institute, Pittsburgh, PA, January 2000.
- [2] J. P. Anderson. Computer security technology planning study. Technical Report ESD-TR-73-51, Electronic Systems Division, Air Force Systems Command, Hanscom AFB, Bedford, MA, October 1972.
- [3] L. R. Bahl, F. Jelinek, and R. L. Mercer. A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(2):179–190, 1983.
- [4] E. Benjamini and S. Leskowitz. *Immunology: A Short Course*. Wiley-Liss, New York, second edition, 1991.
- [5] D. Bradley, C. Ortega-Sanchez, and A. Tyrrell. Embryonics + immunotronics: A bio-inspired approach to fault tolerance. J. Lohn, A. Stoica, and D. Keymeulen, editors, *The Second NASA/DoD workshop on Evolvable Hardware*, pages 205–224, Palo Alto, California, 13-15 2000. IEEE Computer Society.
- [6] P. F. Brown, V. J. D. Pietra, P. V. deSouza, J. C. Lai, and R. L. Mercer. Class-based n -gram models of natural language. *Computational Linguistics*, 18(4):467–479, 1992.
- [7] F. Cohen. Simulating cyber attacks, defenses, and consequences. *Computers and Security*, pages 479–518, March 1999.

- [8] M. Collins. *Head-Driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania, 1999.
- [9] D. Dasgupta and N. Attoh-Okine. Immunity-based systems: A survey. *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, Piscataway, NJ, 1997. IEEE.
- [10] D. Dasgupta and F. González. An immunity-based technique to characterize intrusions in computer networks. *IEEE Transactions on Evolutionary Computation*, 6(3):1081–1088, June 2002.
- [11] L. N. de Castro and F. J. V. Zuben. Artificial immune systems: Part ii - a survey of applications. Technical Report DCA-RT 02/00, Department of Computer Engineering and Industrial Automation, School of Electrical and Computer Engineering, State University of Campinas, SP, Brazil, February 2000.
- [12] D. E. Denning. An intrusion-detection model. *IEEE Transactions on Software Engineering*, SE-13(2):222–232, February 1987.
- [13] S. Forrest and S. A. Hofmeyr. Engineering an immune system. *Graft*, 4(5):5–9, 2001.
- [14] S. Forrest, S. A. Hofmeyr, and A. Somayaji. Computer immunology. *Communications of the ACM*, 40(10):88–96, 1997.
- [15] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff. A sense of self for Unix processes. *Proceedings of the 1996 IEEE Symposium on Research in Security and Privacy*, pages 120–128. IEEE Computer Society Press, 1996.

- [16] S. Forrest, A. S. Perelson, L. Allen, and R. Cherukuri. Self-nonself discrimination in a computer. *Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy*, pages 202–212, Oakland, CA, 1994. IEEE Computer Society Press.
- [17] S. Forrest, R. E. Smith, B. Javornik, and A. S. Perelson. Using genetic algorithms to explore pattern recognition in the immune system. *Evolutionary Computation*, 1(3):191–211, 1993.
- [18] D. A. Frincke, D. Tobin, J. C. McConnell, J. Marconi, and D. Polla. A framework for cooperative intrusion detection. *Proceedings of the 21st NIST-NCSC National Information Systems Security Conference*, pages 361–373, 1998.
- [19] J. M. Hall. A network intrusion detection system inspired by the human immune system. Technical Report CSDS-DF-TR-03-12, University of Idaho, 2002.
- [20] J. M. Hall and D. A. Frincke. An architecture for intrusion detection modeled after the human immune system. *Proceedings of the International Conference on Computer, Communication and Control Technologies*, volume 6, pages 75–78, 2003.
- [21] J. M. Hall and D. A. Frincke. An evaluation of dynamic affinity. Technical Report CSDS-DF-TR-03-21, University of Idaho, 2003.
- [22] S. A. Hofmeyr. An overview of the immune system. Technical Report <http://www.cs.unm.edu/immsec/html-imm/immune-system.html>, University of New Mexico, Albuquerque, NM, 1997.

- [23] S. A. Hofmeyr. *A Immunological Model of Distributed Detection and its Application to Computer Security*. PhD thesis, University of New Mexico, April 1999.
- [24] S. A. Hofmeyr and S. Forrest. Architecture for an artificial immune system. *Evolutionary Computation*, 8(4):443–473, 2000.
- [25] J. O. Kephart. A biologically inspired immune system for computers. *Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, pages 130–139, Cambridge, MA, US, 1994. MIT Press.
- [26] J. O. Kephart, G. B. Sorkin, M. Swimmer, and S. R. White. Blueprint for a computer immune system. *Proceedings of the Virus Bulletin International Conference*, 1997.
- [27] J. Kim and P. J. Bentley. An artificial immune model for network intrusion detection. *Proceedings of the 7th European Conference on Intelligent Techniques and Soft Computing*, 1999.
- [28] J. Kim and P. J. Bentley. The human immune system and network intrusion detection. *7th European Congress on Intelligent Techniques and Soft Computing*, September 1999.
- [29] J. Kim and P. J. Bentley. Negative selection and niching by an artificial immune system for network intrusion detection. S. Brave and A. S. Wu, editors, *Late Breaking Papers at the 1999 Genetic and Evolutionary Computation Conference*, pages 149–158, Orlando, Florida, USA, 13 1999.
- [30] J. Kim and P. J. Bentley. An evaluation of negative selection in an artificial immune system for network intrusion detection. L. Spector, E. D. Goodman,

- A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1330–1337, San Francisco, California, USA, 2001. Morgan Kaufmann.
- [31] J. Kim and P. J. Bentley. Towards an artificial immune system for network intrusion detection: An investigation of dynamic clonal selection. *Proceedings of the Congress on Evolutionary Computation*, May 2002.
- [32] S. Kumar. *Classification and Detection of Computer Intrusions*. PhD thesis, Purdue University, Purdue, IN, 1995.
- [33] W. Lee and S. Stolfo. Data mining approaches for intrusion detection. *Proceedings of the 7th USENIX Security Symposium*, San Antonio, TX, 1998.
- [34] J. Luo. Integrating fuzzy logic with data mining methods for intrusion detection. Master's thesis, Mississippi State Univ., 1999.
- [35] P. A. Porras and P. G. Neumann. EMERALD: event monitoring enabling responses to anomalous live disturbances. *1997 National Information Systems Security Conference*, oct 1997.
- [36] S. R. Snapp, J. Brentano, G. V. Dias, T. Goan, L. T. Heberlein, C.-L. Ho, K. N. Levitt, B. Mukherjee, D. L. Mansur, K. L. Pon, and S. E. Smaha. A system for distributed intrusion detection. *Proceedings of Third International IEEE COMPCON*, pages 170–176, March 1991.
- [37] W. H. Ware. Controls for computer systems: Report of defense science board task force on computer security. Technical Report R609-1, Rand Corporation, 1970.

APPENDIX

This appendix provides an example of the format of detectors using the generic gene format discussed in chapter 4. Each attribute of a detector matches some continuous subset of the possible values. Each attribute also optionally matches a nonexistent attribute. Table A.2 shows an example detector when using the genes listed in table A.1. In order for a detector to match, all attributes must match. This example detector will match any session that includes traffic on port 80 (0x0050) and a source address numerically greater than 127.0.0.1. In this example, the session may include any destination addresses, or even no addresses at all.

Table A.1: Example Genes

Gene	Domain
Source Address	$[0, 2^{32})$
Destination Address	$[0, 2^{32})$
Port	$[0, 2^{16})$

Table A.2: Example Detector

Attribute	Matching Values
Source Address	[0x7F000002, 0xFFFFFFFF]
Destination Address	Empty or [0x00000000, 0xFFFFFFFF]
Port	[0x0050, 0x0050]