Protecting Network Servers

Kenneth Ingham
Department of Computer Science
University of New Mexico
Albuquerque, NM 87131
ingham@i-pi.com

June 2, 2003

Contents

1 Introduction

1.1 The problem

Today, attacks against servers cost billions of dollars. For example, the Code Red worm cost 2.6 billion dollars in July and August 2001 [23]. Code Red is but one example of servers being attacked through the service they offer. To detect such attacks, network and system administrators make use of Intrusion Detection Systems (IDSs). As I will show later in this proposal, in spite of the fact that nearly all attacks are in the network application layer, few IDSs look there for attacks. My dissertation work will address this problem by identifying and implementing algorithms that are promising at the network application layer, identifying proper test data, and then testing the algorithms to see if they would be useful in real-world situations. In this section, I introduce the concepts and proposed research that I will cover in more detail throughout the rest of this proposal.

1.2 The current solutions

Administrators use firewalls and intrusion detection systems as their primary security solutions. However, as shown by Ingham and Forrest [122], firewalls do not protect against attacks aimed at the service provided by a server. Signature-based IDSs only protect against already-known attacks. With one exception [79], anomaly-detecting IDSs work at layers[1] other than the application layer[2]. This omission seems odd, since the attack arrives via the application layer. IDSs that track information such as network connection patterns cannot work well on a server expecting connections from all over the Internet—the first connection from an attacking host is likely to be the one carrying the attack. While some behavior-based IDSs (e.g., pH by Somayaji [225]) have prevented attacks, recognizing an attack before it arrives at the vulnerable program improves our ability to prevent damage.

One additional benefit of network application level intrusion detection is that this approach has the chance to intervene and prevent the intrusion by not allowing anomalous requests to be delivered to the server. The systems working at other ISO levels cannot do this because, for Internet servers, connections from novel addresses are normal[3].

[1] Layers from the International Organization for Standardization (ISO) reference model for networking.

[2] Current anomaly-detection IDSs use primarily the network and transport layers, although the data link layer may also be used

[3] Systems, e.g., *portsentry* (In attempting to find a reference for it I discovered that the corporate sponsor, Psionic, has been purchased by Cisco, and the web page for *portsentry* has disappeared.), do exist which block access if a remote host does something potentially hostile, for example a port scan. However, scans can be easily forged,

1.3 Importance of these problems

As mentioned above, attacks against servers are expensive for the victim. Beyond the Code Red costs mentioned, Nimda prevented hospital workers in Västra Götaland, Sweden from accessing reservations and computer-based medical records [135]. While no lives were lost this time, it shows that the costs could be more than simply monetary.

1.4 What is not known

My review of the literature in Section 2 shows:

- We do not know if network application layer anomaly detection will work.
- The literature is weak when it comes to the question, how do we compare intrusion detection algorithms?
- The theory of intrusion detection is in its infancy.

1.5 Research objectives

To address the problems identified in this section, my dissertation has the following objectives:

1. Identify and implement algorithms that are promising at the network application layer.
2. Identify proper data to use in the testing.
3. Implement and run a suite of evaluations for testing the algorithms.
4. If one or more algorithms work well, generalize the algorithms that work to explain why they work well. If none of the algorithms work well, use this information to answer the question, "What is it about the application layer that prevents all of these anomaly-detection algorithms from working?"
5. Identify and implement potential effector functions for protecting the server.

resulting in blocking access from non-hostile sites.

1.6 Anticipated results, benefits, and societal impact of this work

At the end of this research, we will know if application-level anomaly detection is a good idea (and if not, why not). Additionally, assuming that it works well, we will also have a prototype IDS which will improve the security of servers, with a resulting savings if the results are widely implemented.

1.7 Organization of the rest of the proposal

The rest of this proposal is organized as follows: A more in-depth discussion of intrusion detection, how IDSs have been compared, and what (little) theory exists about intrusion detection is in Section 2. Section 3 provides the details of my proposed research. Finally, Section 4 summarizes this proposal.

2 Background and related work

> History: A recording of mistakes we make so we shall know when we make them again.
>
> —Ambrose Bierce?[4]

This section reviews how network administrators protect servers today through the use of intrusion prevention and detection. It also reviews the existing literature on testing and comparing intrusion detection systems, and intrusion detection theory. After reading this review, you will see:

- systems using the network application layer for anomaly detection are rare
- papers which test or compare IDSs are rare, and well-done comparison papers are rarer
- there is little theoretical work on intrusion detection.

In order to make these claims, I performed a search on Information Service in Physics, Electrotechnology and Control (INSPEC) [123] using the following search:

```
(intrud* or intrusion* or anomal*) <near/2> detect*
```

[4] At `http://seclab.cs.ucdavis.edu/projects/history/summary.html`, they claim that this quote is from Ambrose Bierce, but I have yet to be able to verify this.

in the title, subject, or abstract of the paper. This search resulted in 2385 potential papers. I reviewed the titles of all of these, and for all that appeared relevant, I reviewed the abstract. I obtained copies of the most interesting papers. Additionally, I searched NEC ResearchIndex (CiteSeer) [195], where I reviewed 1022 papers.

2.1 Intrusion Detection Systems (IDSs)

When preparing defenses for computers, the two major approaches are prevention and detection of intrusions. Preventing intrusions is paramount. Methods of intrusion prevention include firewalls [122], good configuration practices (e.g., as described by Hatch et al. [101]), and systems which react dynamically to interrupt intrusions such as Somayaji's pH [225, 226]. It also is important to know when a system is under attack and if the defenses have failed. This knowledge comes from an intrusion-detection system.

As the intrusion detection literature is large, several people have tried to get a handle on it by publishing reviews of the literature, including: Allen [3], Barber [18], Biermann et al. [26], Bilar [27], Jones [132], Kemmerer [136], Kvarnström [134], Lunt [176, 175], Mukherjee et al. [193], Richards [214], Turkia [235], and Verwoerd and Hunt [240]. The best reviews are those which present an unbiased, thorough review of the literature, and/or provide a good taxonomy for describing different intrusion detection methods. Examples of such good reviews include those by Axelsson [11, 15], Debar [59, 60, 61], Alessandri [2] (who works or worked with Debar), and McHugh [185]. While not a review of research IDSs, Jackson [125] wrote an excellent in-depth survey of commercial products.

All current IDSs use one (or both) of the following overall approaches [11, 15, 59, 60, 61]:

Signature detection (also called misuse detection, or detection by appearance) A researcher studies an attack and identifies the characteristics that distinguish this attack from normal data or traffic. These characteristics are known as the *signature*, and the signature becomes part of a database of attack signatures. When the IDS sees the signature, it raises an alarm.

Signature detection requires an attack to be studied before it can be recognized. Systems protected by such a system are vulnerable to new attacks until the updated database is available. Additionally, signature-based systems suffer from false positives, especially if the system configuration or environment changes. Patton et al. [202] described this phenomena, which they call "squealing", and how intruders can negate the benefit of a signature-based IDS with carefully crafted false positives.

Anomaly detection (also called detection by behavior) A program studies normal traffic and/or data and generalizes the observed patterns in it. When traffic or data that does not match the generalized training data appear, the system raises an alarm.

Anomaly-detection systems often have problems with normal behavior generating false alarms because normal behavior has changed over time (called concept drift or normal drift) or because the behavior did not exist in the training data (called perpetual novelty).

In order to fulfill Objective 1 described in Section 1.5, I must know what algorithms[5] exist, and what their strong and weak points are. The following list describes the major algorithms, most of which are ways to model normal or abnormal behavior (depending on whether the IDS is using positive or negative detection).

Pattern matching All signature-based IDSs use some form of pattern matching to identify the strings that might be associated with an intrusion, be it a virus in a file or an attack against a web server such as Code Red. Most commercial IDSs use pattern matching, as does the most popular open-source network IDS, *snort*, described by Fyodor [92] and Roesch [215]. In many pattern-matching systems, a pattern is simply a string which is (hopefully) not found in any normal traffic. Some pattern-matching systems include other information, such as where to look for the string (either by an absolute location, or by specifying a location relative to the network protocol or file format).

For network IDSs, Graham [97] proposed pattern matching combined with protocol analysis to make the pattern match both faster and less prone to false positives. His experiments showed that a network IDS that did protocol analysis in addition to the pattern matching was both faster and more accurate than a simple pattern matching IDS. Kumar et al. used colored petri nets [46, 148, 149, 150, 151] to formalize a pattern matching language which is more expressive than regular expressions. These patterns are better than simple strings because they generalize the problem so that entire classes of attacks can be represented as one pattern. Kumar was not the only person to work on an "attack language". Other researchers who have worked on attack languages include Doyle et al. [74], Sekar et al. [220], and Vigna et al. [78, 241, 242, 243]. Kuri and Navarro [152] improved pattern matching searches for IDSs by a factor of up to 75-fold. These pattern matching methods that generalize the pattern all improve the ability of an IDS to recognize similar but never-before-seen attacks.

[5] Both signature and non-signature based.

The complement of describing attacks would be describing an organization's security policy—sometimes describing what is allowed is easier than describing what is not allowed. Paxson [204, 205] worked on policy languages for intrusion detection.

Rule modeling Instead of the policy and attack languages, where a human describes what is or is not normal, some IDSs learn by looking at one or more data sources and deriving rules which describe the majority of the data. In rule modeling, the system often generalizes the rules that have been discovered to allow similar actions to be considered normal. These systems also tend to generate large numbers of rules. For example, the rule base that Wisdom & Sense generated contained 10^4 to 10^6 rules [237]. The IDS notes whenever these rules are broken. Researchers who have used rule modeling for intrusion detection include Helmer et al. [109], Lee et al. [162, 163, 164, 165], Vaccaro and Liepins [168, 237], and Ye and Li [252].

Expert systems Researchers build an expert system with a domain of knowledge based on whatever data sources the experts deem appropriate. In some cases, the rules are determined by a rule-modeling system, so the line between expert systems and rule modeling is sometimes blurry. Once the system has been set up, data from the running system are fed to the expert system to identify suspicious activities. These activities may be complex and/or consist of multiple, related events. Reasoning about these events is a strength that expert systems bring to the field of intrusion detection.

Researchers who have used expert systems for intrusion detection include Denning [64, 65], Ilgun [118, 120], Tsudik and Summers [234], and Denault et al. [63]. IDSs using expert systems include DIDS [222, 223], part of EMERALD [169, 170, 198, 207, 208, 239], MIDAS [56], NIDES [6, 7, 8, 177], and NADIR [112, 126].

Descriptive statistics Statistical descriptions of normal are popular. In a statistical system, the IDS learns by looking at one or more data sources and calculating statistics deemed by a person or some other portion of the IDS to be significant. The statistical methods used may be simple frequency analysis, Bayesian belief networks, Hidden Markov models, or other measures. The IDS then notes deviations from these statistics. Statistical methods sometimes have the lowest false positive rate when identifying intrusions. For example, Warrender et al. [245] found that a Hidden Markov Model had the best detection rate for true positives while rejecting false positives, although it required more computer resources than the second best method which was almost as effective.

Researchers who have used statistics for intrusion detection include Bronstein et al. [29, 30], Darling and Shayman [54], DuMouchel [76, 77] Endler [80], Eskin et al. [81, 82], Flajolet et al. [89], Helman and Liepens [107], Jha et al. [129], Lane [159], Liepins and Vaccaro [168], Lundin and Jonsson [174], Michael and Ghosh [186, 187, 188], Scott [217, 218], Valdes and Skinner [238], Wu et al. [247], Ye et al. [248, 249, 250, 251, 252, 253], Yeung and Chow [254], and Zhang and Lee [256]. Other IDSs using this technique include Computerwatch [73], part of EMERALD [198, 238], Haystack [221], IDES [127], MIDAS [56], NIDES [6, 7, 177], NSM [103, 104], and USTAT [118, 119]. Kohout et al. [146] used fuzzy measures, of which probability is a special case.

Deterministic Finite Automata (DFA) The IDS looks at one or more data sources and generates a DFA describing the data. Another way of describing this type of system is that it is learning a (formal) language. When the IDS is watching the system, it notes strings which are not a part of the language that the DFA describes. When what is being modeled is described by a language (e.g., network protocols), this approach learns the subset of the actual protocol used by the computer(s) that the IDS is protecting.

Another use of DFAs is to describe an attack rather than what is normal. In this case, if the DFA ever reaches its accepting state, then an attack is suspected. These IDSs are a variant of signature-based systems, in that they require a human to identify the attack before a DFA is generated.

I believe that this approach (or the even better approach of learning a context-free grammar) warrants more research because the current literature is sparse and papers explaining why this approach would be bad do not exist.

Researchers who have used automata for intrusion detection include Bhargavan et al. [25], Marceau [180], and Michael and Ghosh [188].

Neural networks With neural nets, the researchers use one or more data sources to train a neural net to recognize normal behavior. The neural net then identifies behavior not matching its training experience. In other fields, neural networks have shown their ability to detect patterns. The hard problem here is determining an appropriate data representation.

Researchers who have used neural nets for intrusion detection include Cannady [31], Debar et al. [58], Draelos et al. [75], Endler [80], Ghosh et al. [93, 94], Lee and Heinbuch [161], Mukkamala et al. [194], Ryan et al. [216],

Genetic algorithms (GAs) Some researchers use GAs to find a description of normal. The idea is that a GA may be able to do a better job of finding the rules describing normal than a person can. Since GAs are well-known for their ability to search a large space, they show promise. As in neural networks, how to represent the input data is the major problem. Both Neri [196] and Crosbie and Spafford [47] have used GAs.

Support Vector Machines Mukkamala et al. [194] used a Support Vector Machine (SVM) to learn the training data from the 1998 DARPA/MIT Lincoln Labs IDS test [171]. The advantage that SVMs provide is that they are less prone to overfitting data, and as a result may do a better job of classifying attacks versus normal data. Mukkamala et al. found that their SVM performed better and was faster than a neural network (the SVM training time was 17.77 seconds versus the neural net training time of 18 minutes).

***n*-grams** *n*-grams have been used in information retrieval [182, 189], measuring document similarity [51, 84] or categorizing text [38] and similarly for a measure of normal in intrusion detection [116, 90]. Lookahead pairs are similar in effect to *n*-grams, but Somayaji [225] found that they were more efficient.

Some researchers have applied *n*-grams to intrusion detection comparing sequences of system calls, as was done by Hofmeyr et al. [115, 116], Jones and Li [130], Jones and Lin [131], Marceau [180], Michael [188], Somayaji [225] (he also compared using sequences with using lookahead pairs), Researchers working with other data include Forrest et al. [91] looked at file contents for change detection, and May [181] used *n*-grams with SNMP traffic.

Information retrieval Besides *n*-grams, other techniques from information retrieval (IR) for categorizing text and measuring the distance between different texts might be useful for categorizing data as being close to or far from normal. In spite of this parallel, only a few researchers have tried it. Anderson and Khattak [9] and Lane [154, 155, 156, 157, 158, 159] are some of the few.

Multiple sensor fusion Axelsson [12, 13, 14] shows to avoid the the base rate fallacy[6] and hence generate too many false positives, an intrusion detection system must have a high accuracy (or low error rate). In order to deal with this problem, one approach that researchers have

[6] The base rate fallacy is when a detector has what appears as a good accuracy, but the rate of what it detects occurs at a low rate so the result is much poorer performance that might be expected. For example, on 1,000,000 samples, a detector which is 99% accurate for something that occurs once in 10,000, the detector produces a true positive only 1% of the time. See Axelsson's papers for more details.

tried is combining results from different intrusion sensors. If the sensors are independent, then this approach is valid and will work well. Researchers working with sensor fusion include Bass [20, 21, 22], Cuppens [49], Goan [95], and Valdes and Skinner [239]. Intrusion detection systems making use of sensor fusion include DIDS [223] and EMERALD [169, 170, 198, 207, 208, 239].

Immune system The immune system can be thought of as the goal IDS researchers are trying to reach through their research. The ability to recognize and destroy hostile entities (non-self) while not attacking the organism (self) is truly the holy grail of intrusion detection. Researchers including Dasgupta [55], Forrest et al. [91], Harmer et al. [100], Hofmeyr [113, 114, 115] (his IDS was known as LISYS), Kim (sometimes with Bentley) [140, 141, 142, 143, 144], Paula, Reis, et al. [203, 213], Somayaji et al. [225, 227], and Williams et al. [246] have all taken parts of the immune system as their inspiration for their algorithms for intrusion detection.

One of the major ideas from the immune system is negative detection. In the vertebrate immune system, lymphocytes have detectors for molecules or peptide fragments[7] that are not associated with the individual (i.e., foreign). Taking a cue from this portion of biology, systems making use of negative detection generate detectors for data that do not occur in the normal data stream. When one of the detectors matches, an anomaly is presumed to exist. Negative detection has the advantage that it may be easily distributed across all the machines on a network. Forrest et al. [91] used negative detection for virus detection, and Hofmeyr [114] used it for network traffic pattern anomaly detection.

Another immunologically-inspired idea is that of tolerization. When the IDS generates a detector, it is considered naive. If a naive detector matches a string early in its lifetime, the detector is considered to be matching "self", and it will be deleted. If a detector survives this tolerization period, it then is considered mature. When a mature detector matches a string in a request, the request is considered potentially anomalous.

In the immune system, co-stimulation is required before an immune response is generated. In LISYS, a mature detector which matches a request will be deleted unless it receives co-stimulation. This idea is related to that of sensor fusion, although in the case of LISYS, the second sensor is a human.

[7] Peptide fragments are pieces of molecules used by a cell in its normal operations. Some of these molecules are broken into pieces and displayed on the outside of the cell by class I major histocompatibility complex (MHC) molecules.

Agents Instead of simple sensors, some researchers use agents, which may be mobile amongst hosts on a network. Information from the agents is combined, providing a benefit similar to that with sensor fusion. The agents may use any of the algorithms discussed elsewhere in this section. The unique approach offered by agents is the mobility and overall system architecture rather than novel algorithms. Researchers working with agent-based intrusion detection include the group at CERIAS [16, 17, 48, 229, 255], Asaka et al. [10], Carver et al. [32], Ragsdale et al. [212], Helmer et al. [108], Huang et al. [117], and Jha and Hassan [128].

Graphs Communication between machines can be represented as a graph. Under the hypothesis that the graphs for normal and abnormal communication are different, if you can find a subgraph that represents an attack in the graph of current network communication, you have discovered an intrusion. This approach works well at discovering probes sweeping through a network and worms attacking other machines. Tölle and Niggermann [233] have worked with this approach; the IDS GrIDS [34, 232] is also designed around this idea. Both of these groups use a hierarchical view of the network in order to keep the problem tractable for larger networks. Communication on a local-area network (LAN) is reviewed by a local monitor, while a separate monitor watches the traffic between LANs. This approach is a good one, because it scales in a manner similar to the networks they are monitoring.

Because we have no theory of intrusion detection[8], we have do not know for sure what should work well. Instead, researchers are picking ideas based on their prior experience and background. The researchers all tested their algorithms, but usually in isolation. As will be shown in Section 2.2, few of these algorithms have been compared to each other. Additionally, the variety of IDS algorithms described above does not lead, however, to the same variety in data sources. The host-based data sources that these anomaly detection systems use with some example systems are:

- Kernel audit data, such as from the Solaris Basic Security Module (BSM) (Wisdom & Sense [237], IDES [64, 65, 127], NADIR [126], NIDES [6, 7, 8, 177], USTAT [119], the IDS built by Lee [164], and the IDS built by Ko et al. [145])
- System or library calls. (Forrest et al. [90], Marceau [180], Michael [188], Snyder [224], and Somayaji's pH [225, 226]. Jones et al. [131] used library calls instead of system calls because they claimed they are more application-oriented.)

[8] See Section 2.3 for the review of intrusion detection theory.

- File contents (for change detection) (Forrest et al. [91] and Kim and Spafford [137, 138, 139])

The network-based data sources these anomaly detection systems use and some example systems that use them are:

- IP headers (Mahoney and Chan [179])
- TCP connection patterns (Hofmeyr's LISYS [114]) or, more general, network traffic patterns (NTOP [66, 67, 68, 69, 70], NSM [104], Lee's IDS [164], GrIDS [34, 232])

An interesting omission is that none of the anomaly detection systems make use of the data at the network application layer (from the ISO seven-layer model of networking [124]). The closest researchers have come is:

- Williams et al. [246] listed using "packet payloads" as future work. The only followup paper published so far [100], however, does not go in this direction, but instead is an immunological approach to anti-virus and network attack detection.
- Another example of a non-signature system that looks at the network application layer is the commercial package SecureIIS™[79], which enforces a tighter standard for web requests than what is specified in the standards document [87]. SecureIIS only detects anomalies that have been determined by a human as being important.
- Almgren and Lindqvist [5] used an Apache module to collect the data they used for their IDS. In this case, they are not looking at the network, but instead they have a module in the web server that gets a look at the request before it is acted upon by the rest of the web server. It also allows them to see how the web server is reacting to the request.
- Finally, not quite the network application layer, Almgren et al. [4] looked through the log of requests for potential attack signatures.

The network application layer carries most intrusion attempts, yet few have looked there and nobody has tried a general anomaly detection system there[9]. This is a major hole in the field.

[9] In Section 3.1, I further justify why anomaly detection at the network application layer is a good idea.

2.2 IDS testing

Two reasons for testing IDSs exist:

- to verify that an algorithm works.
- to compare two or more algorithms to determine which is better under certain circumstances.

Most researchers test algorithms to claim that a particular algorithm works. This testing is frequently little more than asking, "Does the IDS detect an attack?". Slightly better are researchers who ask the question, "Which of the following attacks can the IDS detect?". Even this testing is often acknowledged as weak, as shown in the following quotes:

> Initial testing shows the algorithm performs satisfactorily.
>
> Preliminary experiments prove...
>
> The preliminary experiment results show the effectiveness of our system.
>
> A realistic UNIX system is much more complex than the one that we have modeled in this paper.

When researchers offer up a comparison between two or more different IDSs, some of these comparisons are in actuality only to justify why the author(s) approach is the "best", and not a real comparison at all. In other cases, the authors present a "straw man" argument, where the algorithm chosen for comparison is unlikely to work well in any production environment. These types of tests show that their IDS is better than an incompetent, or incompetently implemented, algorithm—does this really show that their algorithm is good though?

One place where side-by-side comparisons of multiple IDSs can be found is in the trade literature, where commercial IDSs are compared amongst each other with the intent of finding the "best" product. These types of comparisons are not limited to the trade literature, as sometimes they make it into peer-reviewed journals or conferences. Examples of commercial IDS comparisons include Barber [18], Newman et al. [199], Rae and Ludlow [211], and Richards [214].

As we can see, careful testing IDSs is rare. Several possible explanations for the scarcity of good IDS testing can be made:

- According to Debar, a set of criteria for evaluating an IDS does not currently exist [62].
- Identifying data to use is a difficult problem—the data must be representative of realistic operating conditions. Data collected live from a network may be subject to privacy concerns. Synthetic data (and some live data from departmental networks) must be shown to accurately represent real data on a target network.

- In order to test an intrusion detection system, researchers need a collection of intrusions and vulnerable machines on which to test these intrusions. Because a library of intrusions is similar to a cache of conventional weapons, researchers often use disconnected networks for their testing to ensure that the hostile code does not escape into unprotected networks. Setting up a good, protected network is resource-intensive, both in the costs of the hardware, as well as the costs of the people to set up and maintain the diversity of machines needed to ensure a good test environment.

  Maintaining a collection of vulnerable machines is also difficult. Bugs are discovered as systems evolve. The result is that exploits are often specific to a given operating system (OS) distribution and version, as well as the versions of compilers, libraries and other software installed. Since a given machine (or virtual machine) can only run one version of the operating system at a time, researchers need substantial resources to maintain this collection of vulnerable machines.

- Many intrusions are fragile; if any part of the environment for which the intrusion was written is not as expected, the intrusion is likely to fail. Either nothing bad happens (e.g., only a log entry indicating that something odd occurred), or the intrusion attempt turns into a denial-of-service attack. This fragility means that having a good collection of attacks requires work to ensure that they function in the test environment.

These difficulties should not be considered excuses, as some researchers *have* done a good job at comparing IDSs, as I will show below.

In order to test multiple IDSs, one of the requirements is that the data and environment be reproducible. A framework for testing is one way of achieving this reproducibility by providing a setup where different IDSs can be tested under identical conditions. Four researchers or research groups have set up frameworks for testing:

- The first published papers about an IDS testing framework and methodology were from Puketza et al. [209, 210] at UC Davis. Unless they failed to publish further work, they built their framework and then tested only one IDS (NSM [103, 104]).

- Wan and Yang [244] set up a framework for testing sensors that utilize the Internet Engineering Task Force (IETF) Intrusion Detection Working Group (IDWG) Intrusion Detection Message Exchange Format (IDMEF) [50]. Their framework may be useful, but the paper just describes a preliminary framework.

- The most famous IDS test framework was developed for the DARPA/MIT Lincoln labs IDS testing [98, 99, 172, 173]. This test framework consisted of networks of machines, some of which are attackers, some victims, and some exist simply for traffic generation to make the detection task more difficult. However, McHugh [183] points out that the DARPA/MIT Lincoln Laboratories IDS test used generated data, but the MIT researchers never did any tests to show that the generated data was representative of real data. Additionally, they did no tests to verify that their attacks were representative of real attacks.

- IBM Zurich [62] set up a laboratory for testing IDSs. Their normal data came not only from recordings of user sessions, but also from the IBM test suites for the AIX operating system. While this test suite is not representative of actual user interactions, it is a source of what could be normal. It is also good for testing the false alarm rate, because this test suite exercises parts of the code that are rarely exercised in normal usage.

Once a research group has developed a framework for testing, you might think that they would then turn out many good tests of IDSs. However, this is not the case. Also, some researchers without published papers about frameworks have produced good IDS testing. I found four sets of researchers who have done good work comparing IDSs or their algorithms[10]. I define "good work" as being:

- controlled

- repeatable

- comparing two or more approaches that are all claimed as valid (i.e., not using a straw man argument).

Jha, Tan, and Maxion [129] were primarily presenting their IDS based on Markov chains. However, in order to test their algorithm, they generated a set of test metrics which more thoroughly tested their IDS than any other paper about a specific IDS.

Warrender et al. [245] implemented four different algorithms, ran them on the same data, and evaluated their effectiveness at detecting intrusions.

Lane [159] compared statistical and learning models of user behavior for intrusion detection.

The most famous IDS tests are the DARPA/MIT Lincoln Laboratories IDS tests of 1998 and 1999 [99, 98, 172, 173]. Using their framework described above, they collected data from victim

[10]It is a sad commentary on the intrusion detection field that out of the thousands of papers I reviewed, only four groups had done a good job of comparing IDSs.

machines under attack. This data has since been used by other researchers as a test dataset for their IDS ideas, because they can see how their IDS would have fared, had it been in the original Lincoln Laboratories test.

To researchers participating in the test, they provided training data with attacks identified. They also provided test data containing both old (i.e., identified in the training data) and new attacks. The participating researchers provided information from their IDS about the attacks that they detected. The MIT people then analyzed and published the results. This IDS test is not without its critics. McHugh [183, 184] noted many problems with the test setup, data, and analysis.

Similar to the MIT data, the system-call data generated by Forrest et al. [90] at the University of New Mexico (UNM) has been used by other researchers (e.g., Lee [162]). This reuse of data (both from MIT and UNM) allows a comparison between methods, but in order to make the comparison, a researcher must collect all of the relevant papers and then build the comparison.

Finally, the researchers at IBM Zurich did a thorough job of comparing commercial IDS products. They set up a lab with many commercial IDSs installed. The results included Jackson's survey [125]. However, they produced more than just a survey. Alessandri [1, 2] looked at how an IDS worked, with the goal of being able to predict whether or not it would even be able to detect an attack (assuming a signature existed for the signature-based IDSs). Their goal was a more theoretical coverage of IDSs, and they advanced the state-of-the-art. They were one of few researchers who have looked at any kind of broad theory associated with intrusion detection.

2.3 Theory

As mentioned above, few researchers have worked on a theory of intrusion detection. This section contains a summary of the work by those who have.

Axelsson [12, 13, 14] described how the base rate fallacy[11] applies to intrusion detection, and that an intrusion detection system must be very accurate to avoid producing many more false alarms than true positives.

Vigna and Kemmerer [242, 243] set up a formal model of a network as a finite state automaton with rules describing when the machine could transition to a subsequent state. If the automaton ever reached the accepting state, an intrusion was noted. This theory is limited in that it requires a person to identify what constitutes an intrusion before the corresponding automaton can be built.

Lee and Xiang [166] looked at the entropy of IDSs. This information-theoretic approach allows them to predict whether an algorithm will be suited for use with a specific set of data. It also

[11] See the footnote on page 10 for a description of the base rate fallacy.

allowed them to predict a window size to use when looking at the data.

Julisch [133] looked at the problem of false alarms overwhelming human operators. He clustered alarms to identify the root cause and was successful, reducing the future alarm load by 82%. The theoretical aspect of his paper was to prove that general alarm clustering is $\mathcal{NP}$-complete.

Forrest et al. [91] proposed an immunological approach to computer security using negative detection—in the example presented in the paper, they generated a set of strings which did not appear in files on a system. If these strings showed up in a later scan, an anomaly was noted. Their analysis included a formal look at the relationships between:

- the probability of detection
- the number of detectors needed
- the detector length.

This work was followed by D'haeseleer et al. [72, 71]. They further investigated the information theoretic implications of this work, and explored the existence of "holes", non-detectable strings in the non-self set. Esponda et al. [83] continued this work by

- giving a formal framework for analyzing the tradeoffs between positive and negative detection schemes,
- introducing a new matching rule, r-chunks,
- showing that permutation masks reduce the number of holes, and
- showed how many permutations are required to minimize the holes.

Many people working on graduate degrees in computer science have some theory in their theses or dissertations. Some examples include:

- Hofmeyr [114] provides an analysis of the negative detection system and its the data representation that he developed. This analysis showed the number of detectors he needed for coverage of the space to be as complete as possible. His analysis also looked at "holes", or gaps in coverage, and the tradeoffs associated with closing these holes. He also analyzed the convergence properties of his data with a stochastic model; this analysis is also useful for understanding the perpetual novelty.
- Lane [159] analyzed the concept drift that occurs in most intrusion detection data sets. Concept drift occurs when users learn new ways of solving problems, software on computers is updated, etc. The result is that normal behavior changes.

- Lee [162] provided a formal analysis of the data-mining techniques he used for Intrusion Detection. Without such a discussion, one would wonder how well such an approach can do. His theory supports the potential for data mining to detect abnormal behavior.

- Somayaji [225] formally compared the two ideas of using sequences and of using lookahead pairs for his intrusion detection using system calls.

As you can see, nobody has tried to generate an overall theory of intrusion detection. Nobody has looked at whether intrusion detection is a solvable problem. For many of the algorithms listed above in Section 2.1, nobody knows if using them is appropriate from a theoretical standpoint. The theoretical side of intrusion detection still needs the most basic work. As you will see in Section 3.6, I will increase the anomaly-detection theory that exists.

2.4 Summary

This section contains a quick review of the algorithms used in intrusion detection. Given a collection of algorithms, one might ask how they compare to each other, and for this reason, this section covers how researchers have tested and compared IDSs and their algorithms. Finally, the little IDS theory that exists was reviewed.

From this survey of the prior art, several points are worth noting:

- anomaly detection systems do not make use of the data at the network application layer.

- papers about testing and comparing IDSs are rare, and papers where the testing was well done are even rarer.

- the theory of intrusion detection is in its infancy.

3 My research

> Man is a shrewd inventor, and is ever taking the hint of a new machine from his own structure, adapting some secret of his own anatomy in iron, wood, and leather, to some required function in the work of the world.
>
> —Ralph Waldo Emerson

This section contains details about my proposed dissertation work to identify and test algorithms that may work well at the network application layer. First, I justify why the application

layer in general is a good place for intrusion detection. Second, I justify why web servers are a good model for many different network application layer protocols. Next, in Sections 3.3–3.5, I work through the objectives identified in Section 1.5. Finally, this section of my proposal ends with a summary of the work I propose to do.

3.1 Justification for looking at the application layer

Several reasons exist why I believe that using the application layer is a good place for an anomaly detection system. First, as was shown in Section 2, few papers have been published about using this layer for anomaly detection.

Second, all network services that users employ directly make use of the network application layer. Many of these services have had security problems, including:

- *sshd* has recently had several problems, including remote root exploits [37, 42, 43].
- Several versions of *ftpd* have had security problems, including remote root exploits [39, 40, 41, 110].
- The most commonly attacked programs are web servers[12]. Microsoft's Internet Information Server (IIS) is legendary for the costs that have been incurred by successful attacks [53, 102]. Apache has a good record, but recently a remotely exploitable bug was discovered [36], followed by the publication of a worm which exploits this bug [190].

  Additionally, while Apache itself has had few problems, a common extension to Apache, PHP, has regularly had security problems in it or in code written in it [88, 111].

The application layer carries the attacks against these programs.

Thirdly, I believe that the problem of anomaly detection at this level is tractable. Many application-level protocols use a grammar and/or a DFA to describe the legal communication. For the more complex protocols, the part of the grammar that a server actually uses is likely to be a subset of the complete grammar.

Finally, working at the application layer offers the potential for developing an effector function to protect the server. I will discuss potential effector functions in Section 3.7.

I have not read any published statements about why (nearly) nobody has used the application layer for anomaly detection. In conversations, I have heard the claim that the application layer data is "too complex".

[12] A hypothesis is that web servers are more visible, and hence make good targets.

3.1.1 Preliminary results

In order to show that anomaly detection at the network application layer is likely to work, I have performed some preliminary work looking at the differences between normal web requests and attacks. If normal and abnormal are different, learning algorithms are likely to succeed.

The first step I took was looking at web requests. A normal web request looks like:

```
GET /gg-faq/ HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
Referer: http://www.grumman.net/
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows 98)
Host: www.i-pi.com
Connection: Keep-Alive
```

Attacks against web servers vary. One common attack is known as “Nimda” [52], and it looks like:

```
GET /scripts/..%%35c../winnt/system32/cmd.exe?/c+dir HTTP/1.0
Host: www
Connnection: close
```

At least 16 variations on this attack exist.

An off-by-one error in the Apache web server [36] made it vulnerable to requests such as:

```
POST /x.html HTTP/1.1
Host: 192.168.x.x
Transfer-Encoding: chunked

80000000
Rapid 7
0
```

The code which implements this part of the HTTP protocol is rarely used. It also illustrates why I predict that only a subset of the allowable protocol is actually used by any given web server.

The next step I took was to write a simple *n*-gram program for classifying web requests. For training, it counts the number of times each *n*-gram occurs in the data. The training data I used I collected on the i-pi.com and explorenm.com networks. To ensure that this data had no attacks, I

manually identified attacks and then wrote programs to remove them from the test data. The result was a training set consisting of 3719 requests.

For testing, I wrote a simple program which reads a request and calculates the ratio of novel n-grams to total n-grams[13]. However, the measure I used could distinguish between normal requests and Nimda attacks. One result of this test where $n = 17$ is presented and explained in Figure 1. The test data contained 344 web requests, a mix of attacks and valid requests that were not part of the training data. While it was not a part of the test data that went into Figure 1, the Apache chunk request that was presented above has no 17-grams in common with the requests in the training data. Even at $n = 5$, this attack has an abnormality value of 0.54. For a comparison, at $n = 5$ a Nimda attack has an abnormality value of 0.12.

My n-gram program is simple, yet it can easily detect some attacks. A smarter approach should be able to do better.

3.2 Justification for looking at web servers

I propose to use requests from web browsers to the web server (the HTTP protocol) for my testing[14]. The reason for this decision is that improving the security of web servers will have the greatest pay-off in terms of the cost of recent problems. However, my work will generalize to other network application-layer protocols. Specific examples include:

- The SSH protocol first negotiates cryptographic parameters of the communication and then performs key exchanges. The techniques I will develop in my research would be applicable to this portion of an SSH dialog.

- The control channel for FTP consists solely of highly-structured data, and like HTTP, a given server is likely to see only a subset of the full protocol defined in the standards, and therefore be good for a behavior-based IDS.

- The SMTP protocol first exchanges information about email before the email is transferred. This initial exchange was used by the Morris worm as one method of propagation [228, 230, 231].

After looking at HTTP communications, I will experiment with some of these other protocols.

[13] This measure could be improved—for example, a better measure might take into account the frequency of n-grams in the training data versus the frequency in the request.

[14] See Appendix A for additional details about how HTTP works

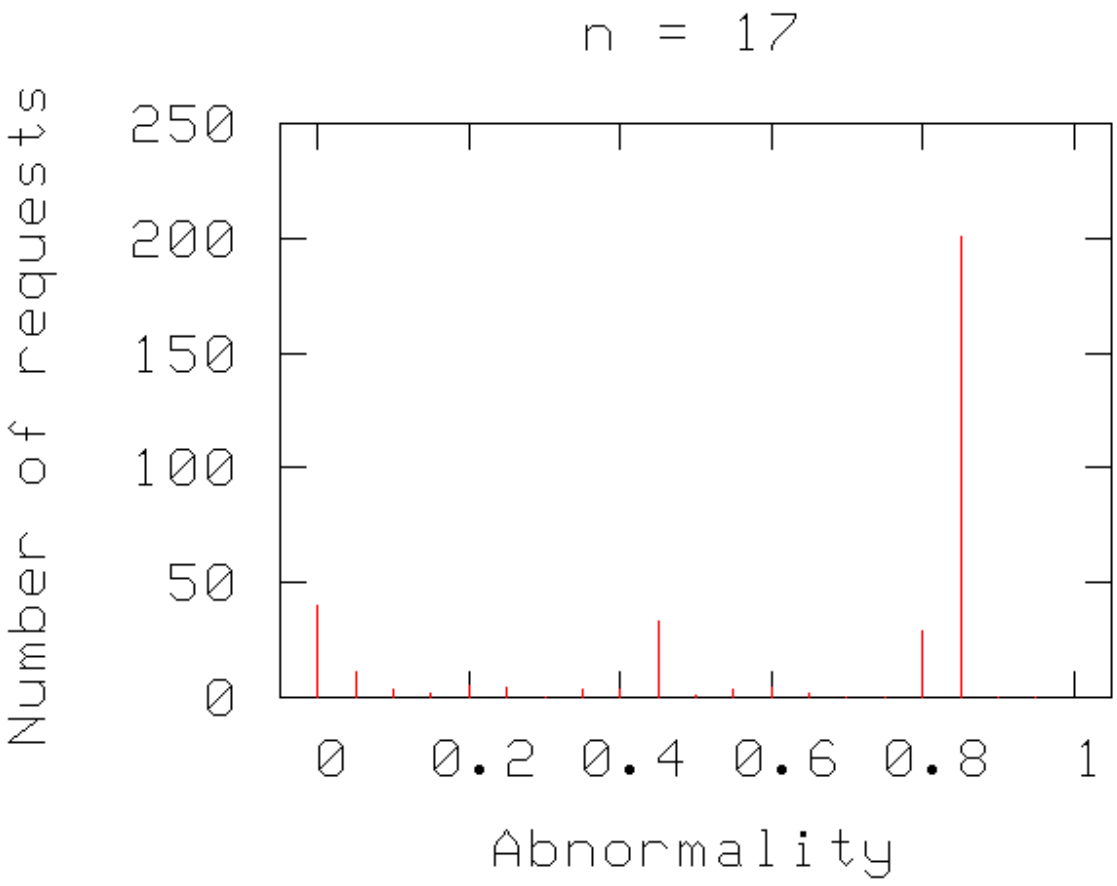

Figure 1: The results of characterizing 344 requests which were a mix of normal requests and attacks. The abnormality measure is 0 for a request which has no unusual *n*-grams in it, and 1 for a request which consists only of novel *n*-grams. For this plot, the abnormality values are binned with a bin width of 0.5. The *y* axis is the number of requests that received this abnormality value. The spikes around 0.4 and 0.8 represent Nimda attacks.

Not surprisingly, requests to a web server have a lot in common with each other. As a simple example, nearly every web request begins with either the string `GET` (the majority) or `POST`. A better measure of similarity can be shown with dot plots [35, 105, 106]. The dot plots I have produced (such as the one in Figure 2) show similarity in web requests.

While the HTTP requests are similar, they are not identical[15]. The following areas have high variability in the requests I have examined: the URL, the referring host, and the originating host. This variability still follows rules for how the parts are constructed (i.e., host names are proper DNS names as described in RFCs 1034 and 1035 [191, 192] or IP addresses from RFC 791 [121] and URLs are as described in RFCs 1630 [24] and 2616 [87]).

Some people assert that extensions to web servers such as PHP make HTTP requests too variable for a learning system. I believe that these assertions are false for the following reasons:

- The variability introduced is only in the URL, which is only one part of the overall HTTP request.
- Extensions written in PHP, Java, Perl, and similar languages still have a restricted form of URL that they recognize, even though it is more varied than simple requests for files.

To summarize, the initial results suggest that legitimate requests sent to web servers are similar to each other and different from attacks. Learning algorithms should be able to tell the difference. The same techniques that work for web servers should also work for other network application layer protocols, such as SSH and FTP. Therefore, my work can potentially be generalized to other types of servers.

3.3 Objective 1: identify and implement algorithms

A good method for detecting attacks at the network application level should include the following characteristics:

accurate Beyond the obvious need for accuracy, if the IDS has an effector function, false positives will cost an e-commerce site money.

fast A real-world solution must not seriously affect throughput or people will not use it.

convergence The system should not require days of CPU time or terabytes of training data in order to converge and be ready for use.

[15] Otherwise the problem would be trivial and not worthy of a dissertation.

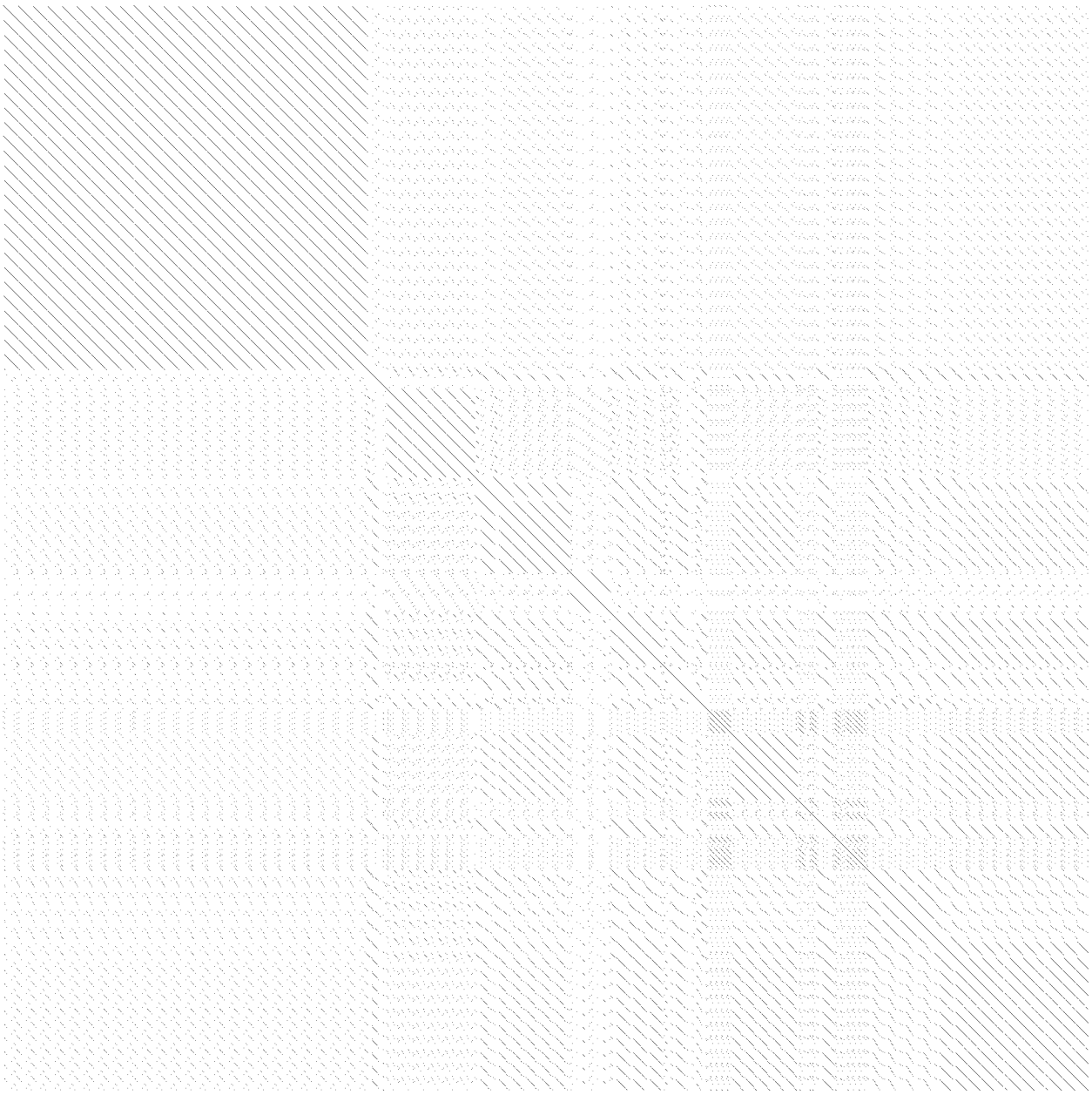

Figure 2: A dotplot showing the similarity of web requests. This plot was generated by breaking a request at white space and then placing the requests along the X and Y axes. A dot is wherever the two words are identical. The line at $x = y$ is the word matching itself. Other dots indicate the same word appearing in another request. Diagonal lines indicate strings of identical words.

dynamic determination of normal As researchers have noted (e.g., [114, 159]), "normal" is not a static state, but a moving target. Re-training as this drift occurs must not be onerous. Ideally, the IDS will drift as the normal behavior of the system drifts (although drifting with normal may also include learning subtle attacks as normal, a problem that needs to be addressed).

Using the network application layer will constrain the possible solutions; the IDS must sit on the web server or on a proxy in front of the web server. High-volume web servers already make use of a proxy for performance reasons (e.g., the Linux Virtual Server project [178]). The IDS could be added there. A firewall router that re-assembles the data stream could run the IDS without receiving the request, but it then loses the ability to do anything unless it can react before the last part of the attack is delivered to the server. In addition to these constraints, the IDS itself could be directly attacked if it is receiving the network connections. The simpler the solution, the less likely it is to contain bugs which can be exploited.

Based on the literature review in Section 2.1, I have identified several algorithms which I will implement at the network application layer.

Simple hash This algorithm works by storing all requests from the learning phase in a hash table. For the test phase, any request not in the hash table will be considered anomalous.

This algorithm is in the list to provide a comparison to the other, hopefully better, algorithms. I would expect it to be poor at handling perpetual novelty. However, due to the market penetration of Microsoft systems, many requests are identical. Therefore, this algorithm may do better than I expect. This algorithm is fast, training is simple, and tracking drifts in normal is as easy as adding new requests to the hash table.

***n*-grams and lookahead pairs** *n*-grams have the advantage that they are easy to compute and fast compared to most other algorithms (e.g., GAs as an extreme example). If the idea that normal session data at the network application layer are similar and the attacks are different is accurate, then this measure will be useful. My proof-of-concept experiment with *n*-grams showed they can distinguish between normal requests and some attacks.

Language learning The HTTP protocol is specified by a context-free grammar. While not all subsets of a language described by a CFG can be described by a CFG, subsets of HTTP used by a web server will be, because the subsets are created by deleting portions of the grammar. The goal of an IDS then would be to learn the subset that the server normally works with.

A serious obstacle to language identification is the result by Gold [96] that shows the generic task of learning a Context-free Grammar (CFG) cannot be done in the absence of negative evidence. However, the problem for my research is not that of learning an unknown language, but instead that of learning a subset of a known language. One simple learning algorithm suggested by Dennis Chao is to take the parser for the full protocol, profile it on normal requests, and then eliminate the code that was never used. Lee [160] surveys several CFG-learning algorithms which also show promise.

Two types of CFGs exist which I will investigate. The simplest is simply the subset of the full language, as described in the prior paragraph. The second, a stochastic CFG, also includes the probabilities of each production; these probabilities are calculated from the training data. A stochastic CFG provides a continuous value indicating the probability of the request occurring, which represents how anomalous the request is.

Immunologically inspired algorithms LISYS used three ideas (described in more detail on page 11) that I can see being possibly useful at the network application layer: negative detection, tolerization, and co-stimulation. LISYS used a human for co-stimulation; I will try co-stimulation by either another detector or another algorithm. For network traffic patterns, Hofmeyr [114] showed that LISYS had low overhead, was able to track normal, and was accurate. If these attributes transfer to use at the network application layer, it should do well.

Sensor fusion For any of the algorithms that perform well on at least part of the test data, I will look to see if combining them would provide a detector that performs better. Besides an improved detector, noting the strengths and weaknesses of various algorithms may provide insight into more theoretical aspects of anomaly detection.

One objection that has been raised is that data representation is a serious issue. In many adaptive computing examples, it is. However, HTTP has structure which I can take advantage of for the algorithms. The simple algorithms (the simple hash and imperfect matching) will work with a whole HTTP request as a unit. Some algorithms will work with simple strings (e.g., *n*-grams and the negative detection). The language learning algorithm will make use of the fact that HTTP is specified by a CFG, and as a result, the tokens are already well-defined.

To implement these algorithms, I will use *perl*. My reason for choosing this language is that I know it well and can produce running code faster in it than any other language. The primary disadvantage of *perl* would be that is slower than other languages, such as C++. However, the *perl* compiler may help with these efficiency problems.

3.4 Objective 2: identify data to use

Having a good source of data is critical for a good test. The data must be representative of real data, and it needs to be suitable for use in testing. Failure to properly ensure that the data is good has been a problem with several published papers. In this section, I will discuss what data I will use and why it will be relevant and suitable.

For attack data, the first question is, "What constitutes an attack?" Consider the following events which may be considered an attack:

1. Somebody maps the protected network, identifying what machines exist and what operating system and/or application software they are running.

2. Somebody probes a machine, looking for signs of weakness. The signs may be a banner identifying a software or operating system version, or it may be by the behavior of the operating system or application software. Consider two variants of this scenario:

   (a) the computer is not vulnerable to the probed weakness

   (b) the computer is vulnerable to the probed weakness.

3. Somebody successfully penetrates the system and is able to control what the computer does.

4. Somebody prevents the valid users from accessing the service(s) offered by the machine (a Denial-of-Service (DoS) attack).

Clearly, the latter two actions are attacks. Depending on the point of view of the organization running the computers, the first may also be considered an attack (for example, the 1999 DARPA MIT Lincoln Laboratories test considered IP sweeps to be an attack [172]). However, some mapping projects are more benign, being used to produce maps such as the Internet Mapping Project by Lumeta [33] or Netcraft's web server statistics [197]. Scenarios 1 and 2a are a normal state of affairs for machines with direct connections to the Internet. For the purposes of my research, I will consider scenarios 2–4 to represent attacks. The reason for this decision is all of these represent hazards to the machine, and either imminent or actual attacks.

One of the sources of data I will use will be data collected from live web servers:

- I run the domains **ExploreNM.com**, **i-pi.com**, and **translucentfirewalls.com**. The web server for these domains receives approximately 1500 hits per day. I have collected a week's worth of client requests to these web servers, and can easily collect more data. **ExploreNM.com** makes heavy use of PHP and *perl* scripts, so the client requests to it are more

varied than those to a web server offering nothing but files. Additionally, the web server also has Jakarta/Tomcat[16] installed and I will be increasing the use of this Apache module to provide a wider variety of requests heading to the web server.

- Mark Costlow is one of the owners of Southwest Cyberport (SWCP), an Albuquerque Internet Service Provider. Web servers at SWCP would be representative of the variety of servers that exist in the world, providing a wider collection of data with which to work. Costlow said "maybe" to my request for network data, depending on privacy concerns. When this proposal is accepted, I will further pursue addressing these concerns.

- Another potential source of data is the HTTP requests destined for the UNM CS departmental web server.

- I have a contact in Australia who may be able to supply web server data from his corporate server there.

I may generate some of the data by running web site walkers such as *checkbot* [57] or *MOMspider* [86]. These programs walk through a web site, requesting all of the web pages they can find a link to from some other page. Their purpose is to find broken links and other HTML errors. However, for my research, they will also be useful for potentially generating requests for objects that may not have been requested by people. This task is also assisted by the various web robots such as those run by search engines such as Google, but these robots take a long time[17] (potentially months) to scan an entire site whereas *checkbot* and *MOMspider* can accomplish the task in minutes or hours.

When collecting data, I need the data to be representative of normal. The first test I will do is to graph the number of unique requests versus total requests received. Ideally, this ratio will drop to a low value, indicating the novelty has fallen to a low value and that the data at this point is somewhat representative of normal. What would be missing at this point would be data to describe the perpetual novelty and the normal drift.

The actual amount of data I need is likely to depend on the algorithms I am using. From the collection of algorithms, I will determine the maximum amount needed and ensure that I have a sufficient amount.

To ensure that the data has not been modified by a successful attack, I will generate Tripwire[18]

[16] Jakarta/Tomcat adds a Java interpreter into the server, allowing more dynamic content to be requested (and generated).

[17] They work slowly to avoid using the server's bandwidth.

[18] Tripwire generates and stores cryptographic signatures and other file information about all of the important files on the system.

[137, 138, 139, 200] databases for the machines before they are connected to the Internet. As the machines run and when the data collection is finished, I will confirm that the machines are unlikely to be compromised by verifying them against their Tripwire database.

Besides the concern that the data must not be modified by an attack on the machine collecting the data, much of the data will be obtained from live networks. In order to collect data, I will need to ensure that some of the data is anonymized to protect the privacy of the web clients and web server owners, especially for the data collected from Southwest Cyberport. I need to further research anonymizing methods which will still allow valid testing of the data.

Another characterization of the data that I will do is to determine the actual distribution of attacks for a machine on the Internet. In order to calculate this, I will write a collection of programs:

1. verify that the request is valid HTTP.
2. the requests are for URLs that exist on the server.
3. For each request that is flagged as abnormal, I will personally examine it. Assuming it is an attack (as compared with a user typo in a URL), I will use a program to count and remove it from the rest of the data that needs to be manually examined.

Another measure for the real data will be the similarity of the requests, both from the same machine and between machines. Some of the ways I will calculate similarity include:

- ratio of attacks to valid requests
- ratio of HTTP GET to POST requests
- for HTTP GET requests, the number of unique pathnames requested
- graphing novel *n*-grams as new requests are added. I expect to see near-convergence after some period of time.
- graphing the distribution of request sizes.
- looking at the distributions of the request inter-arrival times (e.g., as Paxson and Floyd did for TELNET and FTP [206]).
- producing dotplots [35, 105, 106], using:
  - HTTP tokens,
  - entire HTTP requests, and

  - the higher-variability portions of the requests.

- looking at the results of the web server analysis tools *webalizer* [19] and *analog* [236], and WEBMINER [45] for more ideas.

These measures will help with ensuring any generated data matches real data, as well as providing me with a feel for the data.

The data that I have mentioned so far is primarily normal data. In order to have a thorough collection of attacks, I will make use of the Core Technologies Software that was recently purchased. I will also extract attacks from the real network data. These attacks provide real-world relevance to the tests I will do. However, many attacks exist that have not made it into script kiddie scripts. These attacks are described on full-disclosure mailing lists, such as BugTraq [219]. As of this writing, I have 23 such attacks. In addition to the real attacks, Fan et al. [85] discuss using artificial anomalies for testing anomaly detection systems. Using their methods, I will generate artificial anomalies for testing the systems. I will also see if the PROTOS test suite [201] would be useful. Finally, I will use the attack and vulnerability classifications by Bishop [28], Krsul et al. [147], and Landwehr et al. [153], to be sure that all classes of attack are covered. Using the tests described above, I will validate all generated data by comparing with real data to justify its actual usefulness. The final results of all this work will convince the reader of the adequacy of the test data and its suitability for the purpose to which I am putting it.

3.5 Objective 3: implement a suite of evaluations and test the algorithms

Because I will be testing many algorithms on the same data, the only reasonable way to run the tests will be off-line. All algorithms will be fed complete HTTP requests. Besides making the problem tractable, off-line testing eliminates potential load-based problems where an algorithm might miss part or all of a request. I will build a test framework which will allow me to plug each of the algorithms in, allowing identical test conditions for each one. This test framework will be open-source, covered by the GNU Public License.

I foresee two difficulties with the test framework. The first is how to choose values for tunable parameters. I will start with values (if any) listed in the paper(s) describing the algorithm. If values are missing, I will contact the authors to ask them what values they chose. In any case, I will contact the authors to ask why they chose the values they did, and if they have any advice for other values. In order to see how the algorithm fails, I will choose values at the extremes. The second difficulty will be dealing with continuous versus discrete data. Some algorithms return a probability that an attack is occurring. Others simply return a yes/no for whether something

represents an attack. Determining how to deal with this problem will require further research. However, McHugh [184] pointed out flaws in how the 1999 MIT Lincoln Labs test handled this problem, so the MIT solution is one I will not use.

Once I have a framework, I need metrics for comparing the algorithms. I intend to use the following:

Standard analysis of algorithms I will calculate the worst and average case time and space efficiencies. Additionally, I will add in real-world time and space needs to tie the theoretical to the practical.

Ease of implementation I will note the number of lines of code required for each algorithm, as well as if any part of the algorithm was difficult to implement. This information would be useful to anybody implementing these algorithms in a production system.

Unusual data I will look at how the algorithm performs on pathological uses of the HTTP protocol and bugs in the web server.

Learning curves For algorithms that learn, I will generate a learning curve to show that the system is converging. Such a curve is is important to show that the algorithm has learned the training data.

Ease of evading detection I will look for ways that an attacker who knows the algorithm(s) in use could craft an attack to evade detection.

True and false positives, true and false negatives Axelsson [12, 14] points out that if you assume that attacks are not the bulk of the data, an IDS must have a very low false positive rate in order to avoid swamping the system administrator with false positives. Knowing the actual rate of attacks, I can calculate the probability that the algorithm will detect an attack.

Independence Algorithms that fail independently can be combined to reduce the false positive rate, a possible solution to the base-rate fallacy problem that Axelsson noted.

Sensitivity Fan et al. [85] perturbed normal data to see how far it could be changed before it was classified as abnormal. Depending on how normal drifts, the sensitivity could measure how well an algorithm will be able to handle normal requests not in the training data, or it could show that the algorithm will miss nearly-normal attacks.

Other fields I will research other fields for other evaluation criteria that might be applicable. For example, in nuclear reactor operations, how do they evaluate their anomaly detection systems?

In order to determine which algorithms would work best in real-world situations, I will first rank the algorithms from best to worst using the criteria from Section 3.5; my goal would be to produce a *Consumer Reports*-style table where the algorithms are the vertical axis, the criteria the horizontal axis, and a symbol indicating how well the algorithm performed is at all of the intersections.

Besides the table, I will answer additional questions which are not encountered in Objective 3 in order to apply the results to real-world solutions:

- What is the latency provided by the algorithm (in case it was run on a proxy in front of the real web server(s))?
- What are the throughput limits (for either a proxy or running on a server)?
- Are the algorithms affected by system or network load?
- How would the alarm load affect a system operator (i.e., given a certain level of web requests, how often would the algorithm generate alarms)?
- How easily does the system adapt to a moving target as "normal" drifts? A system that has to be retrained regularly on normal data is not as useful as one that can retrain itself as it runs.

I foresee no major problems running the tests. I expect that they may take days to run, so I may run them on as many machines as I can get access to. Additionally, disk space may be a minor problem, but it is cheap—a slow 160GB disk can be obtained for under $300.00 (January 16, 2003; by the time I would need the space, it will be less expensive).

3.6 Objective 4: Generalize why the algorithms work or explain why they do not

If, as I suspect will be the case, one or more algorithms work well, I will look at them and explain formally why they work, with a goal of proving that these algorithms will also work well for other protocols, such as FTP and SSH. The way to do this formal explanation will be:

1. Note that all of these protocols use a CFG to describe the legal communications.
2. Normal communications are likely to be a subset of the legal communications, and the subset is generated by deleting parts of the grammar.
3. Show how the algorithm can learn this subset.
4. Show that most (all?) of the exploits at the application-layer are not a part of this subset.

If, however, none of the algorithms work well, I will then look at the application layer data and explain what about the data prevents algorithms such as those I tested from working. The result will be knowledge about what algorithms to try in the future, and which algorithms are not likely to work.

3.7 Objective 5: Develop effector functions

I see several potential effector functions for protecting a server:

Drop suspect non-URI portions of a request In GET requests, most browsers send additional information which may be useful for the server in determining what to send the client (e.g., which languages the browser's user is willing to accept). Most of these parts of a request are optional, and at worst, dropping them will cause the server to send the default version (e.g., use the default language) of a web page. For optional parts, the result would be a graceful degradation of the service offered.

However, since some of this additional information is not optional[19], either the effector function needs to understand HTTP, or the system administrator must accept that requests may be rendered invalid.

Verify the URI For requests where the URI is anomalous, several potential verification strategies exist:

- One verification method might be to make use of the `100 Continue` option. When this option is used, a client is in effect asking the server, "Can you handle this request?" Unfortunately, as currently specified in the standard, this option only is valid when the client has something to transmit in the request (e.g., uploading a file, or sending a response to a form). I will investigate the option of extending the protocol to allow this option for all request types.

[19] Such as `Content-Length` in some circumstances.

- A HEAD request is identical to a GET request, except that a body is not returned.
- The HTTP protocol standard specifies a request method of OPTIONS, which is to be used by a client to determine the resource or communication options associated with a URL without implying resource action or initiating a resource retrieval.
- Another option is modifying the server or writing a separate program which will check to see if a URL is valid. This alternate approach offers the benefit of a different code path, but may also offer a new place for bugs to be introduced, possibly allowing a different attack to succeed.

All of the URI verification options that communicate with the server run the risk of allowing an exploit to be delivered.

Block the request If a request is deemed anomalous, in some circumstances, sending an error response back to the user and dropping the request may be the best solution. This approach may be used in addition to other approaches. For example, if an anomaly sensor is continuous rather than binary, "very" anomalous requests might merit this response, while "slightly" anomalous requests might merit a different response.

Cooperate with other IDSs For a system running some form of host-based IDS (e.g., pH [225]), it may be possible to communicate with the IDS to indicate that an anomalous request is about to be delivered. This can cause the host-based system to be more vigilant, and possibly prevent an attack from succeeding. Of all of the options, this one is the best, but it presumes an adaptive host-based IDS with communication abilities.

Using an effector function means that an anomaly detection system must not only identify that something is anomalous, but it also may have to identify *what part* of the request is anomalous.

3.8 Summary

My research uses the requests sent to web servers to test several algorithms for their suitability for use at the network application layer. Besides determining which are likely to work, I will collect well-validated data and run tests on several algorithms in a thorough manner. With these quality data and testing methodologies, I will produce good algorithm comparisons. Additionally, my results will show which algorithms are likely to be useful in real-world, production systems. Additionally, I will investigate effector functions that an IDS might use to protect the web server.

Once I know which algorithms work well (if any), I will generalize why they work, with the goal of showing that they will also work well for other application-layer protocols. If they do not work, I will explain what about the application-layer data prevents them all from working.

4 Conclusion

Attacks against servers cost society billions of dollars, and the current solutions are not working against novel attacks. In order to improve this situation, I will:

1. Identify and implement algorithms that are promising at the network application layer.
2. Identify proper data to use in the testing.
3. Implement and run a suite of evaluations for testing the algorithms.
4. If one or more algorithms work well, generalize the algorithms that work to explain why they work well. If none of the algorithms work well, use this information to answer the question, "What is it about the application layer that prevents all of these anomaly-detection algorithms from working?"
5. Identify and implement potential effector functions for protecting the server.

The results of my work will be:

- network application-layer implementation of an IDS. This has never been done in the manner in which I propose.
- an excellent comparison of algorithms, something that is rarely done.
- theory explaining why the algorithms work (or do not work).
- one or more possible effector functions for protecting web servers.

My work is novel. Only one weak network application level anomaly detection system exists [79]. Comparisons in intrusion detection are rare; good comparisons are even rarer. Little has been published about the theory of intrusion detection. Therefore, my work will add to the corpus of knowledge.

My work has real-world relevance. Code Red cost society 2.6 billion dollars. The January 2003 Microsoft SQL Sapphire/Slammer worm cost between $950 million and $1.2 billion in the first

five days [167]. The 2002 Computer Security Institute/FBI survey [44] found that the forty-four percent of the respondents who were willing and/or able to quantify their financial losses reported $455,848,000.00 in financial losses. My work will reduce these losses and make the world a safer place to compute.

References

[1] ALESSANDRI, D. Using rule-based activity descriptions to evaluate intrusion-detection systems. In *Recent Advances in Intrusion Detection. Third International Workshop, RAID 2000, 2–4 Oct. 2000, Toulouse, France* (Berlin, Germany, 2000), H. Debar, L. Me, and S. Wu, Eds., Springer-Verlag, pp. 183–96.

[2] ALESSANDRI, D. Towards a taxonomy of intrusion detection systems and attacks. Tech. Rep. RZ 3366, IBM Research, Zurich Research Laboratory, Säumerstrasse 4, CH-8803 Rüschlikon, Switzerland, Sept. 2001.

[3] ALLEN, J., CHRISTIE, A., FITHEN, W., MCHUGH, J., PICKEL, J., AND STONER, E. State of the practice of intrusion detection technologies. Tech. Rep. CMU/SEI-99TR-028, Carnegie Mellon University, Software Engineering Institute, 2000.

[4] ALMGREN, M., DEBAR, H., AND DACIER, M. A lightweight tool for detecting web server attacks. In *Network and Distributed Systems Security (NDSS 2000) Symposium Proceedings* (2000), pp. 157–170.

[5] ALMGREN, M., AND LINDQVIST, U. Application-integrated data collection for security monitoring. In *Recent Advances in Intrusion Detection (RAID 2001)* (Davis, California, October 2001), LNCS, Springer, pp. 22–36. `http://www.sdl.sri.com/papers/raid2001/` Accessed 20 August 2002.

[6] ANDERSON, D., FRIVOLD, T., TAMARU, A., AND VALDES, A. Next-generation intrusion detection expert system (NIDES), software users manual, beta-update release. Tech. Rep. SRI–CSL–95–07, Computer Science Laboratory, SRI International, 333 Ravenswood Avenue, Menlo Park, CA 94025-3493, May 1994.

[7] ANDERSON, D., FRIVOLD, T., AND VALDES, A. Next-generation intrusion detection expert system (NIDES): A summary. Tech. Rep. SRI-CSL-95-07, SRI International, Computer Science Laboratory, May 1995.

[8] ANDERSON, D., LUNT, T. F., JAVITZ, H., TAMARU, A., AND VALDES, A. Detecting unusual program behavior using the statistical components of NIDES. Tech. Rep. SRI-CSL-95-06, SRI Computer Science Laboratory, May 1995. http://www.sdl.sri.com/papers/5sri/. Accessed 22 August 2002.

[9] ANDERSON, R., AND KHATTAK, A. The use of information retrieval techniques for intrusion detection. In *First International Workshop on Recent Advances in Intrusion Detection (RAID'98)* (1998).

[10] ASAKA, M., TAGUCHI, A., AND GOTO, S. The implementation of IDA: An intrusion detection agent system, 1999.

[11] AXELSSON, S. Research in intrusion-detection systems: A survey. Tech. Rep. 98-17, Department of Computer Engineering, Chalmers University of Technology, SE-412 96 Göteborg, Sweden, Dec. 1998.

[12] AXELSSON, S. The base-rate fallacy and its implications for the difficulty of intrusion detection. In *ACM Conference on Computer and Communications Security* (1999), pp. 1–7. http://www.ce.chalmers.se/staff/sax/difficulty.ps Accessed 19 August 2002.

[13] AXELSSON, S. On a difficulty of intrusion detection. In *Recent Advances in Intrusion Detection* (1999).

[14] AXELSSON, S. The base-rate fallacy and the difficulty of intrusion detection. *ACM Transactions on Information and Systems Security 3*, 3 (August 2000), 186–205.

[15] AXELSSON, S. Intrusion detection systems: A survey and taxonomy. Tech. Rep. 99-15, Dept. of Computer Engineering, Chalmers University of Technology, SE-412 96 Gteborg, Sweden, March 2000. http://www.ce.chalmers.se/staff/sax/taxonomy.ps Accessed 27 Feb 2002.

[16] BALASUBRAMANIYAN, J., GARCIA-FERNANDEZ, J., ISACOFF, D., SPAFFORD, E., AND ZAMBONI, D. An architecture for intrusion detection using autonomous agents. In *14th Annual Computer Security Applications Conference, 7–11 Dec. 1998, Phoenix, AZ, USA* (Los Alamitos, CA, USA, 1998), IEEE Computer Society, pp. 13–24.

[17] BALASUBRAMANIYAN, J. S., GARCIA-FERNANDEZ, J. O., ISACO, D., SPAFFORD, E., AND ZAMBONI, D. An architecture for intrusion detection using autonomous agents.

Tech. Rep. 98/05, Center for Education and Research in Information Assurance and Security (CERIAS), Purdue University, June 1998.

[18] BARBER, R. The evolution of intrusion detection systems—the next step. *Computers & Security 20*, 2 (2001), 132–45.

[19] BARRETT, B. L. Home of the webalizer. `http://www.mrunix.net/webalizer/` Accessed 8 January 2003.

[20] BASS, T. Multisensor data fusion for next generation distributed intrusion detection systems. In *Proceedings, 1999 IRIS National Symposium on Sensor and Data Fusion, May 1999.* (1999).

[21] BASS, T. Intrusion detection systems and multisensor data fusion. *Communications of the ACM 43*, 4 (April 2000), 99–105.

[22] BASS, T., AND GRUBER, D. A glimpse into the future of ID. *;login:* (September 1999). `http://www.usenix.org/publications/login/1999-9/features/future.html`. Accessed 29 July 2002.

[23] BERGHEL, H. The Code Red Worm. *Communications of the ACM 44*, 12 (December 2001), 15–19.

[24] BERNERS-LEE, T. Universal resource identifiers in www: A unifying syntax for the expression of names and addresses of objects on the network as used in the world-wide web, June 1994. RFC 1630. `ftp://ftp.isi.edu/in-notes/rfc1630.txt` Accessed March 25, 2002.

[25] BHARGAVAN, K., CHANDRA, S., MCCANN, P. J., AND GUNTER, C. A. What packets may come: automata for network monitoring. *ACM SIGPLAN Notices 36*, 3 (2001), 206–219.

[26] BIERMANN, E., CLOETE, E., AND VENTER, L. A comparison of intrusion detection systems. *Computers & Security 20*, 8 (2001), 676–83.

[27] BILAR, D., AND BURROUGHS, D. Introduction to state-of-the-art intrusion detection technologies. *Proceedings of the SPIE—The International Society for Optical Engineering 4232* (2001), 123–33.

[28] BISHOP, M. A taxonomy of Unix system and network vulnerabilities. Tech. Rep. CSE-9510, Department of Computer Science, University of California at Davis, May 1995.

[29] BRONSTEIN, A., DAS, J., DURO, M., FRIEDRICH, R., KLEYNER, G., MUELLER, M., SINGHAL, S., AND COHEN, I. Self-aware services: using Bayesian networks for detecting anomalies in internet-based services. In *2001 International Symposium on Integrated Network Management, 14–18 May 2001, Seattle, WA, USA* (Piscataway, NJ, USA, 2001), G. Pavlou, N. Anerousis, and A. Liotta, Eds., IEEE, pp. 623–38.

[30] BRONSTEIN, A., DAS, J., DURO, M., FRIEDRICH, R., KLEYNER, G., MUELLER, M., SINGHAL, S., AND COHEN, I. Self-aware services: Using Bayesian networks for detecting anomalies in internet-based services. Tech. Rep. HPL-2001-23, Internet Storage and Systems Laboratory, HP Laboratories, Palo Alto, CA, US, Oct. 2001.

[31] CANNADY, J. Artificial neural networks for misuse detection. In *Proceedings of the 1998 National Information Systems Security Conference (NISSC'98) October 5–8 1998. Arlington, VA.* (1998), pp. 443–456.

[32] CARVER, C., HILL, J., SURDU, J., AND POOCH, U. A methodology for using intelligent agents to provide automated intrusion response. In *Proceedings of the IEEE Systems, Man, and Cybernetics Information Assurance and Security Workshop, West Point, NY, June 6–7, 2000* (2000), pp. 110–116. `http://www.itoc.usma.edu/marin/Wshop/Papers2000/TP1_1.pdf` Accessed 30 December 2002.

[33] CHESWICK, B., AND BURCH, H. Internet mapping project. `http://research.lumeta.com/ches/map/` Accessed 7 January 2003.

[34] CHEUNG, S., CRAWFORD, R., DILGER, M., FRANK, J., HOAGLAND, J., LEVITT, K., ROWE, J., STANIFORD-CHEN, S., YIP, R., AND ZERKLE, D. The design of GrIDS: A graph-based intrusion detection system. Tech. Rep. CSE-99-2, U.C. Davis Computer Science Department, Jan. 1999. `http://seclab.cs.ucdavis.edu/arpa/grids/grids.ps` Accessed 6 June 2002.

[35] CHURCH, K. W., AND HELFMAN, J. I. Dotplot: a program for exploring self-similarity in millions of lines of text and code. *The Journal of Computational and Graphical Statistics 2*, 2 (1993), 153–174. Available at `http://www.research.att.com/~kwc/dotplot.ps`. Accessed 2 October 2002.

[36] COHEN, C. F. CERT advisory CA-2002-17 Apache web server chunk handling vulnerability, July 2002. http://www.cert.org/advisories/CA-2002-17.html. Accessed 24 July 2002.

[37] COHEN, C. F. CERT advisory CA-2002-18 OpenSSH vulnerabilities in challenge response handling, July 2002. http://www.cert.org/advisories/CA-2002-18.html. Accessed 24 July 2002.

[38] COHEN, W. W., AND SINGER, Y. Context-sensitive learning methods for text categorization. In *Proceedings of SIGIR-96, 19th ACM International Conference on Research and Development in Information Retrieval* (Zürich, CH, 1996), H.-P. Frei, D. Harman, P. Schäuble, and R. Wilkinson, Eds., ACM Press, New York, US, pp. 307–315.

[39] COMPUTER EMERGENCY RESPONSE TEAM (CERT). CERT advisory CA-1992-09 AIX anonymous FTP vulnerability, September 1997. http://www.cert.org/advisories/CA-1992-09.html Accessed 13 August 2002.

[40] COMPUTER EMERGENCY RESPONSE TEAM (CERT). CERT advisory CA-1999-03 FTP buffer overflows, July 1999. http://www.cert.org/advisories/CA-1999-03.html Accessed 13 August 2002.

[41] COMPUTER EMERGENCY RESPONSE TEAM (CERT). CERT advisory CA-1999-13 multiple vulnerabilities in WU-FTPD, November 1999. http://www.cert.org/advisories/CA-1999-13.html Accessed 13 August 2002.

[42] COMPUTER EMERGENCY RESPONSE TEAM (CERT). CERT advisory CA-1999-15 buffer overflows in SSH daemon and RSAREF2 library, Mar. 2000. http://www.cert.org/advisories/CA-1999-15.html. Accessed 24 July 2002.

[43] COMPUTER EMERGENCY RESPONSE TEAM (CERT). CERT advisory CA-2001-35 recent activity against secure shell daemons, Dec. 2001. http://www.cert.org/advisories/CA-2001-35.html. Accessed 24 July 2002.

[44] COMPUTER SECURITY INSTITUTE. Cyber crime bleeds U.S. corporations, survey shows, Apr. 2002. http://www.gocsi.com/press/20020407.html Accessed 16 January 2003.

[45] COOLEY, R., MOBASHER, B., AND SRIVASTAVA, J. Data preparation for mining world wide web browsing patterns. *Knowledge and Information Systems 1*, 1 (1999), 5–32.

[46] CROSBIE, M., DOLE, B., ELLIS, T., KRSUL, I., AND SPAFFORD, E. IDIOT—user guide. Tech. Rep. TR-96-050, Purdue University, West Lafayette, IN, US, Sept. 1996.

[47] CROSBIE, M., AND SPAFFORD, E. H. Applying genetic programming to intrusion detection. In *Working Notes for the AAAI Symposium on Genetic Programming* (MIT, Cambridge, MA, USA, 10–12 1995), E. V. Siegel and J. R. Koza, Eds., AAAI, pp. 1–8.

[48] CROSBIE, M., AND SPAFFORD, G. Defending a computer system using autonomous agents. In *8th National Information Systems Security Conference* (1995).

[49] CUPPENS, F. Managing alerts in a multi-intrusion detection environment. In *Seventeenth Annual Computer Security Applications Conference, 10-14 Dec. 2001, New Orleans, LA, USA* (Los Alamitos, CA, USA, 2001), IEEE Comput. Soc, pp. 22–31.

[50] CURRY, D., AND DEBAR, H. Intrusion detection message exchange format data model and extensible markup language (XML) document type definition, Dec. 2002. `http://www.ietf.org/internet-drafts/draft-ietf-idwg-idmef-xml-09.txt` Accessed 1 January 2003.

[51] DAMASHEK, M. Gauging similarity with n-grams: language-independent categorization of text. *Science 267*, 5199 (1995), 843–848.

[52] DANYLIW, R., DOUGHERTY, C., HOUSEHOLDER, A., AND RUEFLE, R. CERT advisory CA-2001-26 Nimda worm, September 2001. `http://www.cert.org/advisories/CA-2001-26.html`.

[53] DANYLIW, R., AND HOUSEHOLDER, A. CERT advisory CA-2001-19 “Code Red” worm exploiting buffer overflow in IIS indexing service DLL, August 2001. `http://www.cert.org/advisories/CA-2001-19.html` Accessed 13 August 2002.

[54] DARLING, T., AND SHAYMAN, M. A Markov decision model for intruder location in IP networks. In *39th IEEE Conference on Decision and Control, 12–15 Dec. 2000, Sydney, NSW, Australia* (Piscataway, NJ, USA, 2000), IEEE, pp. 1858–63 vol.2.

[55] DASGUPTA, D. Immunity-based intrusion detection system: a general framework. In *22nd National Information System Security Conference, 18–21 Oct. 1999, Arlington, VA, USA* (Gaithersburg, MD, USA, 1999), NIST, pp. 147–60 vol.1.

[56] DAVIS, R. Using an expert system to peel the computer virus onion. *EDPACS 20*, 2 (August 1992), 1–12.

[57] DE GRAAFF, H. Checkbot - home page, December 2001. `http://degraaff.org/checkbot/` Accessed 22 Feb 2002.

[58] DEBAR, H., BECKER, M., AND SIBONI, D. A neural network component for an intrusion detection system. In *1992 IEEE Computer Society Symposium on Research in Security and Privacy, 4-6 May 1992, Oakland, CA, USA* (1992), Los Alamitos, CA, USA : IEEE Computer Society Press, 1992, pp. 240–50.

[59] DEBAR, H., DACIER, M., AND WESPI, A. A revised taxonomy for intrusion-detection systems. Tech. Rep. RZ 3176 (# 93222), IBM Research, Zurich Research Laboratory, 8803 Rüschlikon, Switzerland, Oct. 1999.

[60] DEBAR, H., DACIER, M., AND WESPI, A. Towards a taxonomy of intrusion-detection systems. *Computer Networks 31*, 8 (Apr. 1999), 805–822.

[61] DEBAR, H., DACIER, M., AND WESPI, A. A revised taxonomy for intrusion-detection systems. *Annales des Telecommunications*, 7-8 (July-August 2000), 361–378.

[62] DEBAR, H., DACIER, M., WESPI, A., AND LAMPART, S. An experimentation workbench for intrusion detection systems. Tech. Rep. RZ 6519, IBM Research Division, Zurich Research Laboratory, 8803 Rüuschlikon, Switzerland, Sept. 1998.

[63] DENAULT, M., GRITZALIS, D., KARAGIANNIS, D., AND SPIRAKIS, P. Intrusion detection: approach and performance issues of the SECURENET system. *Computers & Security 13*, 6 (1994), 495–508.

[64] DENNING, D. An intrusion-detection model. In *1986 IEEE Symposium on Security and Privacy, 7-9 April 1986, Oakland, CA, USA* (1986), Washington, DC, USA : IEEE Comput. Soc. Press, 1986, pp. 118–31.

[65] DENNING, D. An intrusion-detection model. *IEEE Transactions on Software Engineering SE-13*, 2 (February 1987), 222–32.

[66] DERI, L. Ntop: a lightweight open-source network IDS. `http://jake.unipi.it/~deri/LightweightIDS.pdf` Accessed 30 July 2002., Sept. 2000.

[67] DERI, L., AND SUIN, S. Ntop: beyond ping and traceroute. In *Active Technologies for Network and Service Management. 10th IFIP/IEEE International Workshop on Distributed Systems. Operations and Management, DSOM'99. Proceedings, 11–13 Oct. 1999, Zurich, Switzerland* (Berlin, Germany, 1999), R. Stadler and B. Stiller, Eds., Springer-Verlag, pp. 271–83.

[68] DERI, L., AND SUIN, S. Improving network security using ntop. In *RAID 2000 Workshop on the Recent Advances in Intrusion Detection* (2000). http://jake.unipi.it/~deri/RAID.pdf Accessed 30 July 2002.

[69] DERI, L., AND SUIN, S. Practical network security: experiences with ntop. *Computer Networks 34*, 6 (December 2000), 873–80.

[70] DERI, L., SUIN, S., AND MASELLI, G. Design and implementation of an anomaly detection system: an empirical approach, Aug. 2001. http://luca.ntop.org/ADS.pdf Accessed 13 August 2002.

[71] D'HAESELEER, P. An immunological approach to change detection: theoretical results. In *9th IEEE Computer Security Foundations Workshop, 10-12 June 1996, Kenmare, Ireland* (Los Alamitos, CA, USA, 1996), IEEE Computer Society Press, pp. p.18–26.

[72] D'HAESELEER, P., FORREST, S., AND HELMAN, P. An immunological approach to change detection: Algorithms, analysis and implications. In *1996 IEEE Symposium on Security and Privacy, 6-8 May 1996, Oakland, CA, USA* (1996), Los Alamitos, CA, USA : IEEE Computer Society Press, 1996, pp. 110–119. http://cs.unm.edu/~forrest/publications/ieee-sp-96-neg-selec.pdf Accessed 6 June 2002.

[73] DOWELL, C., AND RAMSTEDT, P. The ComputerWatch data reduction tool. In *13th National Computer Security Conference. Proceedings. Information Systems Security. Standards - the Key to the Future, 1-4 Oct. 1990, Washington, DC, USA* (Gaithersburg, MD, USA, 1990), NIST, pp. 99–108 vol.1.

[74] DOYLE, J., SHROBE, H., AND SZOLOVITS, P. On widening the scope of attack recognition languages. http://medg.lcs.mit.edu/doyle/publications/ Accessed 29 July 2002, 2000.

[75] DRAELOS, T., COLLINS, M., DUGGAN, D., THOMAS, E., AND WUNSCH, D. Experiments on adaptive techniques for host-based intrusion detection. Tech. Rep. SAND2001-3065, Sandia National Laboratories, Albuquerque, NM, 2001. http://infoserve.sandia.gov/cgi-bin/techlib/access-control.pl/2001/013065.pdf. Accessed 12 December 2002.

[76] DUMOUCHEL, W. Computer intrusion detection based on Bayes factors for comparing command transition probabilities. Tech. Rep. TR91, National Institute of Statistical Sciences (NISS), 1999. http://www.niss.org/technicalreports/tr91.pdf. Accessed 29 July 2002.

[77] DUMOUCHEL, W., AND SCHONLAU, M. A comparison of test statistics for computer intrusion detection based on principal components regression of transition probabilities. In *Proceedings of the 30th Symposium on the Interface: Computing Science and Statistics* (1999), pp. 404–413.

[78] ECKMANN, S., VIGNA, G., AND KEMMERER, R. STATL: an attack language for state-based intrusion detection. *Journal of Computer Security 10*, 1–2 (2002), 71–103.

[79] EEYE DIGITAL SECURITY. SecureIIS^{TM}web server protection. At http://www.eeye.com/html/Products/SecureIIS/index.html. Accessed 27 December 2002.

[80] ENDLER, D. Intrusion detection. applying machine learning to Solaris audit data. In *14th Annual Computer Security Applications Conference, 7–11 Dec. 1998, Phoenix, AZ, USA* (Los Alamitos, CA, USA, 1998), IEEE Computer Society, pp. 268–79.

[81] ESKIN, E. Anomaly detection over noisy data using learned probability distributions. In *Proc. 17th International Conf. on Machine Learning* (2000), Morgan Kaufmann, San Francisco, CA, pp. 255–262.

[82] ESKIN, E., MILLER, M., ZHONG, Z., YI, G., LEE, W., AND STOLFO, S. Adaptive model generation for intrusion detection. In *Proceedings of the ACMCCS Workshop on Intrusion Detection and Prevention, Athens, Greece, 2000.* (2000).

[83] ESPONDA, F., FORREST, S., AND HELMAN, P. A formal framework for positive and negative detection schemes. *IEEE transactions on systems, man and cybernetics. Part B. Cybernetics.* in press.

[84] FALOUTSOS, C., AND OARD, D. W. A survey of information retrieval and filtering methods. Tech. Rep. CS-TR-3514, Electrical Engineering Department, University of Maryland, College Park, MD 20742, 1995.

[85] FAN, W., MILLER, M., STOLFO, S., LEE, W., AND CHAN, P. Using artificial anomalies to detect unknown and known network intrusions. In *2001 IEEE International Conference on Data Mining, 29 Nov.-2 Dec. 2001, San Jose, CA, USA* (Los Alamitos, CA, USA, 2001), IEEE Computer Society, pp. 123–30.

[86] FIELDING, R. Momspider: Multi-owner maintenance spider, May 1998. `http://www-old.ics.uci.edu/pub/websoft/MOMspider/` Accessed 22 Feb 2002.

[87] FIELDING, R., GETTYS, J., MOGUL, J., FRYSTYK, H., MASINTER, L., LEACH, P., AND BERNERS-LEE, T. Hypertext transfer protocol—HTTP/1.1, June 1999. RFC 2616. `ftp://ftp.isi.edu/in-notes/rfc2616.txt` Accessed 2002 Oct 2.

[88] FINLAY, I. A. CERT advisory CA-2002-21 vulnerability in PHP, July 2002. `http://www.cert.org/advisories/CA-2002-21.html` Accessed 14 August 2002.

[89] FLAJOLET, P., GUIVARC'H, Y., SZPANKOWSKI, W., AND VALLE, B. Hidden pattern statistics. In *Automata, Languages and Programming. 28th International Colloquium, ICALP 2001. Proceedings, 8–12 July 2001, Crete, Greece* (Berlin, Germany, 2001), Springer-Verlag, pp. 152–65.

[90] FORREST, S., HOFMEYR, S., SOMAYAJI, A., AND LONGSTAFF, T. A sense of self for unix processes. In *1996 IEEE Symposium on Security and Privacy, 6-8 May 1996, Oakland, CA, USA* (Los Alamitos, CA, USA, 1996), IEEE Computer Society Press, pp. 120–128.

[91] FORREST, S., PERELSON, A. S., ALLEN, L., AND CHERUKURI, R. Self-nonself discrimination in a computer. In *1994 IEEE Computer Society Symposium on Research in Security and Privacy, 16-18 May 1994, Oakland, CA, USA* (1994), Los Almitos, CA, USA : IEEE Computer Society Press, 1994, pp. 202–212.

[92] FYODOR, Y. 'SnortNet'—a distributed intrusion detection system, June 2000. `http://snortnet.scorpions.net/snortnet.ps` Accessed 5 August 2002.

[93] GHOSH, A., WANKEN, J., AND CHARRON, F. Detecting anomalous and unknown intrusions against programs. In *Proceedings of the 1998 Annual Computer Security*

Applications Conference (ACSAC'98), December 1998. (Los Alamitos, CA, USA, 1998), IEEE Computer Society, pp. 259–267.

[94] GHOSH, A. K., SCHWARTZBARD, A., AND SCHATZ, M. Learning program behavior profiles for intrusion detection. In *Proceedings 1st USENIX Workshop on Intrusion Detection and Network Monitoring* (Apr. 1999), pp. 51–62.

[95] GOAN, T. A cop on the beat: collecting and appraising intrusion. *Communications of the ACM 42*, 7 (July 1999), 46–52.

[96] GOLD, E. M. Language identification in the limit. *Information and Control 10* (1967), 447–474.

[97] GRAHAM, R. Nids—pattern search vs. protocol decode. *Computers & Security 20*, 1 (2001), 37–41.

[98] HAINES, J., ROSSEY, L., LIPPMANN, R., AND CUNNINGHAM, R. Extending the DARPA off-line intrusion detection evaluations. In *DARPA Information Survivability Conference and Exposition II. DISCEX'01, 12–14 June 2001, Anaheim, CA, USA* (Los Alamitos, CA, USA, 2001), IEEE Computer Society, pp. 35–45 vol.1.

[99] HAINES, J. W., LIPPMANN, R. P., FRIED, D. J., TRAN, E., BOSWELL, S., AND ZISSMAN, M. A. 1999 DARPA intrusion detection system evaluation: Design and procedures. Tech. Rep. TR-1062, Lincoln Laboratory, Massachusetts Institute of Technology, Lexington, MA, USA, Feb. 2001.

[100] HARMER, P., WILLIAMS, P., GUNSCH, G., AND LAMONT, G. An artificial immune system architecture for computer security applications. *IEEE Transactions on Evolutionary Computation 6*, 3 (June 2002), 252–80.

[101] HATCH, B., LEE, J., AND KURTZ, G. *Hacking Linux Exposed: Lunix Security Secrets & Solutions*. Osborne/McGraww-Hill, 2001.

[102] HAVRILLA, J. S. CERT advisory CA-2001-13 buffer overflow in IIS indexing service DLL, June 2001. `http://www.cert.org/advisories/CA-2001-13.html` Accessed 13 August 2002.

[103] HEBERLEIN, L. Network security monitor (NSM)—final report. Tech. rep., University of California at Davis Computer Security Lab, 1995. Lawrence Livermore National

Laboratory project deliverable.
http://seclab.cs.ucdavis.edu/papers/NSM-final.pdf.

[104] HEBERLEIN, L., DIAS, G., LEVITT, K., MUKHERJEE, B., WOOD, J., AND WOLBER, D. A network security monitor. In *1990 IEEE Computer Society Symposium on Research in Security and Privacy, 7–9 May 1990, Oakland, CA, USA* (Los Alamitos, CA, USA, 1990), IEEE Computer Society Press, pp. 296–304.

[105] HELFMAN, J. Similarity patterns in language. In *Visual Languages* (1994), pp. 173–175.

[106] HELFMAN, J. Dotplot patterns: a literal look at pattern languages. *TAPOS 2*, 1 (1995), 31–41. http://citeseer.nj.nec.com/cache/papers/cs/3880/http:zSzzSzwww.cs.unm.eduzSz~jonzSzdotplotzSztapos.pdf/helfman95dotplot.pdf. Accessed 2 October 2002.

[107] HELMAN, P., AND LIEPINS, G. Statistical foundations of audit trail analysis for the detection of computer misuse. *IEEE Transactions on Software Engineering 19*, 9 (September 1993), 886–901.

[108] HELMER, G., WONG, J., HONAVAR, V., AND MILLER, L. Intelligent agents for intrusion detection. In *1998 IEEE Information Technology Conference, Information Environment for the Future, 1-3 Sept. 1998, Syracuse, NY, USA* (New York, NY, USA, 1998), IEEE, pp. 121–4.

[109] HELMER, G., WONG, J., HONAVAR, V., AND MILLER, L. Automated discovery of concise predictive rules for intrusion detection. *Journal of Systems and Software 60*, 3 (February 2002), 165–75.

[110] HERNAN, S. V. CERT advisory CA-2001-07 file globbing vulnerabilities in various FTP servers, May 2001. http://www.cert.org/advisories/CA-2001-07.html Accessed 13 August 2002.

[111] HERNAN, S. V. CERT advisory CA-2002-05 multiple vulnerabilities in PHP fileupload, February 2002. http://www.cert.org/advisories/CA-2002-05.html Accessed 13 August 2002.

[112] HOCHBERG, J., JACKSON, K., STALLINGS, C., MCCLARY, J., DUBOIS, D., AND FORD, J. Nadir: an automated system for detecting network intrusion and misuse. *Computers & Security 12*, 3 (May 1993), 235–48.

[113] HOFMEYR, S., AND FORREST, S. Immunity by design: an artificial immune system. In *Proceedigns GECCO-99. Genetic and Evolutionary Computation Conference. Eighth International Conference on Genetic Algorithms (ICGA-99) and the Fourth Annual Gentic Programming Conference (GP-99), 13-17 July 1999, Orlando, FL, USA* (San Francisco, CA, USA, 1999), W. Banzhaf, J. Daida, A. Eiben, M. Garzon, V. Honavar, M. Jakiela, and R. Smith, Eds., Morgan Kaufmann Publishers, pp. 1289–96 vol.2.

[114] HOFMEYR, S. A. *An Immunological Model of Distributed Detection and Its Application to Computer Security*. PhD thesis, University of New Mexico, Computer Science Department, May 1999.

[115] HOFMEYR, S. A., AND FORREST., S. Immunizing computer networks: Getting all the machines in your network to fight the hacker disease. In *1999 IEEE Symposium on Security and Privacy* (1999), pp. 9–12.

[116] HOFMEYR, S. A., FORREST, S., AND SOMAYAJI, A. Intrusion detection using sequences of system calls. *Journal of Computer Security 6*, 3 (1998), 151–80. `http://cs.unm.edu/~forrest/publications/int_decssc.pdf` Accessed 13 March 2002.

[117] HUANG, M.-Y., JASPER, R., AND WICKS, T. A large scale distributed intrusion detection framework based on attack strategy analysis. *Computer Networks 31*, 23–24 (December 1999), 2465–75.

[118] ILGUN, K. USTAT: A real-time intrusion detection system for UNIX. Master's thesis, University of California Santa Barbara, Nov. 1992.

[119] ILGUN, K. USTAT: a real-time intrusion detection system for UNIX. In *IEEE Symposium on Research in Security and Privacy, 24-26 May 1993, Oakland, CA, USA* (1993), Los Alamitos, CA, USA : IEEE Comput. Soc. Press, 1993, pp. 16–28.

[120] ILGUN, K., KEMMERER, R., AND PORRAS, P. State transition analysis: a rule-based intrusion detection approach. *IEEE Transactions on Software Engineering 21*, 3 (March 1995), 181–99.

[121] INFORMATION SCIENCES INSTITUTE, UNIVERSITY OF SOUTHERN CALIFORNIA. Internet Protocol: DARPA internet program protocol specification, Sept. 1981. RFC 791. `ftp://ftp.isi.edu/in-notes/rfc791.txt` Accessed March 20, 2003.

[122] INGHAM, K., AND FORREST, S. A history and survey of network firewalls. Tech. Rep. 2002-37, University of New Mexico Computer Science Department, 2002. http://www.cs.unm.edu/colloq-bin/tech_reports.cgi?ID=TR-CS-2002-37. Accessed 29 December 2002. May be accepted for publication in the International Journal of Information Security.

[123] INSTITUTION OF ELECTRICAL ENGINEERS. IEE - INSPEC®- the quality database for physics, electronics and computing. At http://www.iee.org/Publish/INSPEC/. Accessed 27 December 2002. Available at UNM from the Library web page.

[124] ISO/TC97/SC16. Reference model of open systems interconnection. Tech. Rep. N. 227, International Organization for Standardization, June 1979.

[125] JACKSON, K. A. Intrusion detection system (IDS) product survey. Tech. Rep. LA-UR-99-3883, Distributed Knowledge Systems Team, Computer Research and Applications Group, Computing, Information, and Communications Division, Los Alamos National Laboratory, Los Alamos, New Mexico, USA, 1999.

[126] JACKSON, K. A., DUBOIS, D. H., AND STALLINGS, C. A. NADIR–A prototype network intrusion detection system. Tech. Rep. LA-UR–90-3726, Los Alamos National Laboratory, Los Alamos, NM USA 87545, 1990.

[127] JAVITZ, H., AND VALDES, A. The SRI IDES statistical anomaly detector. In *1991 IEEE Computer Society Symposium on Research in Security and Privacy, 20–22 May 1991, Oakland, CA, USA* (Los Alamitos, CA, USA, 1991), IEEE Computer Society Press, pp. 316–326.

[128] JHA, S., AND HASSAN, M. Building agents for rule-based intrusion detection system. *Computer Communications 25*, 15 (September 2002), 1366–73.

[129] JHA, S., TAN, K., AND MAXION, R. Markov chains classifiers and intrusion detection. In *14th IEEE Computer Security Foundations Workshop, 11–13 June 2001, Cape Breton, NS, Canada* (Los Alamitos, CA, USA, 2001), IEEE Computer Society, pp. 206–19.

[130] JONES, A., AND LI, S. Temporal signatures for intrusion detection. In *Seventeenth Annual Computer Security Applications Conference, 10–14 Dec. 2001, New Orleans, LA, USA* (Los Alamitos, CA, USA, 2001), IEEE Computer Society, pp. 252–61.

[131] JONES, A. K., AND LIN, Y. Application intrusion detection using langauge library calls. In *Seventeenth Annual Computer Security Applications Conference, 10-14 Dec. 2001, New Orleans, LA, USA* (Los Alamitos, CA, USA, 2001), IEEE Computer Society, pp. 22–31.

[132] JONES, A. K., AND SIELKEN, R. S. Computer system intrusion detection: A survey. Tech. rep., University of Virginia Computer Science Department, 1999. http://www.cs.virginia.edu/~jones/IDS-research/Documents/jones-sielken-survey-v11.pdf Accessed 6 June 2002.

[133] JULISCH, K. Mining alarm clusters to improve alarm handling efficiency. In *Seventeenth Annual Computer Security Applications Conference, 10-14 Dec. 2001, New Orleans, LA, USA* (Los Alamitos, CA, USA, 2001), IEEE Comput. Soc, pp. p.12–21.

[134] KAN KVARNSTRÖM, H. A survey of commercial tools for intrusion detection. Tech. Rep. 99-8, Department of Computer Engineering, Chalmers University of Technology, Götenborg, Sweden, Oct. 1999.

[135] KANSON, P. H. All public hospitals in Gothenburg Sweden crippled by Nimda. *Forum on Risks to the Public in Computers and Related Systems 21*, 67 (Oct. 2001). http://catless.ncl.ac.uk/Risks/21.67.html#subj13. Accessed 27 Dec 2002.

[136] KEMMERER, R., AND VIGNA, G. Intrusion detection: a brief history and overview. *Computer 35*, 4 (Apr. 2002), 27–30.

[137] KIM, G. H., AND SPAFFORD, E. H. Experiences with Tripwire: Using integrity checkers for intrusion detection. In *Proceedings of the Systems Administration, Networking and Security Conference III (SANS); Washington DC* (1993). http://coast.cs.purdue.edu/pub/papers/Tripwire/Tripwire-SANS.pdf Accessed 19 July 2002.

[138] KIM, G. H., AND SPAFFORD, E. H. The design and implementation of Tripwire: A file system integrity checker. In *2nd ACM Conference on Computer and Communications Security, 2–4 Nov. 1994, Fairfax, VA, USA* (New York, NY, USA, 1994), Association for Computing Machinery, pp. 18–29.

[139] KIM, G. H., AND SPAFFORD, E. H. Writing, supporting, and evaluating Tripwire: A publicly available security tool. In *1994 USENIX UNIX Applications Development*

Symposium, 25–28 April 1994, Toronto, Ontario, Canada (Berkeley, CA, USA, 1994), USENIX Association, pp. 89–107.

[140] KIM, J. An artificial immune system for network intrusion detection. In *Graduate Student Workshop, Genetic and Evolutionary Computation Conference, Orlando, Florida. July 13–17 (GECCO-99)* (July 1999), U.-M. O'Reilly, Ed., pp. 369–370.

[141] KIM, J., AND BENTLEY, P. The artificial immune model for network intrusion detection. In *7th European Conference on Intelligent Techniques and Soft Computing (EUFIT'99), Aachen, Germany* (1999).

[142] KIM, J., AND BENTLEY, P. The human immune system and network intrusion detection. In *7th European Congress on Intelligent Techniques and Soft Computing (EUFIT '99), Aachen, Germany, September 13–19* (1999).

[143] KIM, J., AND BENTLEY, P. Negative selection and niching by an artificial immune system for network intrusion detection. In *Late Breaking Papers at the 1999 Genetic and Evolutionary Computation Conference* (1999), pp. 149–158.

[144] KIM, J., AND BENTLEY, P. J. Towards an artificial immune system for network intrusion detection: An investigation of dynamic clonal selection. In *Proceedings of the Congress on Evolutionary Computation (CEC-2002), Honolulu, HI* (May 2002), pp. 1015–1020.

[145] KO, C., FINK, G., AND LEVITT, K. Automated detection of vulnerabilities in privileged programs by execution monitoring. In *Tenth Annual Computer Security Applications Conference, 5-9 Dec. 1994, Orlando, FL, USA* (Los Alamitos, CA, USA, 1994), IEEE Computer Society Press, pp. 134–144.

[146] KOHOUT, L., YASINSAC, A., AND MCDUFFIE, E. Activity profiles for intrusion detection. In *2002 Annual Meeting of the North American Fuzzy Information Processing Society Proceedings, 27-29 June 2002, New Orleans, LA, USA* (Piscataway, NJ, USA, 2002), J. Keller and O. Nasraoui, Eds., IEEE, pp. 463–8.

[147] KRSUL, I., SPAFFORD, E., AND TRIPUNITARA, M. Computer vulnerability analysis. Tech. rep., COAST Laboratory, Purdue University, West Lafayette, IN 47907-1398 USA, May 1998.

[148] KUMAR, S. *Classification and detection of computer intrusions*. PhD thesis, Purdue University, 1995.

[149] KUMAR, S., AND SPAFFORD, E. H. A Pattern Matching Model for Misuse Intrusion Detection. In *Proceedings of the 17th National Computer Security Conference* (1994), pp. 11–21.

[150] KUMAR, S., AND SPAFFORD, E. H. An Application of Pattern Matching in Intrusion Detection. Tech. Rep. 94–013, Department of Computer Sciences, Purdue University, 1994.

[151] KUMAR, S., AND SPAFFORD, E. H. A software architecture to support misuse intrusion detection. In *Proceedings of the 18th National Information Security Conference* (1995), pp. 194–204.

[152] KURI, J., AND NAVARRO, G. Fast multipattern search algorithms for intrusion detection. In *SPIRE'2000—String Processing and Information Retrieval, 27–29 Sept. 2000, A Curuna, Spain* (2000), pp. 169–180.

[153] LANDWEHR, C., BULL, A., MCDERMOTT, J., AND CHOI, W. A taxonomy of computer program security flaws. *ACM Computing Surveys 26*, 3 (September 1994), 211–54.

[154] LANE, T., AND BRODLEY, C. Approaches to online learning and concept drift for user identification in computer security. In *Fourth International Conference on Knowledge Discovery and Data Mining, 27–31 Aug. 1998, New York, NY, USA* (Menlo Park, CA, USA, 1998), P. Agrawal, R.; Stolorz, Ed., AAAI Press, pp. 259–63.

[155] LANE, T., AND BRODLEY, C. Temporal sequence learning and data reduction for anomaly detection. In *5th ACM Conference on Computer and Communications Security, 2–5 Nov. 1998, San Francisco, CA, USA* (New York, NY, USA, 1998), ACM, pp. 150–8.

[156] LANE, T., AND BRODLEY, C. E. Detecting the abnormal: Machine learning in computer security. Tech. Rep. ECE-97-1, Department of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907, Jan. 1997.

[157] LANE, T., AND BRODY, C. E. An application of machine learning to anomaly detection. In *20th Annual National Information Systems Security Conference* (1997), vol. 1, pp. 366–380.

[158] LANE, T., AND BRODY, C. E. Sequence matching and learning in anomaly detection for computer security. In *Proceedings of the AAAI-97 Workshop on AI Approaches to Fraud Detection and Risk Management* (1997), pp. 43–49.

[159] LANE, T. D. *Machine learning techniques for the domain of anomaly detection for computer security*. PhD thesis, Department of Electrical and Computer Engineering, Purdue University, July 1998.

[160] LEE, L. Learning of context-free languages: A survey of the literature. Tech. Rep. TR-12-96, Harvard University, 1996. ftp://deas-ftp.harvard.edu/techreports/tr-12-96.ps.gz.

[161] LEE, S., AND HEINBUCH, D. Training a neural-network based intrusion detector to recognize novel attacks. *IEEE Transactions on Systems, Man & Cybernetics, Part A (Systems & Humans) 31*, 4 (July 2001), 294–9.

[162] LEE, W. *A Data Mining Framework for Constructing Features and Models for Intrusion Detection Systems*. PhD thesis, Columbia University, 1999.

[163] LEE, W., AND STOLFO, S. A framework for constructing features and models for intrusion detection systems. *ACM Transactions on Information and Systems Security 3*, 4 (November 2000), 227–61.

[164] LEE, W., STOLFO, S., AND MOK, K. A data mining framework for building intrusion detection models. In *1999 IEEE Symposium on Security and Privacy, 9–12 May 1999, Oakland, CA, USA* (Los Alamitos, CA, USA, 1999), IEEE Computer Society, pp. 120–32.

[165] LEE, W., STOLFO, S. J., AND MOK, K. W. Mining in a data-flow environment: experience in network intrusion detection. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining* (1999), ACM Press, pp. 114–124.

[166] LEE, W., AND XIANG, D. Information-theoretic measures for anomaly detection. In *2001 IEEE Symposium on Security and Privacy. S&P 2001, 14–16 May 2001, Oakland, CA, USA* (2001), pp. 130–143.

[167] LEMOS, R. Counting the cost of slammer, Jan. 2003. http://news.com.com/2100-1001-982955.html?tag=mainstry Accessed 10 February 2003.

[168] LIEPINS, G. E., AND VACCARO, H. S. Intrusion detection: Its role and validation. *Computers & Security 11*, 4 (July 1992), 347–355.

[169] LINDQVIST, U., AND PORRAS, P. Detecting computer and network misuse through the production-based expert system toolset (P-BEST). In *1999 IEEE Symposium on Security and Privacy, 9–12 May 1999, Oakland, CA, USA* (Los Alamitos, CA, USA, 1999), IEEE Computer Society, pp. 146–161.

[170] LINDQVIST, U., AND PORRAS, P. A. eXpert-BSM: A host-based intrusion detection solution for Sun Solaris. In *Proceedings of the 17th Annual Computer Security Applications Conference (ACSAC 2001)* (New Orleans, Louisiana, Dec.10–14 2001), IEEE Computer Society, pp. 240–251.

[171] LIPPMANN, R., FRIED, D., GRAF, I., HAINES, J., KENDALL, K., MCCLUNG, D., WEBER, D., WEBSTER, S., WYSCHOGROD, D., CUNNINGHAM, R., AND ZISSMAN, M. Evaluating intrusion detection systems: the 1998 DARPA off-line intrusion detection evaluation. In *DARPA Information Survivability Conference and Exposition, 2000. DISCEX '00. Proceedings, Volume: 2* (1999), pp. 12–26.

[172] LIPPMANN, R., HAINES, J., FRIED, D., KORBA, J., AND DAS, K. The 1999 DARPA off-line intrusion detection evaluation. *Computer Networks 34*, 4 (October 2000), 579–95.

[173] LIPPMANN, R., HAINES, J., FRIED, D., KORBA, J., AND DAS, K. Analysis and results of the 1999 DARPA off-line intrusion detection evaluation. In *Recent Advances in Intrusion Detection. Third International Workshop, RAID 2000, 2–4 Oct. 2000, Toulouse, France* (Berlin, Germany, 2000), H. Debar, L. Me, and S. Wu, Eds., Springer-Verlag, pp. 162–82.

[174] LUNDIN, E., AND JONSSON, E. Some practical and fundamental problems with anomaly detection. In *Proceedings of the Fourth Nordic Workshop on Secure IT systems (NORDSEC'99)* (1999).

[175] LUNT, T. A survey of intrusion detection techniques. *Computers & Security 12*, 4 (June 1993), 405–18.

[176] LUNT, T. F. Automated audit trail analysis and intrusion detection: A survey. In *11th National Computer Security Conference* (Oct. 1988).

[177] LUNT, T. F. Detecting Intruders in Computer Systems. In *1993 Conference on Auditing and Computer Technology* (1993).

http://www.sdl.sri.com/papers/c/a/canada93/canada93.ps.gz Accessed 22 August 2002.

[178] MACK, J., MAROWSKY-BRE, L., ROBERTSON, A., AND ZHANG, W. The Linux virtual server project - Linux server cluster, January 2002. http://www.linuxvirtualserver.org/ Accessed 22 Feb 2002.

[179] MAHONEY, M. V., AND CHAN, P. K. Detecting novel attacks by identifying anomalous network packet headers. Tech. Rep. CS-2001-2, Florida Tech, 2001.

[180] MARCEAU, C. Characterizing the behavior of a program using multiple-length n-grams. In *New Security Paradigms Workshop 2000, 18–22 Sept. 2000, Ballycotton, Ireland* (New York, NY, USA, 2001), ACM, pp. 101–10. http://www.atc-nycorp.com/papers/Marceau_multiLengthStrings.pdf Accessed 13 August 2002.

[181] MAY, J., PETERSON, J., AND BAUMAN, J. Attack detection in large networks. In *DARPA Information Survivability Conference and Exposition II. DISCEX'01, 12–14 June 2001, Anaheim, CA, USA* (Los Alamitos, CA, USA, 2001), IEEE Computer Society, pp. 15–21 vol.1.

[182] MAYFIELD, J., MCNAMEE, P., AND PIATKO, C. The jhu/apl haircut system at trec-8. In *Information Technology: Eighth Text REtrieval Conference (TREC-8), 16-19 Nov. 1999, Gaithersburg, MD, USA* (2001), pp. 445–452.

[183] MCHUGH, J. The 1998 Lincoln Laboratory IDS evaluation—a critique. In *Recent Advances in Intrusion Detection. Third International Workshop, RAID 2000, 2–4 Oct. 2000, Toulouse, France* (Berlin, Germany, 2000), H. Debar, L. Me, and S. Wu, Eds., Springer-Verlag, pp. 145–61.

[184] MCHUGH, J. Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory. *ACM Transactions on Information and Systems Security 3*, 4 (November 2000), 262–94.

[185] MCHUGH, J. Intrusion and intrusion detection. *International Journal of Information Security 1*, 1 (August 2001), 14–35.

[186] MICHAEL, C., AND GHOSH, A. Using finite automata to mine execution data for intrusion detection: a preliminary report. In *Recent Advances in Intrusion Detection. Third*

International Workshop, RAID 2000, 2-4 Oct. 2000, Toulouse, France (Berlin, Germany, 2000), H. Debar, L. Me, and S. Wu, Eds., Springer-Verlag, pp. 66–79.

[187] MICHAEL, C., AND GHOSH, A. Simple state-based approaches to program-based anomaly detection. *ACM Transactions on Information and Systems Security 5*, 3 (August 2002), 203 – 37.

[188] MICHAEL, C. C., AND GHOSH, A. Two state-based approaches to program-based anomaly detection. In *Annual Computer Security Applications Conference: Practical Solutions to Real Security Problems December 11–15, 2000 New Orleans, LA, USA* (Dec. 2000), Applied Computer Security Associates, IEEE Computer Society, Los Alamitos, CA, USA, pp. 21–30. http://www.acsac.org/2000/papers/96.pdf Accessed 13 August 2002.

[189] MILLER, E. L., SHEN, D., LIU, J., AND NICHOLAS, C. Performance and scalability of a large-scale N-gram based information retrieval system. *Journal of Digital Information (online refereed journal)* (2000).

[190] MITUZAS, D. Apache worm in the wild, June 2002. Posted on the BUGTRAQ mailing list. http://online.securityfocus.com/archive/1/279529. Accessed 24 July 2002.

[191] MOCKAPETRIS, P. Domain names—concepts and facilities, Nov. 1987. RFC 1034. ftp://ftp.isi.edu/in-notes/rfc1034.txt Accessed March 20, 2003.

[192] MOCKAPETRIS, P. Domain names—implementation and specification, Nov. 1987. RFC 1035. ftp://ftp.isi.edu/in-notes/rfc1035.txt Accessed March 20, 2003.

[193] MUKHERJEE, B., HEBERLEIN, L., AND LEVITT, K. Network intrusion detection. *IEEE Network 8*, 3 (May 1994), 26–41.

[194] MUKKAMALA, S., JANOSKI, G., AND SUNG, A. Intrusion detection using neural networks and support vector machines. In *2002 International Joint Conference on Neural Networks (IJCNN), 12–17 May 2002, Honolulu, HI, USA* (Piscataway, NJ, USA, 2002), vol. 2, IEEE, pp. 1702–1707. http://www.cs.nmt.edu/~IT/papers/hawaii7.pdf Accessed 13 August 2002.

[195] NEC RESEARCH INSTITUTE. CiteSeer: The NEC research institute scientific literature digital library. At http://citeseer.nj.nec.com/cs. Accessed 27 Dec. 2002.

[196] NERI, F. Comparing local search with respect to genetic evolution to detect intrusions in computer networks. In *2000 Congress on Evolutionary Computation, 16–19 July 2000, La Jolla, CA, USA* (Piscataway, NJ, USA, 2000), IEEE, pp. 238–43 vol.1.

[197] NETCRAFT. Netcraft web server survey. http://www.netcraft.com/survey/ Accessed 7 January 2003.

[198] NEUMANN, P. G., AND PORRAS, P. A. Experience with EMERALD to date. In *First USENIX Workshop on Intrusion Detection and Network Monitoring (ID'99), 9–12 April 1999, Santa Clara, CA, USA* (apr 1999), USENIX Association, Berkeley, CA, USA, pp. 73–80. http://www.sdl.sri.com/papers/det99/ Accessed 20 August 2002.

[199] NEWMAN, D., GIORGIS, T., AND YAVARI-ISSALOU, F. Intrusion detection systems: suspicious finds. *Data Communications International 27*, 11 (August 1998), 72–8, 80, 82. Online at http://www.networkmagazine.com/article/printableArticle?doc_id=DCM20000510S0034 and http://www.data.com/article/DCM20000510S0034. Accessed 29 December 2002.

[200] OPEN SOURCE DEVELOPMENT NETWORK. Sourceforge.net: Project info - tripwire, 2003. http://sourceforge.net/projects/tripwire/ Accessed 25 March 2002.

[201] OUSPG: OULU UNIVERSITY SECURE PROGRAMMING GROUP. PROTOS test-suite: c05-http-reply, Sept. 2001. http://www.analog.cx/ Accessed 10 January 2003.

[202] PATTON, S., YURCIK, W., AND DOSS, D. An Achilles' heel in signature-based IDS: Squealing false positives in SNORT. In *RAID 2001 Fourth International Symposium on Recent Advances in Intrusion Detection October 10–12, 2001, Davis, CA, USA* (2001). Available online only. http://www.raid-symposium.org/raid2001/papers/patton_yurcik_doss_raid2001.pdf Accessed 3 January 2003.

[203] PAULA, F. S., REIS, M. A., FERNANDES, D. M., AND GEUS, P. L. ADenoIdS: A hybrid IDS based on the immune system. In *Proceedings of the ICONIP2002: 9th International Conference on Neural Information Processing, Special Session on Artificial Immune Systems and Their Applications* (Nov. 2002). http://www.dcc.unicamp.br/~ra000504/papers/AdenoidsMod_ICONIP2002.pdf Accessed 25 April 2003.

[204] PAXSON, V. Bro: a system for detecting network intruders in real-time. In *Seventh USENIX Security Symposium, 26–29 Jan. 1998, San Antonio, TX, USA* (Berkeley, CA, USA, 1998), USENIX Association, pp. 31–51.

[205] PAXSON, V. Bro: a system for detecting network intruders in real-time. *Computer Networks 31*, 23–24 (December 1999), 2435–63.

[206] PAXSON, V., AND FLOYD, S. Wide area traffic: the failure of Poisson modeling. *IEEE/ACM Transactions on Networking 3*, 3 (1995), 226–244.

[207] PORRAS, P. A., AND NEUMANN, P. G. EMERALD: conceptual overview statement. `http://www.sdl.sri.com/papers/emerald-position1/` Accessed 20 August 2002., dec 1996.

[208] PORRAS, P. A., AND VALDES, A. Live traffic analysis of TCP/IP gateways. In *Internet Society's Networks and Distributed Systems Security Symposium* (March 1998). `http://www.sdl.sri.com/papers/gateway98/` Accessed 20 August 2002.

[209] PUKETZA, N., CHUNG, M., OLSSON, R., AND MUKHERJEE, B. A software platform for testing intrusion detection systems. *IEEE Software 14*, 5 (September 1997), 43–51.

[210] PUKETZA, N. J., ZHANG, K., CHUNG, M., MUKHERJEE, B., AND OLSSON, R. A. A methodology for testing intrusion detection systems. *IEEE Transactions on Software Engineering 22*, 10 (1996), 719–729.

[211] RAE, D., AND LUDLOW, D. Halt! who goes there? [internet intrusion detection benchtest]. *Network News (UK Edition)* (February 2000), 31–7.

[212] RAGSDALE, D., CARVER, C.A., J., HUMPHRIES, J., AND POOCH, U. Adaptation techniques for intrusion detection and intrusion response systems. In *IEEE International Conference on Systems, Man, and Cybernetics, 8–11 Oct. 2000, Nashville, TN, USA* (Piscataway, NJ, USA, 2000), IEEE, pp. 2344–9 vol.4.

[213] REIS, M. A., PAULA, F. S., FERNANDES, D. M., AND GEUS, P. L. A hybrid ids architecture based on the immune system. In *Anais do Wseg2002: Workshop em Seguranc, a de Sistemas Computacionais, Búzios, RJ, Maio de 2002. Workshop realizado durante o SBRC2002: Simpósio Brasileiro de Redes de Computadores* (May 2002). `http://www.dcc.unicamp.br/~ra000504/papers/IDShyb_WSEG2002.pdf` Accessed 25 April 2003.

[214] RICHARDS, K. Network based intrusion detection: a review of technologies. *Computers & Security 18*, 8 (1999), 671–82.

[215] ROESCH, M. Snort—lightweight intrusion detection for networks. In *13th Systems Administration Conference—LISA '99* (1999), pp. 229–238. http://www.usenix.org/events/lisa99/roesch.html Accessed 30 June 2002.

[216] RYAN, J., LIN, M.-J., AND MIIKKULAINEN, R. Intrusion detection with neural networks. In *Advances in Neural Information Processing Systems* (1998), M. I. Jordan, M. J. Kearns, and S. A. Solla, Eds., vol. 10, The MIT Press.

[217] SCOTT, S. L. Detecting network intrusion using a markov modulated nonhomogeneous poisson process. *Journal of the American Statistical Association* (2002). Submitted, under revision. Available at http://www-rcf.usc.edu/~sls/mmnhpp.ps Accessed 31 March 2003.

[218] SCOTT, S. L., AND SMYTH, P. The markov modulated poisson process and markov poisson cascade with applications to web traffic modeling. *Bayesian Statistics 7* (2003).

[219] SECURITYFOCUS. What is bugtraq? http://online.securityfocus.com/popups/forums/bugtraq/intro.shtml Accessed 10 January 2003.

[220] SEKAR, R., GUANG, Y., VERMA, S., AND SHANBHAG, T. A high-performance network intrusion detection system. In *ACM Conference on Computer and Communications Security* (1999), pp. 8–17.

[221] SMAHA, S. Haystack: an intrusion detection system. In *Fourth Aerospace Computer Security Applications Conference, 12–16 Dec. 1988, Orlando, FL, USA* (Washington, DC, USA, 1988), IEEE Computer Society Press, pp. 37–44.

[222] SNAPP, S., SMAHA, S., TEAL, D., AND GRANCE, T. The DIDS (distributed intrusion detection system) prototype. In *USENIX Association. Proceedings of the Summer 1992 USENIX Conference, 8-12 June 1992, San Antonio, TX, USA* (1992), Berkeley, CA, USA : USENIX Association, 1992, pp. 227–33.

[223] SNAPP, S. R., BRENTANO, J., DIAS, G. V., GOAN, T. L., HEBERLEIN, L. T., LIN HO, C., LEVITT, K. N., MUKHERJEE, B., SMAHA, S. E., GRANCE, T., TEAL, D. M., AND

MANSUR, D. DIDS (Distributed Intrusion Detection System) – motivation, architecture, and an early prototype. In *Proceedings of the 14th National Computer Security Conference* (Washington, DC, 1991), pp. 167–176.

[224] SNYDER, D. On-line intrusion detection using sequences of system calls. Master's thesis, Department of Computer Science, Florida State University, 2001.

[225] SOMAYAJI, A. *Operating System Stability and Security through Process Homeostasis*. PhD thesis, University of New Mexico, 2002. `http://www.cs.unm.edu/~soma/pH/uss-2000.pdf` Accessed 31 May 2002.

[226] SOMAYAJI, A., AND FORREST, S. Automated response using system-call delays. In *Proceedings of the 9th USENIX Security Symposium* (Berkeley, CA, US, August 2000), USENIX Association, pp. 185–197. `http://www.cs.unm.edu/~soma/pH/uss-2000.pdf` Accessed 31 May 2002.

[227] SOMAYAJI, A., HOFMEYR, S., AND FORREST, S. Principles of a computer immune system. In *Meeting on New Security Paradigms, 23-26 Sept. 1997, Langdale, UK* (1997), New York, NY, USA : ACM, 1998, pp. 75–82.

[228] SPAFFORD, E. The internet worm incident. In *ESEC '89. 2nd European Software Engineering Conference Proceedings, 11-15 Sept. 1989, Coventry, UK* (Berlin, West Germany, West Germany, 1989), C. Ghezzi and J. McDermid, Eds., Springer-Verlag, pp. 446–68.

[229] SPAFFORD, E., AND ZAMBONI, D. Intrusion detection using autonomous agents. *Computer Networks 34*, 4 (October 2000), 547–70.

[230] SPAFFORD, E. H. The Internet worm program: An analysis. Tech. Rep. Purdue Technical Report CSD-TR-823, Department of Computer Science, Purdue University, West Lafayette, IN 47907-2004, 1988. `ftp://ftp.cs.purdue.edu/pub/reports/TR823.PS.Z` Accessed 2002 Feb 20.

[231] SPAFFORD, E. H. The Internet worm incident. Tech. Rep. Purdue Technical Report CSD-TR-933, Department of Computer Science, Purdue University, West Lafayette, IN 47907-2004, 1991.

[232] STANIFORD-CHEN, S., CHEUNG, S., CRAWFORD, R., DILGER, M., FRANK, J., HOAGLAND, J., LEVITT, K., WEE, C., YIP, R., AND ZERKLE, D. GrIDS–A graph-based

intrusion detection system for large networks. In *Proceedings of the 19th National Information Systems Security Conference* (1996).

[233] TÖLLE, J., AND NIGGERMANN, O. Supporting intrusion detection by graph clustering and graph drawing. *Lecture Notes in Computer Science 1907* (2000). in Recent Advances in Intrusion Detection (RAID) Workshop, 2000.

[234] TSUDIK, G., AND SUMMERS, R. Audes-an expert system for security auditing. *Computer Security Journal 6*, 1 (1990), 89–93.

[235] TURKIA, M. Intrusion detection systems. Available at `http://www.cs.helsinki.fi/u/asokan/distsec/documents/turkia.ps.gz` Accessed 30 July 2002., 2000.

[236] TURNER, S. Analog: WWW logfile analysis. `http://www.analog.cx/` Accessed 8 January 2003.

[237] VACCARO, H. S., AND LIEPINS, G. E. Detection of anomalous computer session activity. In *1989 IEEE Symposium on Security and Privacy, 1-3 May 1989, Oakland, CA, USA* (1989), Washington, DC, USA : IEEE Comput. Soc. Press, 1989, pp. 280–289.

[238] VALDES, A., AND SKINNER, K. Adaptive, model-based monitoring for cyber attack detection. In *Recent Advances in Intrusion Detection (RAID 2000)* (Toulouse, France, October 2000), H. Debar, L. Me, and F. Wu, Eds., no. 1907 in Lecture Notes in Computer Science, Springer-Verlag, pp. 80–92.

[239] VALDES, A., AND SKINNER, K. Probabilistic alert correlation. In *Recent Advances in Intrusion Detection (RAID 2001)* (2001), no. 2212 in Lecture Notes in Computer Science, Springer-Verlag. `http://www.sdl.sri.com/papers/r/a/raid2001-pac/prob_corr.pdf` Accessed 20 August 2002.

[240] VERWOERD, T., AND HUNT, R. Intrusion detection techniques and approaches. *Computer Communications 25*, 15 (September 2002), 1356–65.

[241] VIGNA, G., ECKMANN, S., AND KEMMERER, R. Attack languages. In *ISW 2000. 34th Information Survivability Workshop, 24–26 Oct. 2000, Cambridge, MA, USA* (Piscataway, NJ, USA, 2000), IEEE, pp. 163–6.

[242] VIGNA, G., AND KEMMERER, R. NetSTAT: a network-based intrusion detection approach. In *14th Annual Computer Security Applications Conference, 7–11 Dec. 1998, Phoenix, AZ, USA* (Los Alamitos, CA, USA, 1998), IEEE Computer Society, pp. 25–34.

[243] VIGNA, G., AND KEMMERER, R. A. NetSTAT: a network-based intrusion detection system. *Journal of Computer Security 7*, 1 (1999), 37–71.

[244] WAN, T., AND YANG, X. D. IntruDetector: a software platform for testing network intrusion detection algorithms. In *Seventeenth Annual Computer Security Applications Conference, 10–14 Dec. 2001, New Orleans, LA, USA* (Los Alamitos, CA, USA, 2001), IEEE Computer Society, pp. 3–11.

[245] WARRENDER, C., FORREST, S., AND PEARLMUTTER, B. A. Detecting intrusions using system calls: Alternative data models. In *IEEE Symposium on Security and Privacy* (1999), pp. 133–145.

[246] WILLIAMS, P., ANCHOR, K., BEBO, J., GUNSCH, G., AND LAMONT, G. CDIS: Towards a computer immune system for detecting network intrusions. In *Lecture Notes in Computer Science 2212* (2001), Springer-Verlag, pp. 117–133. Presented at the 4th International Symposium on Recent Advanced in Intrusion Detection (RAID 2001). `http://en.afit.edu/ISSA/publications/WilliamsRAID.pdf` Accessed 19 August 2002.

[247] WU, S., CHANG, H., JOU, F., WANG, F., GONG, F., SARGOR, C., QU, D., AND CLEAVELAND, R. Jinao: Design and implementation of a scalable intrusion detection system for the ospf routing protocol. To appear in Journal of Computer Networks and ISDN Systems. `http://projects.anr.mcnc.org/JiNao/JiNaoJournal.ps` Accessed 30 December 2002.

[248] YE, N. A Markov chain model of temporal behavior for anomaly detection. In *Proceedings of the 2000 IEEE Systems, Man, and Cybernetics Information Assurance and Security Workshop, 2000* (2000), pp. 171–174.

[249] YE, N., CHEN, Q., EMRAN, S. M., AND NOH, K. Chi-square statistical profiling for anomaly detection. In *IEEE Systems, Man, and Cybernetics Information Assurance and Security Workshop June 6–7, 2000 at West Point, New York* (June 2000), pp. 187–193.

[250] YE, N., EHIABOR, T., AND ZHANG, Y. First-order versus high-order stochastic models for computer intrusion detection. *Quality and Reliability Engineering International 18*, 3 (May 2002), 243–50.

[251] YE, N., EMRAN, S., CHEN, Q., AND VILBERT, S. Multivariate statistical analysis of audit trails for host-based intrusion detection. *IEEE Transactions on Computers 51*, 7 (July 2002), 810–20.

[252] YE, N., AND LI, X. Application of decision tree classifiers to computer intrusion detection. In *DATA MINING 2000 Data Mining Methods and Databases for Engineering, Finance and Other Fields, July 2000, Cambridge, UK* (Southampton, UK, 2000), N. Ebecken and C. Brebbia, Eds., WIT Press, pp. 381–90.

[253] YE, N., YU, M., AND EMRAN, S. M. Probabilistic networks with undirected links for anomaly detection. In *IEEE Systems, Man, and Cybernetics Information Assurance and Security Workshop June 6–7, 2000 at West Point, New York* (June 2000), pp. 175–179.

[254] YEUNG, D.-Y., AND CHOW, C. Parzen-window network intrusion detectors. In *16th International Conference on Pattern Recognition, 11–15 Aug. 2002, Quebec City, Que., Canada* (Los Alamitos, CA, USA, 2002), R. Kasturi, D. Laurendeau, and C. Suen, Eds., IEEE Computer Society, pp. 385–8 vol.4.

[255] ZAMBONI, D. Doing intrusion detection using embedded sensors—thesis proposal. Tech. Rep. 2000-21, CERIAS, Purdue University, 2000.

[256] ZHANG, Y., AND LEE, W. Intrusion detection in wireless ad-hoc networks. In *MobiCom 2000. Sixth Annual International Conference on Mobile Computing and Networking, 6–11 Aug. 2000, Boston, MA, USA* (New York, NY, USA, 2000), ACM, pp. 275–83.

A Web server traffic

In my proposed research, I will be working with web servers and the requests that they receive. The justification for using web server requests was presented in Section 3.2. This section provides a general introduction and overview of these requests.

Communication between web browsers and servers follows the HTTP protocol standard (version 1.1 is specified in RFC 2616 [87]). A client sends a request to a server, and this request always begins with either `GET` or `POST`. The server then replies with the requested object, or an error indicating that the object cannot be delivered.

More than one request can be sent over a TCP (or other transport-layer) channel. However, the grammar describes how to determine when the end of a request has been reached.

The grammar which describes the HTTP protocol is straightforward and easy to implement. The result is that a simple web request looks like:

```
GET /gg-faq/ HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
Referer: http://www.grumman.net/
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows 98)
Host: www.i-pi.com
Connection: Keep-Alive
```

Attacks against web servers vary. One common buffer overflow attack is known as “Code Red”. Here is a Code Red attack, where I have deleted most of the attack; the original was 7914 bytes long:

```
GET /default.ida?XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXX%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u685
090%u9090%u8190%u00c3%u0003%u8b00%u531b%u53ff%u0078%u0000%u00=a
HTTP/1.0
Content-type: text/xml
Content-length: 3379
```

```
\C8\C8\01\00‘\E8\03\00\00\00\CC\EB\FEdg\FF6\00\00dg\89&\00\00\E8
\01\00\00\8D\85\\FE\FF\FFP\FFU\9C\8D\85\\FE\FF\FFP\FFU\98\8B@\10
\FF\FF\FFU\E4=\04\04\00\00\0F\94\C1=\04\08\00\00\0F\94\C5\CD\0F\
FF\FF\8Bu\08\81˜0\9A\02\00\00\0F\84\C4\00\00\00\C7F0\9A\02\00\00
85l\FF\FF\FF\00\00\00\00\C7\85‘\FF\FF\FF\01\00\00\00\8BE\80\89\8
```

Other attacks are more subtle. An off-by-one error in the Apache web server [36] made it vulnerable to requests such as:

```
POST /x.html HTTP/1.1
Host: 192.168.x.x
Transfer-Encoding: chunked

80000000
Rapid 7
0
```