Revisiting LISYS: Parameters and Normal Behavior

Justin Balthrop, Stephanie Forrest, and Matthew R. Glickman
Computer Science Deptartment
University of New Mexico
Albuquerque, NM 87131
artis@cs.unm.edu

Abstract - This paper studies a simplified form of LISYS, an artificial immune system for network intrusion detection. The paper describes results based on a new, more controlled data set than that used for earlier studies. The paper also looks at which parameters appear most important for minimizing false positives, as well as the trade-offs and relationships among parameter settings.

I. INTRODUCTION

A growing body of work explores the use of immunologically inspired methods to address the problem of network intrusion detection [8, 19, 12]. Although this work has not yet produced industrial-strength network intrusion detection systems (IDS), it has provided a convenient common framework for exploring and comparing various immune-inspired algorithms. Different experiments by different groups have achieved markedly different results on this problem, presumably because of differences among the models, algorithms, and data sets. We view this diversity of formalisms and experimental test-beds as positive, in the sense that at this early stage it is important to explore a family of immune-inspired algorithms rather than standardizing too early on arbitrary choices. Nevertheless, it is important to understand how, why, and under what circumstances various methods perform well or badly and to account for discrepancies in reported results.

This paper addresses some of these issues, in the context of LISYS, Hofmeyr's artificial immune system (IS) framework applied to network intrusion detection. We report results on a new small data set for network intrusion detection, which is intended to capture network traffic similar to that of a home networking environment or small corporate intranet. We study how LISYS characterizes the normal behavior of a network, de-emphasizing the question of how well it detects a wide range of intrusions. It might seem odd for a paper on intrusion detection to de-emphasize intrusions, but in our experience, the success of intrusion detection systems depends heavily on their false-positive rates, and these can be studied largely independently of real attacks. Also, many of the immune system abstractions that were used in LISYS have the effect of controlling false positive rates, so a detailed look at how the system represents normal behavior and controls false positives is warranted. However, we do report data on LISYS' performance under some attack conditions.

II. OVERVIEW OF LISYS

In this paper, we explore several of the mechanisms that LISYS uses to control false positives. In order to get a clear view of these mechanisms, we use a simplified version of LISYS. In this section we summarize briefly the basic features that are a part of this simplified LISYS.

LISYS is situated in a local-area network (LAN) and used to protect the LAN from network-based attacks. In this domain, *self* is defined to be the set of normal pairwise TCP/IP connections between computers, and *non-self* is the set of connections, potentially an enormous number, which are not normally observed on the LAN. A connection can occur between any two computers in the LAN as well as between a computer in the LAN and an external computer. It is defined in terms of its "data-path triple"—the source IP address, the destination IP address, and the port by which the computers communicate [14, 6]. In LISYS, the connection information is compressed in two ways to form a single 49-bit string [7]. First, it is assumed that one of the IP addresses is always internal, so only the final byte of this address needs to be stored. The port number is also compressed from 16 bits to 8 bits by re-mapping the ports into several different classes.

LISYS consists of a set of *detectors*. Each detector is a 49-bit string and a small amount of local state. A perfect *match* between a detector and a compressed SYN packet means that at each location in the 49-bit string, the symbols are identical. However, perfect matching is rare in the immune system and improbable between strings of any significant length, so LISYS uses a partial matching rule known as r-contiguous bits matching [15]. Under this rule, two strings match if they are identical in at least r contiguous locations.

LISYS uses *negative detection* in the sense that valid detectors are those that fail to match the normally occurring connections in the network. Detectors are generated randomly and then detectors that match connections observed in the network during the *tolerization period* are eliminated. Detectors also have a fixed probability of dying randomly at each time step. The finite lifetime of detectors, when combined with detector re-generation and tolerization, results in *rolling coverage* of the self set. For the r-contiguous bits matching rule and fixed self sets which don't change over time, randomly generating detectors is inefficient. More efficient algorithms based on dynamic programming methods allow us to generate detectors in linear time [3, 2, 17, 16, 18]. However, when generating detectors asynchronously for a dynamic self set, such as the network intrusion detection setting, we have found that random generation works well.

LISYS also uses *activation thresholds*. Each detector must match multiple packets before it is activated. Each detector records the number of times it matches (the *match count*) and raises an alarm only when the number of matches exceeds the activation threshold. Once a detector has raised an alarm, it returns its match count to zero. This mechanism also has a time horizon: Over time the match count slowly returns to zero. Thus, only repeated occurrences of structurally similar and temporally clustered strings will trigger the detection system.

The original LISYS uses a number of other mechanisms not mentioned here, including *distributed detectors*, *sensitivity levels*, *permutation masks*, *memory detectors* and *costimultation*. For more details on full LISYS, the reader is referred to [7, 8]. For the remainder of this paper, the term LISYS will refer to the simplified version described above. Although we collected the data on-line in a production environment, we performed our analysis off-line on a single computer. This made it possible to compare performance across many different parameter values.[1]

III. RELATED WORK

At least two other IS-based network IDSs have been described in the literature [10, 11, 19], as well as an architecture proposal that targets IS-based intrusion detection at both the host and network levels [1].

Kim and Bentley describe an architecture and report experimental results using some of the their system's key components [10, 11, 12]. Some commonalities with LISYS include distribution of detection among multiple hosts, the use of memory detectors and a mechanism for lowering the system's anomaly threshold as detector match rates increase. Unlike LISYS, however, detector generation is centralized at a single node, and the decision about whether to signal an anomaly relies on communication between nodes. Another difference is the method of detector generation. While LISYS produces detectors via a simple process of random generation in conjunction with filtering via negative selection, Kim and Bentley describe a more extensive process involving a genetic algorithm, niching, and feedback from matched detectors to boost the diversity and utility of detectors. Kim and Bentley examine all TCP packets and group them according to the connection with which they are associated, thus considering a wider range of data than LISYS. This increased scope comes at the cost of a significantly more complicated representation. Kim and Bentley report difficulties with the negative selection algorithm (see Section VII).

[1]The programs used to generate the results in this paper are available from http://www.cs.unm.edu/∼immsec. The programs are part of the LISYS package and are found in the LisysSim directory.

A second instance of an IS-based network IDS, known as CDIS (Computer Defense Immune System), is described by Williams et al [19]. They report very positive experimental results for this system, which also uses negative selection. Like LISYS, CDIS processes the data-stream at the packet level; however, rather than restricting attention to TCP SYN packets, CDIS examines all packets for three different protocols: TCP, UDP, and ICMP. The number of fields represented is significantly larger than is used by LISYS, but in order to accomplish imperfect matching, detectors in CDIS do not have to specify values for every field. CDIS includes some other interesting features, including *affinity maturation* to maximize the generality of detectors and *costimulation* between detectors.

IV. DATA SET

One challenge in intrusion detection is finding good data sets for experiments and testing. A commonly used data set is the one from Lincoln Laboratories (LL). This is the data set used with CDIS in [19]. Some of the weaknesses of the LL data set are pointed out in [13]. The chief criticisms are that it was not generated by actual users, that it is "too clean" (not enough noise), and that it is well known where the legitimate attacks occur. Data from the 'Internet Exploration Shootout' (IES) is used by Kim and Bentley in [11, 12]. There is a high volume of traffic in this data set, but it only takes place over a period of about 16 minutes, not enough time to reasonably characterize normal behavior, in our opinion. Because of these factors, we decided not to use either of these data sets. Although we have access to the data set used in the original LISYS studies, we decided to collect a new one, both to see how LISYS performed on a different data set and because the original data set is now several years old.

Our objective was to control the data set as much as possible while still collecting data in a realistic context. We chose to collect data from an internal restricted network of computers in our research group at UNM. The six internal computers in this network connected to the Internet through a single Linux machine that acted as a firewall, router and masquerading server. This network provided a data set that satisfies both of our objectives. The internal restricted network is much more controlled than the the external university network. In this environment, we can understand all of the connections, and we can limit attacks. Moreover, this environment is realistic. Many corporations have intranets where activity is somewhat restricted and external connections must pass through a firewall. This environment could also model the increasingly common home network that connects to the Internet through a cable or DSL modem and has a single external IP address. Attacks are a reality in these environments, and we designed attack scenarios that correspond to likely occurrences in this class of environment.

The normal network data consisted 22,329 TCP SYN packets collected over the course of two weeks in November, 2001. Thus, there was an average of about 1600 packets per day during this period. Because we are trying to capture the network activity of local users, we determined that two weeks was the shortest period of time that would give us an adequate picture of self. In [7], network connections to web servers are removed by filtering out all connections to port 80. We did something similar, but instead of completely removing all web connections, we simulated the use of a proxy server. All outgoing connections to port 80 or 443 were re-mapped to port 3128 on the proxy machine. This produced a similar effect to that of the web proxy cache SQUID.

V. Representing Normal Behavior

In this section, we study the effect of different components of LISYS on its ability to capture normal behavior. These include: the specificity of detection (the r parameter), the number of detectors, the tolerization period, and the activation threshold. The following subsections explore these issues.

A. The r parameter and number of detectors

The value r is a threshold which determines the specificity of detectors, in that it controls how many strings can potentially be matched by a single detector. For example, if $r = l$ (49 in our case), the match is maximally specific, and the detector can match only a single string—itself. As shown in [4], the number of strings a detector matches increases exponentially as the value of r decreases.

More specific detectors match fewer strings and thus can distinguish more precisely between observed self and everything else. Additionally, as the specificity of detectors increases, the number of detectors required to achieve the same level of coverage increases (see [5] for probabilistic estimates). So in order to minimize the number of detectors, the trick is to find the smallest r which still provides reasonable discrimination. This specific value is clearly dependent upon the particular data-stream being monitored. For our data set, figure 1 plots the number of detectors generated against the number of unique matches between detectors and strings outside the training set.

To make this graph, we divided the data into two sets: the first 15,000 normal packets were the *training set*, and we combined the remaining normal packets with the attack data (see section VI) to make the *test set*. Mapping the packets into the 49-bit representation yields a total of 131 unique strings in the training set, and 557 strings in the test set.[2] Negative selection was performed by randomly generating detectors and discarding all those that matched at least one string in the training set. We continued this process until a very large number of detectors (10,000 – 100,000) were accumulated for each of the three

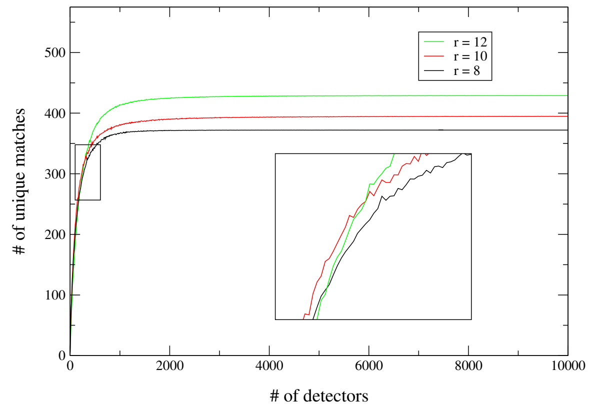

Fig. 1. The average number of unique matches as a function of detector set size for r =8, 10, and 12.

values of r. Each of the detectors was then compared to each string in the test set, and the strings that matched were recorded for each detector. Then the curves were derived by randomly selecting sample sets of detectors of each size from 1 to 10,000 and calculating the mean number of unique matches against the test set. The number of samples was doubled until the difference between successive means dropped below 1%.

Due to the rules of negative selection, none of the 81 strings in common between the training and test sets could ever be matched by a detector. Thus, an upper bound on the number of unique matches that might be achieved by a set of detectors is $557 - 81 = 476$. Figure 1 shows (as expected) that larger values of r correspond to more extensive coverage (i.e. a larger number of matches). With $r = 12$, a maximum of approximately 429 matches (90% of the upper bound of 476) is achieved. Although it is generally true that a larger value of r provides greater coverage given enough detectors, it is also the case that the lower the threshold r, the larger the number of strings that can be matched by a single detector. Thus, with small detector sets, smaller values of r will yield greater coverage. An instance of this phenomenon is illustrated by the inset of figure 1, which is an expanded view of the transition points where the detector set size grows large enough that $r = 12$ first begins to provide greater coverage than $r = 8$ and $r = 10$.

At some point, adding detectors no longer adds significant new coverage because new detectors are simply covering the same space as old detectors. We refer to this point as the saturation point. From the figure, it can be seen that the number of detectors at the saturation point is higher for $r = 12$ than $r = 10$, and likewise it is higher for $r = 10$ than $r = 8$. However, near-maximal coverage is achieved for all three r values using a relatively small number of detectors (between about 1,000 and 2,000).

[2]Each group in the test set had only unique strings, but there were some repeats among groups.

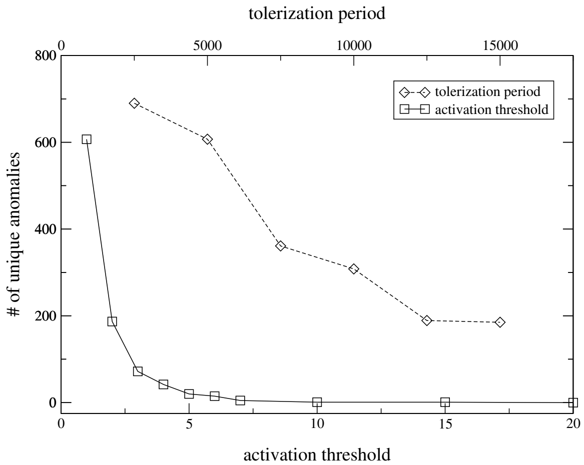

Fig. 2. The number of anomalies signaled in the normal data set as a function of tolerization period and activation threshold.

B. Tolerization Period

We expect, in general, that increasing the tolerization period, will reduce the number of false positives, because as the tolerization period becomes longer, detectors are exposed to a larger and presumably more representative sample of self. To test our expectations, we studied the effect of changing the tolerization period on the number of false positives. These experiments were run with $r = 12$.

In order to compare detector performance on exactly the same set of packets, the results shown in figure 2 were produced by co-varying the start point in the data set with the tolerization period such that the point at which the first detectors become mature is the same (after $15,000$ packets) for all tolerization periods. Thus, while runs using a tolerization period of $15,000$ were started from the beginning of the data set, runs using a tolerization period of $5,000$ were started after skipping the first $10,000$ packets.[3] Interestingly, the relationship between tolerization period and false positives (number of unique anomalies found in normal data) is roughly linear. We do not yet have a theoretical derivation of this result.

C. Activation thresholds

Activation thresholds are a mechanism explicitly designed to reduce false positives. The motivation for activation thresholds is based on the observation that anomalies from intrusion attempts tend to be temporally clustered, while anomalies from false positives tend to occur at a relatively constant rate. When using activation thresholds, if a detector matches only infrequently, its match count will likely decay significantly between matches and never reach the threshold. If, however, a detector matches packets in rapid succession, its match count is likely to increase more quickly than the decay rate. The magnitude of a detector's match count thus serves as an indicator of its *recent* match rate. Intrusion attempts which generate temporally clustered spikes matching a single detector, are likely to produce match counts that will exceed the activation threshold, and an anomaly will be signalled.

Figure 2 illustrates how the number of false positives falls off as the activation threshold increases. These experiments were run with $r = 12$ and a *tolerization period* of 15,000. For this data set, an activation threshold of 10 appears to minimize the number of detected anomalies.

[3]The tolerization periods are expressed in number of packets. The number of packets per day varies, so these numbers don't correspond to an exact period of time, but 5000 packets is roughly 3 days.

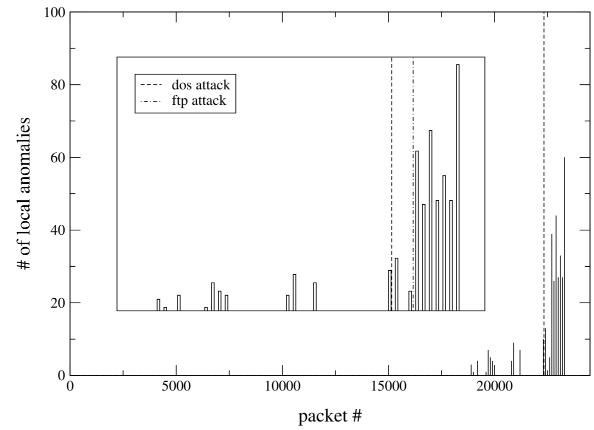

Fig. 3. Local anomalies for attack group one.

VI. DETECTING ABNORMAL BEHAVIOR

In order to verify that the process of reducing false positives didn't decrease LISYS' ability to detect true positives, we performed several attacks using the free security scanner Nessus. The attacks took place a week after the normal period ended and consisted of 76,179 TCP SYN packets over the course of two days. All of the attacks, with the exception of the denial of service attack were performed from a laptop which was assigned a dynamic IP address because it had a physical connection to the internal network. We tested to see if LISYS could detect the attacks by running LISYS on a data set that consisted of the normal data (training and test) followed by the data from one of three attack groups. Based on the experiments in the previous section, we chose the following parameters for LISYS: $r = 10$, $tolerization\ period = 15000$ and $activation\ threshold = 10$.

The first attack group consisted of five attacks:

- A denial of service (DOS) attack from an internal computer to an external computer.
- A firewall attack against the router machine.
- An ftp attack against an internal ftp server.
- SSH probes against several internal machines.
- An attack that probes for services like chargen and telnet. Nessus referrs to these as "useless services".

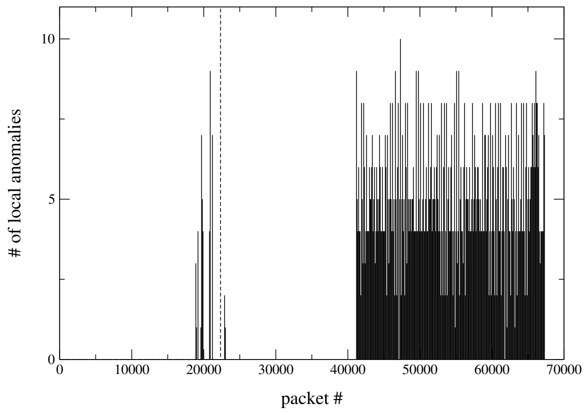

Fig. 4. Local anomalies for attack group two.

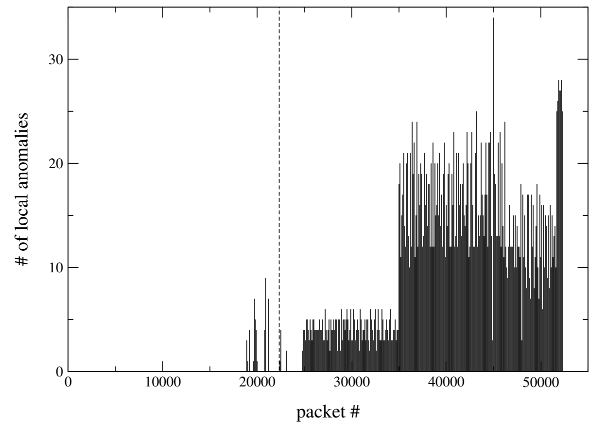

Fig. 5. Local anomalies for attack group three.

The second attack group consisted of a TCP SYN scan followed by an nmap tcp connect() scan. Both of these scans are port scans, but a SYN scan is a stealth scan, while a tcp connect() scan is noisy. The third attack group consisted of a full nmap port scan.

Figure 3 shows the LISYS anomaly data for attack group one. Local anomalies are displayed using windows of 100. This means that each bar is an indication of the number of anomalies that occurred in the last 100 packets. We can see from this graph that LISYS was able to detect the denial of service attack and the ftp attack. The vertical lines indicate the beginning of these two attacks, and there are spikes shortly after these attacks began. The spikes for the ftp attack are significantly higher than those for the DOS attack, but both attacks have spikes that are higher than the spikes in the normal data, indicating a clear separation between true and false positives. This view is interesting because the height of the spikes can be interpreted as the system's confidence that there is an anomaly occurring at that point in time.

Figure 4 shows the LISYS anomaly data for attack group two. By looking at this figure, we can see that there is something anomalous in the last half of the attack data, but LISYS was unable the find anything wrong in the first half of the attack data. Although the spikes are roughly the same height as the spikes in the normal data, the temporal clustering of the spikes also indicates that there is something anomalous. Figure 5 shows the LISYS anomaly data for attack group three. The figure indicates that LISYS overwhelmingly found the nmap to be anomalous. Not only are the majority of the spikes significantly higher than the normal data spikes, but there is a huge number of temporally clustered spikes.

VII. KIM AND BENTLEY

Kim and Bentley considered negative detection using one representation and the r-contiguous matching rule [10, 11]. They represented network traffic as strings of length 33 defined over an alphabet of 10 values. Using the r-contiguous bits matching rule with a threshold of 4, they concluded that detector generation was impossible in a reasonable period of time. Although they fail to report how many retries were required on average to generate a valid detector, they do report that in one day they failed to generate a single valid detector. They then tried using a match threshold of 9. With this threshold, generating detectors became computationally feasible, but it was then very difficult for them to generate enough detectors to cover non-self adequately (see section V-A). From these two experiments, they concluded that the negative selection algorithm suffers from "severe scaling problems" and they "raise doubt whether this algorithm should be used for network intrusion detection." Results such as these serve to highlight the importance of choosing an appropriate representation for a given problem, an issue whose importance has long been recognized in other fields, such as genetic algorithms.

There are several explanations for their results, and the explanations provide interesting insights into the importance of representation. First, the difference in the size of coverage between an $r = 9$ detector and an $r = 4$ detector is huge. The matching threshold r is exponentially sensitive, so an $r = 4$ detector matches roughly 100,000 times as many strings as an $r = 9$ detector. When attempting to tune such a sensitive parameter, a jump from 4 to 9 hardly seems justifiable. A second factor, is their choice of representation, which makes r even more sensitive. An alphabet size of 2 means that decreasing r by 1 roughly doubles the number of strings that a detector can match, and hence roughly doubles the amount of coverage of non-self. With an alphabet size of 10, however, a decrease of r by 1 increases detector size by an order of magnitude.

Finally, Kim and Bentley's conclusion, that negative detection does not work because it cannot generate enough detectors to cover non-self in a reasonable amount of time, seems to be overstated. Showing that negative detection doesn't work for one representation with two match thresholds does not support such a strong conclusion. It is important here to remember that negative detection and the r-contiguous match rule are not the same thing. They are rather two pieces of a system, either of which could be replaced if determined to be hindering performance.

VIII. CONCLUSION

In this study, we introduced a new data set, which was intended to model some of the features of a corporate intranet or a small home networking environment. This data set is small and well controlled, yet it features "live" traffic with real users. We used the new data set to confirm many of the results published in the original papers on LISYS. In particular, we discovered that certain parameter settings that worked well in the original paper, also work well for our new data set. These include a tolerization period of 3-4 days and an activation threshold of 10. We discovered that a detector threshold of $r = 10$ performed better than the original published value of $r = 12$, although we don't completely understand the reason for this difference. In this paper, we also confirmed the importance of several LISYS mechanisms for controlling false positives, including tolerization periods, activation thresholds, and rolling coverage. Other mechanisms, not considered in this paper, include: sensitivity levels, co-stimulation, and permutation masks. Understanding the role of these additional mechanisms is an important avenue for future investigation. We also investigated certain negative results reported by Kim and Bentley and offered some explanations for them.

In this paper we have used the network intrusion detection problem as a device for exploring how various components of LISYS contribute to its performance. In order to use LISYS seriously as a network intrusion detection system, we believe that several extensions would be required. Probably the most important of these would be to modify the representation to look more deeply in the network stack. An informal survey of recent network-based intrusion methods suggests that a perfect intrusion-detection system which monitored only SYN packets would detect at most about 40% of the different kinds of common attacks [9].

However, the restriction to SYN packets in LISYS is not hard-wired, and the system could be extended to consider other aspects of network traffic. The advantages of restricting ourselves to SYN packets for this study were simplicity and efficiency. LISYS performs surprisingly well, even when limited to SYN packet traffic only. And, by only monitoring SYN packets, the system is much more efficient. It can monitor every SYN packet in the network, with modest computational requirements.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the support of the National Science Foundation (grants IRI-9711199, CDA-9503064, and ANIR-9986555), the Office of Naval Research (grant N00014-99-1-0417), the Defense Advanced Projects Agency (grant AGR F30602-00-2-0584), the Intel Corporation, and the Santa Fe Institute. Many people have contributed to LISYS and have offered helpful comments and advice over the past three years, including Steven Hofmeyr, Jason Stewart, Todd Kaplan, Hajime Inoue, Dennis Chao, Fernando Esponda, and Paul Helman.

References

[1] D. Dasgupta. Immunity-based intrusion detection system: A general framework. In *Proc. of the 22nd National Information Systems Security Conf.*, 1999.

[2] P. D'haeseleer. An immunological approach to change detection: theoretical results. In *Proc. of the 9th IEEE Comp. Security Foundations Workshop.*. IEEE Comp. Society Press, 1996.

[3] P. D'haeseleer, S. Forrest, and P. Helman. An immunological approach to change detection: algorithms, analysis and implications. In *Proceedings of the 1996 IEEE Symposium on Computer Security and Privacy.* IEEE Press, 1996.

[4] F. Esponda. Detector coverage with the r-contiguous bits matching rule. Tech. Report TR-CS-2002-03, UNM, 2002.

[5] S. Forrest, A. S. Perelson, L. Allen, and R. C. Kuri. Self-nonself discrimination in a computer. In *Proc. of the IEEE Symp. on Research in Security and Privacy.* IEEE Press, 1994.

[6] L. Heberlein, G. Dias, K. Levitte, B. Mukherjee, J. Wood, and D. Wolber. A network security monitor. In *Proc. of the IEEE Symp. on Security and Privacy.* IEE Press, 1990.

[7] S. Hofmeyr. *An immunological model of distributed detection and its application to computer security.* PhD ths., UNM, 1999.

[8] S. Hofmeyr and S. Forrest. Architecture for an artificial immune system. *Evolutionary Comp. Jrnl.*, 8(4):443–473, 2000.

[9] K. Ingham, 2001. Personal communication.

[10] J. Kim and P. Bentley. The artificial immune model for network intrusion detection. In *7th European Conf. on Intelligent Techniques and Soft Computing.* Aachen, Germany, 1999.

[11] J. Kim and P. Bentley. Negative selection and niching by an artificial immune system for network intrusion detection. In *GECCO-99 Proceedings*, p. 149–158, 1999.

[12] J. Kim and P. Bentley. An evaluation of negative seelection in an artificial immune system for network intrusion detection. In *GECC0-2001 Proceedings*, p. 1330–1337, 2001.

[13] J. McHugh. Testing intrusion detection systems: A critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory. In *ACM Transactions, November 2000*, 3(4):262–294, ACM, 2000.

[14] B. Mukherjee, L. T. Heberlein, and K. N. Levitt. Network intrusion detection. *IEEE Network*, pages 26–41, 1994.

[15] J. Percus, O. Percus, and A. Perelson. Predicting the size of the antibody combining region from consideration of efficient self/non-self discrimination. *Proc. of the National Academy of Science*, 90:1691–1695, 1993.

[16] S. T. Wierzchon. Discriminative power of the receptors activated by k-contiguous bits rule. *Journal of Computer Science and Technology*, 1(3):1–13, 2000.

[17] S. T. Wierzchon. Generating optimal repertoire of antibody strings in an artificial immune system. In *Intelligent Information Systems*, 119–133, 2000.

[18] S. T. Wierzchon. Deriving concise description of non-self patterns in an artificial immune system. In *New Learning Paradigm in Soft Computing*, 438–458, 2001.

[19] P. Williams, K. Anchor, J. Bebo, G. Gunsch, and G. Lamont. CDIS: Towards a computer immune system for detecting network intrusions. In *RAID 2001*, 2212:117–133, 2001.