

University of Richmond

UR Scholarship Repository

Honors Theses

Student Research

4-22-2004

Discovering the potential for advancements in intrusion detection systems

Kenneth J. Buonforte
University of Richmond

Follow this and additional works at: <https://scholarship.richmond.edu/honors-theses>



Part of the [Computer Sciences Commons](#), and the [Mathematics Commons](#)

Recommended Citation

Buonforte, Kenneth J., "Discovering the potential for advancements in intrusion detection systems" (2004). *Honors Theses*. 432.

<https://scholarship.richmond.edu/honors-theses/432>

This Thesis is brought to you for free and open access by the Student Research at UR Scholarship Repository. It has been accepted for inclusion in Honors Theses by an authorized administrator of UR Scholarship Repository. For more information, please contact scholarshipprepository@richmond.edu.

Discovering the Potential for Advancements in Intrusion Detection Systems

Kenneth J. Buonforte

Honors Thesis*

Department of Mathematics & Computer Science
University of Richmond

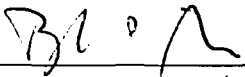
April 22, 2004

Abstract

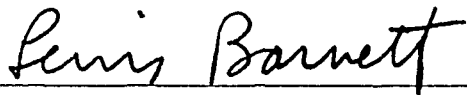
An intrusion detection system (IDS) is a collection of monitors strategically placed on a network or individual host in order to detect anomalous behavior. Since James Anderson introduced one of the first frameworks for an intrusion detection system in [1], researchers have extended the capabilities of these systems. IDSs take many forms, from systems as basic as command line audit logs to those that emulate the defense mechanisms of the human immune system. However, as intrusion detection has evolved, scientists and administrators alike are beginning to question the need for this technology due to its lack of quantifiable performance and in ability to handle increasingly fast networks. A report published by Richard Stiennon, Vice President of Gartner Inc., asserts that intrusion detection systems will be obsolete by the year 2005. The controversy surrounding Stiennon's statements has forced researchers to seriously consider the viability of these systems. Yet, despite valid complaints and concerns with the progress of intrusion detection systems, we feel these systems should continue to undergo research and development as there are still many unanswered questions in regards to its future use. Additionally, the failures that have spurred discontent with intrusion detection systems have often resulted from careless configuration and implementation rather than from their design. This paper demonstrates that there is still potential for futher exploration into the development of intrusion detection systems.

*Under the direction of Dr. Doug Szajda

This paper is part of the requirements for honors in computer science. The signatures below, by the advisor, a departmental reader, and a representative of the departmental honors committee, demonstrate that Kenneth J. Buonforte has met all the requirements needed to receive honors in computer science.



(advisor)



(reader)



(honors committee representative)

Contents

1	Introduction	4
2	Outline	5
3	History of Intrusion Detection Systems	5
3.1	Computer Security Threat Monitoring and Surveillance	6
3.2	Dorothy Denning and IDES	6
3.3	SHADOW	7
3.4	SNORT	7
4	Classifications of Intrusion Detection Systems	8
5	Insertion and Evasion: Staying Under the Radar	10
5.1	Insertion	11
5.2	Evasion	11
6	Denial of Service Attacks	12
7	The Snort IDS	14
8	Recent Advances in IDS Technology	16
8.1	Detecting Service Violations and DoS Attacks	17
8.2	A Virtual Machine Introspection Based Architecture for Intrusion Detection . .	18
8.3	Global Intrusion Detection in the DOMINO Overlay System	20
8.4	Building Attack Scenarios through Integration of Complementary Alert Correlation Methods	23
9	Conclusion	24

1 Introduction

Intrusion detection systems (IDS) have two major purposes: to efficiently detect malicious behavior on an individual host or network of hosts, and to initiate some kind of incident response mechanism to restore normalcy to a system. IDSs come in many forms ranging from basic command line audit loggers to full-scale emulations of the defense mechanisms found in the human immune system [14]. The framework mentioned in [1] deals with security audit logs, how they are monitored by company personnel, and what happens in the event of a suspected attack. IDS researchers consider this paper to be one of the first in intrusion detection. It illustrates the imminent need for a system of accurate aggregation and recognition of misuse patterns on an individual host.

As access control mechanisms are installed in the operating systems, the need for security audit trail data will be even greater; it will not only be able to record attempted unauthorized access, but will be virtually the only method by which user actions which are authorized but excessive can be detected.

Intrusion detection systems can be categorized by their methodology, or combination of methodologies, as well as by their topology in regards to placement of individual monitors (See **Section 3**). False positives, which are false alarms, and the usual complaint that they aren't fast enough to be real-time alarms are two major concerns with intrusion detection systems. But even more dangerous is that intrusion detection systems are also vulnerable to attack. The monitors of an IDS can be evaded, temporarily halted, tampered with, and attacked with enough brute force to eventually cause them to crash [8]. Thus, as stand-alone devices, intrusion detection systems cannot yet be trusted to provide a perfectly accurate view of what is actually occurring on a network.

As networks invariably become larger in scale and bandwidth, the challenge of developing a resilient, accurate, and efficient intrusion detection system intensifies. Security administrators are constantly challenged with new kinds of attacks in the forms of viruses, worms, denial of service, brute force, masquerading, social engineering, etc. Each of these attacks increase daily in their potential for inflicting significant financial and infrastructural damage. Recent statistics gathered by Carnegie Mellon's Computer Emergency Response Team (CERT) indicate that companies reported 137,529 incidences for the year of 2003, up 55,435 from 2002 and 84,871 from 2001. There is a very real pressure on those developing computer security devices to produce successful results.

The various weaknesses of current commercial intrusion detection systems continue to mar the overall reputation of the technology as solid and important security solutions. Thus, in an attempt to justify the necessity of continuing research in intrusion detection systems, this paper highlights a few of the recent advancements in the field that indicate promise for a more dependable solution.

2 Outline

Section 3: History of Intrusion Detection Systems In this section, we give an overview of the history of intrusion detection starting from James Anderson's first contributions to current commercial products including Snort.

Section 4: Classifications of Intrusion Detection Systems This section provides an introduction to the current classifications of intrusion detection namingly misuse detection and anomaly-based detection schemes in addition to two different physical frameworks known as host-based intrusion detection systems and network-based intrusion detection systems. Lastly, this section introduces some hybrid solutions.

Section 5: Staying Under the Radar via Insertion and Evasion Insertion and evasion attacks take advantage of a situation in which IDS monitors are not aligned with the hosts they protect and therefore can have their view of network/host activity skewed. This section covers examples of these kinds of attacks and potential solutions.

Section 6: Denial of Service In this section, we discuss the Denial of Service attack, another method of IDS interference. Attacks discussed in this section include the "Ping of Death", SYN Flooding, and Distributed Denial of Service (DDoS).

Section 7: Snort and the Open Source Community One of the most popular IDSs today is Snort, the Open Source Initiative's response to over-priced commercial IDSs. This section covers the advantages and shortcomings of using Snort in addition to some examples of its use.

Section 8: Recent Advances in IDS Technology Despite the challenges of designing a practical, resilient, and real-time IDS, recent advancements in intrusion detection offer some optimistic blue-prints for future solutions. This section will address new approaches to distributed intrusion detection, building an IDS with virtual machines, attack correlation, and improving the detection of denial of service attacks.

Section 9: Conclusion In the conclusion, we offer three reasons as to why we feel intrusion detection systems are important security devices. Additionally, we summarize the topics covered in this paper.

3 History of Intrusion Detection Systems

An intrusion detection systems is a collection of nodes responsible for detecting malicious activity on a host or network of hosts. Early forms of these models include audit logs that had two purposes: to log the actions of a user on a specific host, and to log the behavior/throughput on a network.

3.1 Computer Security Threat Monitoring and Surveillance

In 1980, James Anderson introduced a method for leveraging information gleaned by audit logs in order to come to certain conclusions regarding the behavior of a system, providing a framework for an intrusion detection system. It is considered by many researchers as the first academic acknowledgement of intrusion detection. Anderson discusses the importance of maintaining accurate audit logs and goes into some depth about the practices and steps taken to derive valuable behavioral characteristics about the integrity of a system. An important contribution that he makes in this paper is his observation of how easy it is to compromise system logs or avoid them altogether.

It is evident that such audit trails are not complete. Users (particularly ODP personnel with direct programming access to datasets) may operate at a level of control that bypasses the application level auditing and access controls. [...] Programmers with the ability to use access method primitives can frequently access database files directly without leaving any trace in the application access control and audit logs.

This paper concludes that one of the easiest ways for an attacker to avoid being caught is to alter the audit logs in order to erase the trail of intrusion.

3.2 Dorothy Denning and IDES

Dorothy Denning continued research in intrusion detection throughout the 1980's, making her most well known contribution in the form of the detailed model she describes in [13].

An important objective of our current research is to determine what activities and statistical measures provide the best discriminating power; that is, have a high rate of detection and a low rate of false alarms.

Denning claimed that her model, otherwise known as Intrusion Detection Expert System (IDES), was real-time and could run independently with other system applications. Despite the fact that her predictions about the real-time capabilities of these systems have not yet come true, her work provided a foundation for future models of research.

Another one of Denning's important contributions is the idea that the behavior of an anomalous presence in a system, either in the form of a virus or human attacker, has recognizable patterns different from regular activity. This idea of identifying abnormal behavior through offline and real-time statistical analysis is an intrusion detection scheme that continues to challenge researchers.

Finally, Denning considers the paradigm of network intrusion detection in addressing virus and worm patterns in her paper. This an evolution from Anderson's approach in that instead of intrusion detection being applicable strictly to an individual host, in her proposal intrusion detection should involve monitoring an entire network of hosts.

3.3 SHADOW

The U.S. military has been a long time proponent, through allocation of funds and resources, of developing intrusion detection. One of the most prominent voices in intrusion detection research is Stephen Northcutt. At the Naval Surface Warfare Center Dahlgren Division (NSWCDD), Northcutt developed a intrusion detection system dubbed Secondary Heuristic Analysis for Defensive Online Warfare (SHADOW). Built in 1994, this IDS is still available today although the latest release is a year old. According to the tool's documentation [15], SHADOW consists of a sensor located at some point near a system's firewall and another sensor placed elsewhere within a network. Through utilization of the `tcpdump` and `libpcap` libraries for network packet capture, SHADOW identifies events that match a network wide definition of interesting behavior where interesting means potentially malicious.

The key to effective use of SHADOW is intelligent definition of the `tcpdump` filters based on the network environment and educated recognition of known and potential exploits from traffic patterns.

Additionally, since this product is open source, it is freely distributed, which also means that the administrators of the tool are responsible for its successful performance, a recurrent theme found in the shortcomings of most security devices. Therefore, the developers of SHADOW emphasize in [15] the importance of having an intimate knowledge of the user's Linux installation and the interworking of his network.

Stephen Northcutt's contribution to IDS research also includes several books, among them [3] and [16].

3.4 SNORT

Similar to the Department of Defense's open source solution SHADOW is Snort, an IDS developed by Martin Roesch in 1998. Snort is one of the most widely used intrusion detection systems currently available. Snort's impressive collection of tools related to intrusion detection combined with the support of the open source community and that fact that it is freely distributable make it a highly marketable solution. According to [11], there are an estimated 200,000 installations worldwide and it is said to be as widely distributed as its closed market competitors. Snort is a signature-based tool that relies on a strong database of attack signatures for identifying potentially malicious network traffic. A core set of currently updated and widely applicable rules are included in a Snort download. Another powerful feature is the ability to write custom rules that are more specific to a user's network. Since Snort's syntax for rule creation is easy to learn, security administrators are better able to configure their IDS solution. The Statistical Packet Anomaly Detection Engine (SPADE), a heuristic method for detecting intrusion, is another feature available to Snort users. **Section 7** goes into more depth about some of these features as well as the advantages and disadvantages of Snort.

Other commercial IDSs include Cisco's NetRanger, Internet Security Systems's RealSecure, Symantec's Intruder Alert and NetProwler, and Enterasys's Dragon network intrusion detection solution and Squire host-based intrusion detection system. Many of these systems offer a hybrid approach to intrusion detection combining the advantages of predominant intrusion detection architectures.

4 Classifications of Intrusion Detection Systems

Understanding the tradeoffs and compromises associated with each categorization of an intrusion detection system is important in making optimal design decisions for a given organization. Therefore, it is necessary to recognize the different categorizations of the methods for intrusion detection and their physical implementation.

For any security administrator, the initial decisions about the IDS he wishes to implement must be based his organizations's mission-critical resources. For example, an IDS for a Department of Defense computer network should be configured to devote most of its attention toward hosts containing classified information. In a setting where the prevention of information leaks is of utmost importance, IDS sensors should be finely tuned for monitoring the traffic coming in and out of a network and also for watching individual hosts to make sure employees aren't distributing classified information to unauthorized persons. Since the main objective in this environment is to ensure the integrity of its mission-critical assets, larger than usual overhead costs are tolerable shortcomings if it means better protection of said assets. A corporation in the private sector is also concerned about corporate espionage and propriety information leaks. However, a corporation is less likely to enforce a host by host analysis of its employees' computers because it either wishes to establish an environment of trust amongst its personnel or it decides that the money spent on maintaining a complex IDS is simply not worth it. In fact, as described by [3], it can be relatively difficult to persuade the upper-management of a company that their organization needs a security solution as involved as an IDS. On the other hand, since so many organizations have reported losing billions of dollars due to network breaches and fraud, lately it appears that companies are more willing to invest in intrusion detection, which means that furthering the development of successful systems will become more important in the commercial setting.

There are two essential characterizations of the methods for intrusion detection: misuse detection and anomaly-based detection. Both methods define what is considered normal, acceptable behavior for the use of individual hosts on a network and the stream of traffic between hosts and from external networks.

Anomaly-based detection involves attempting to identify anomalous behavior through aggregation of various system or user statistics. For example, there are current efforts on the development of identification algorithms for profiling specific user based information including the commands he enters at a shell prompt and his unique keyboard entry patterns. The latter technique is often referred as keyboard dynamics. Currently inaccurate algorithms for determining these dynamics prevent this approach from acting as a stand-alone authentication scheme, but

it is an additional step to an more in-depth solution. In a sense, the IDS monitors become more autonomous in anomaly-based detection because they are the trained monitors that make deductions as to whether or not the current behavior of the system they're protecting is malicious. Unlike misuse detection, a well implemented anomaly-based detection scheme may be able to detect unknown attacks which may have very different attack signatures from other previously known attacks. While it is possible to successfully detect new attacks within the network paradigm, defining anomalous behavior for a host is often a more effective because the IDS monitors have access to more specific behavioral information with which to make IDS decisions.

A second method for intrusion detection is misuse (signature-based) detection where a monitor compares system behavior with a set of rules, also known as attack signatures, in order to recognize previously executed attacks. Misuse detection accomplishes this by parsing through either network data streams or user commands to discover characteristics of attack signatures indicative of a potential attack scenario. For example, Snort, is powerful because members of the Snort community are encouraged to report different attacks and develop new rules to protect against other similar attacks. This allows the IDS to have a thorough collection of attack signatures that, if updated appropriately, should prevent attacks from reoccurring. Another nice feature about this type of detection is that since the identifiable attacks have already occurred, it is a given that some system administrator will have knowledge as to how to deal with the attack and initiate effective incident response. Misuse detection is only as good as the community that develops the attack signatures and appropriate incident responses.

Placement of IDS monitors can also be divided into two dominant strategies. A network-based intrusion detection system (NIDS) attempts to identify malicious/abnormal behavior by focusing its analysis on incoming network traffic, usually at the packet level. Because network packets can have large payloads, analyzing a packet in its entirety is not ideal as it places execution strains on the IDS and possibly the network. Instead, most NIDSs do their primary analysis in the header of incoming packets where they can monitor protocol types, destination internet protocol (IP) addresses and ports, source IP addresses and ports, and other vital network activity information. Drawbacks to most NIDSs include slow execution, especially in large gigabit networks, and complaints that they don't have a specific enough view of what is actually happening at the host level. In other words, these IDSs have poor visibility.

Host-based intrusion detection systems (HIDS) remedy the problem of host level ignorance by monitoring specific user actions on an individual host. Keyboard and command loggers, CPU monitors, and other user-level modus operandi for collecting information are all considered frameworks for HIDSs. In earlier versions of this framework, HIDSs were blind to activity on other hosts as they were configured to focus on monitoring only the behavior of the individual user on one host. In a large network setting, HIDSs are still effective individually as their only responsibilities are still in monitoring individual hosts, however they now have the option of communicating with each other for better propagation of information essential for accurate intrusion detection. Additionally, in this scenario it doesn't matter how many new hosts are added to a network as long as each host is given its own monitor. In fact, recent advances in intrusion detection frequently include a protocol for host-to-host communication in combining

host-level monitors within an NIDS to build a more powerful hybrid solution. Similarly, misuse detection and anomaly-based detection can be combined to develop a stronger attack detection scheme which can detect old and new attacks. However, careful thought is required in developing a hybrid solution as the potential for an IDS that is cumbersome, intrusive, and vulnerable to attack increases if the paradigms are not combined in a careful manner. For example, if an IDS sensor is responsible for matching attack signatures with incoming packets and computing information regarding the stream of packets to hopefully discover anomalous behavior, the sensor will almost certainly become a network bottleneck. That is why there are research efforts still in progress to effectively examine the advantages and disadvantages of NIDS, HIDS, misuse detection, and anomaly detection in order to derive the most potent solution.

An example of a hybrid solution to intrusion detection is a distributed intrusion detection system (DIDS). Perhaps one of the most interesting attempts at a DIDS can be found in [14]. Authors Stephanie Forrest and Steven A. Hofmeyr model their intrusion detection system after the human immune system. They even give intrusion detection nodes proper cell names (i.e. t-lymphocyte) to differentiate between the purposes of each nodes in their DIDS. [18] is another form of DIDS that is built on an overlay network and is highly scalable and heterogeneous. The hierarchical design of DIDSs increases the redundancy of the system and the aggregation of data regarding system activity. Additional work in this area is found in [4,5,19,20].

Another interesting approach toward developing a hybrid solution is in using an artificial neural network as a framework for an IDS. These systems leverage the ability of neural networks that have been trained to understand normal system behavior to teach the IDS how to autonomously detect malicious behavior, thereby alleviating the reliance on human intervention for incident response. This has been a promising realm of research and development, primarily in academia [22,23,24,25].

5 Insertion and Evasion: Staying Under the Radar

Despite the fact that insertion and evasion have opposite definitions, they both can occur when there are inconsistencies between the NIDS monitors and the hosts a network. [9] graphically depicts this situation. In this figure, the NIDS resides at the same hierarchical level as the hosts it defends, unobtrusively reading packets off the line in a promiscuous mode setting, an ideal setting for an attacker to execute an evasion or insertion attack. Another property of NIDSs that make these attacks plausible is that the NIDS only checks IP packet headers rather than the entire packet. As previously mentioned, a good way to speed up a potentially slow NIDS is to focus only on the headers of incoming packets as payload sizes can be varied and unpredictable and header information usually provides enough detail about the entire packet for the purpose of intrusion detection. If a cracker successfully executes either method of IDS subversion correctly, he can infiltrate a network with a reasonable assurance that he will not be caught, at least until a security administrator recognizes that something is amiss with a host on the network and manually sounds the alarm. Finally, if the IDS resides on a network segment other than that of the hosts it defends, insertion attacks can still defeat the IDS through clever

manipulation of the packets with which the attacker executes his attack.

5.1 Insertion

Insertion occurs in a system where the hosts abide by more restricting rules regarding what packets to accept and reject than the actual IDS monitor. Therefore, if an attacker has done enough reconnaissance, or if he is just plain lucky, packets can be sent to a target network in a way such that only the IDS will accept all of them, even the malicious ones. If a hacker can deduce certain features about the target host of his attack, he can send some packets that he knows the host will not accept. In this situation, the IDS monitors do not have the same packet rejection rules as the hosts, and therefore, the attacker can send packets in a manner that essentially skews the view of the IDS monitor, and therefore the attacker can continue launching the attack while "staying under the radar". To clarify, the attacker takes advantage of IP fragmentation by sending a series of fragments that will be reconstructed to form a packet with a payload that has no real meaning and does not appear malicious; however, the attacker can include packet fragments in the stream that the host will reject and thus not include in its reconstruction of the entire packet. Instead, the reconstructed packet may contain an unauthorized system call or other malicious instructions. [9] gives a very detailed description as to the specifics of this attack due to their analysis of the IP code for the BSD 4.4 Operating System.

According to [9], a very simple way to ensure a packet is accepted by an IDS sensor but rejected by target host is through manipulating the header fields of the maliciously formed IP packet. For example, setting an invalid checksum will most likely make the sent packet unacceptable by a host. However, if the IDS monitor is not checking that all packets have valid checksums, a bad checksum will not cause the IDS to signal an alert. Another quick manipulation would be to set the time to live header field to some value where once the packet is passed through the IDS, its TTL value would decrement to zero and would therefore be dropped by a host but accepted by an IDS monitor as valid traffic.

5.2 Evasion

The attack opposite of insertion is evasion. This involves an attacker "evading" an IDS by sending packets that he knows will be rejected by the IDS, and therefore will be unanalyzable, but accepted by individual hosts. In a sense, the attacker is covering his tracks because he's guaranteeing that the packets he sends are flat out rejected by the IDS monitors and therefore, the attacker sends a packet stream with packets that he knows the IDS will reject but that the target host will accept.

This is more difficult to execute than insertion because the attacker must be sure that the IDS monitors are not placed on a hierarchical level above the hosts. In other words, if the IDS monitors are acting as a gateway in deciding what packets the hosts should and shouldn't receive, evasion may not work the way the attacker intends it to. In the event that a monitor is placed on the same hierarchical level as a host, since the attacker knows the packets will

reach the end-user regardless of whether or not it is rejected by the IDS monitors, he can simply create a malicious attack stream that will deceive the IDS but elegantly succeed in exploiting the target host.

Both of these kinds of attacks are indeed clever, but they are based on concepts that were central to older versions of IDSs . These days, IDS monitors are more likely to check for bad checksums on packets and set other header field rules that should prevent potentially malicious packets from escaping its sensors. Still, those NIDSs concerned with increasing detection speed may overlook or even sacrifice many of the necessary configurations for preventing either of these simple attacks. The larger a network becomes, the more bandwidth the NIDS monitors are responsible for handling and therefore tradeoffs must be made. Therefore, according to [9], often the only way to ensure prevention of evasion or insertion is to incorporate a host-based solution. Additionally, the HIDS and the NIDS monitors along with the other hosts on the network must be synchronized and therefore, the responsibility lies on the system administrator who must iron out any inconsistencies. This last point identifies a major concern with IDS technology: the human element. This idea will invariably resurface with every practical and conceptual IDS design as the success in implementing a secure solution is only possible when the administrator of that solution is proactive and responsive.

6 Denial of Service Attacks

A Denial of Service attack is dangerous not only in its potential for incurring network outages but also in the ease of its execution. This attack is aimed at flooding a network entity with usually nonsensical traffic in order to overwhelm the entity to the point that it can no longer service the requests of other users as it is either in a busy/waiting state or has crashed from being overloaded.

An example of the Denial of Service attack involved one of the most notorious and severely prosecuted hackers in modern cyber crime, Kevin Mitnick. In his final black-hat attack Mitnick would take advantage of a very simple flaw in the TCP protocol. This attack is known as SYN flooding and is carefully documented [10] by the man who Mitnick attacked in his final moments. The participants in this attack include a trusted server A, a host B involved in a trust relationship with server A, and an attacking host C. In order to avoid leaving a trail of intrusion, the attacker from host C spoofs the source IP address in the header of the malformed packets he sends, often borrowing an unused IP address on the target network. After careful reconnaissance regarding the trust relationships existing between server A and various other hosts on the targeted network, the attacker will attempt to overwhelm server A with the maliciously formed packets thus exploiting a flaw in the transfer control protocol (TCP) three-way handshake. Pursuant to TCP, in order to open a communication the initiator, in this case host C, must request that the receiver, server A, allocate a buffer in its transfer queue and send a confirmation ACK/SYN packet back to the initiator. If the initiator does not finish the third part of the three way handshake, a predefined timeout value will cause the receiver, under the assumption that the connection has gone bad, to close the communication and deallocate all buffer space in its

queue which relates to the failed communication. The exploit exists in the moment between the second part of the TCP communication initiation and the timeout where, if host C sends an overwhelming amount of communication requests and leaves them in a half-open state, the receiver's queue will eventually fill up to capacity and it will no longer be able to service any other hosts on the network. When the attacker is positive that server A has reached maximum capacity, if he has also determined the sequence that host B uses to increment its own ACK packets, the attacker can spoof the source information of server A and finally establish a trusted communication with host B.

As a result of the media frenzy surrounding Mitnick's arrest and prosecution, the SYN Flood attack has been heavily analyzed by many experts in the field, and is therefore much more difficult to execute on a network with a semi-dutiful security administrator. Even during the time of the original attack, many steps could have been taken to detect and possibly prevent it earlier. A simple precaution to take would be to install a lightweight firewall or filtering router to protect a network from reconnaissance in addition to manually disabling the r-utilities thereby making it nearly impossible for the attacker to develop a threatening level of knowledge of the trust relationships between network entities. [3]

In terms of defeating an IDS, a SYN flood attack can be directed at a network monitor provided the attacker knows which hosts are monitors and the monitors have not been configured to safeguard against the attack. Unfortunately for the administrator of the IDS, Denial of Service attacks can come in even simpler forms than the SYN Flood. Among these forms, one entails sending ICMP packets to a host in order to "ping" it, or find out if it is online and accessible, at such a rapid pace that the host being pinged finally crashes. This attack is known as the "Ping of Death" attack. The seemingly infinite number of ping requests exhausts the resources of a targeted host. When a host's resources are overwhelmed in this manner, it has no choice but to shut down, allowing the attacker to redirect traffic dedicated for the pinged host to another IP address.

Distributed denial of service (DDoS) attacks augment conventional DoS attacks by utilizing multiple sources to attack a host. This reduces the threat of detection to an attacker as he can send less data from individual nodes by spreading out his attack. Additionally, since the attack originates from multiple sources, narrowing down the identity of the cracker becomes more difficult. This is especially true when the attack comes from compromised hosts on other third-party networks (i.e. commercial networks). In this case, the attacker is essentially hiding behind an otherwise trusted network to further distort his identity. In these situations, the organizations of the hijacked third-party networks must also claim a degree of liability for potential flaws in their security policy that lead to the final DDoS attack.

A denial of service attack can be prevented by setting thresholds on various system metrics. For example, recent IDS frameworks have [18] installed packet-filtering schemes on various network nodes that will drop packets from an IP address outside of a given range. Thresholds on other metrics such as delay, loss, and bandwidth consumption place restrictions on a host so that it is not overwhelmed by any other host, whether or not that host resides on the same network. These measurements of network activity allow for quantifiable evidence that a host is potentially under a DoS attack. Section 8 goes into more depth on recent research in this area.

7 The Snort IDS

One of the most well known and widely used intrusion detection systems which benefits from the support of the open source community is the Snort intrusion detection system. Originally created by Martin Roesch, this tool is a signature based IDS that has many advantages for personal computing purposes as well as medium, and in some cases even large, sized corporations. The most appealing advantage of the Snort intrusion detection system is that its free to download and distribute, a liberty that can be attributed to GNU General Public License published by the Free Software Foundation. In other words, Snort can be freely installed on any number of machines just as long any modifications made to Snort's source code are posted and documented for public use.

Another key advantage of Snort is that it is constantly being developed by a community of programmers that range from system administrators, security administrators, analysts, developers, engineers, to other parties interested in maintaining the enormous signature library that currently (as of March 20, 2004) includes around 1,500 prewritten rules. Of course, the success of the tool relies entirely on the administrator and his consistency in upgrading the definitions. Provided that the those responsible for administering this product are diligent in its upkeep, having an entire community of like-minded ethical hackers and problem solvers working at devising solutions to recent attacks is similar to having an entire community of security engineers for the price of an internet connection to a BBS (Bulletin Board System) or Snort mailing list. Even if a black hat decided to post a fraudulent rule to the message board, one can expect that the creators and maintainers of Snort would eventually find the inconsistencies and the intentional deception in the fake rule.

Lastly, Snort is highly portable. Its current version can run on architectures including i386, SPARC, Alpha, and Motorola 68000/Power PC as well as being compatible with operating systems such as Linux, OpenBSD, FreeBSD, Solaris, HP-UX, Win32, Mac OS X, and more. The most current version that has just been released is Snort v2.1.1, which according to its website (www.snort.org) addresses such issues as updating its templates, a fixed port-scan alert bug, removing the escaping of '%' and '-', and many more examples of the ongoing maintenance effort. It is also a relatively small application, only consisting of a few megabytes of space. However, where memory usage becomes a factor is in storing the information that Snort produces, which can be gigabytes. Output generated by Snort can come in the following forms: syslog, tcpdump, Text Logfile, XML, Relational database, SNMP, and Snort Unified. In fact, Snort supports every major relational database platform from MySQL to Oracle and even Microsoft's SQL Server. [11] The fastest among these is the tcpdump option because of the simplicity and locality of the log file whereas, despite its convenience and manageability, the relational database approach can be a serious strain on network performance, a prevalent issue with all NIDSs. The ideal situation involves utilizing the last format on the previous list of available options: the Snort Unified Format and a stand-alone partner process to Snort known as Barnyard. In order to find a solution that allowed their users to have more accessibility to their network performance logs, the initial creators of Snort developed a method whereby the Snort daemon handling network traffic can focus primarily on its intended job of packet analysis by off-loading the storage task to Barnyard. Because Barnyard is its own process, it can run along with the Snort daemon and

do the work of sorting the output data into relational databases greatly reducing the bandwidth strain that would otherwise be unequivocally present in the absence of the helpful plug-in.

Barnyard is one of the many possible plug-ins that Snort offers its vast community of users. A heuristic library has also been added to enable anomaly-based detection to a predominantly rule based system. This library is known as the Statistical Packet Anomaly Detection Engine (SPADE). In SPADE, traffic is assigned a risk metric and when the metric exceeds a certain threshold, that particular portion of traffic is labelled bad, causing Snort to generate an alert. This is effective in preventing low probes from distributed sources because of the use of a metric that will keep track of bad traffic aimed at a host on the protected network, even if it is fingered once a day, or from multiple sources.

However, since one of the most attractive features of the Snort IDS is its simple yet powerful syntax for devising rules, one must actually examine a few samples to understand exactly how the tool works. In downloading the latest version of Snort (version 2.1.2), one can access the complete list of current rule sets that include the following:

attack-responses.rules, smtp.rules, backdoor.rules, snmp.rules, bad-traffic.rules, multimedia.rules, mysql.rules, sql.rules, chat.rules, netbios.rules, telnet.rules, ddos.rules, oracle.rules, p2p.rules, virus.rules, policy.rules, web-attacks.rules, experimental.rules, pop2.rules, finger.rules, porn.rules, ftp.rules, web-frontpage.rules, rpc.rules, rservices.rules, icmp.rules and x11.rules,

Altogether there are forty-eight rule sets along with a few configuration files and other miscellany. The following is an example of what these rules look like and how they are decoded by the Snort signature-matching engine:

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any
(msg:"ICMP PING NMAP"; dsize: 0; itype 8;)
```

The above rule is broken down in this manner: an alert should be generated for ANY ICMP packet that comes from outside the host network, has an empty payload (dsize: 0) and has a type field of 8. The msg tells the admin that the reason for the alert is due to a possible ICMP NMAP PING, which indicates that an attacker is using NMAP, a network surveillance tool, to traverse HOME_NET. The markups in mixed with the syntax allow Snort to parse the alerts that it encounters for easy reporting capabilities.

Snort also allows its users to configure and create new rules that are more appropriate and specific to a client's network, a fairly unique feature not found in most commercial products. Allowing users to write their own rules, which isn't difficult once the syntax for Snort is understood, increases the granularity of an IDS. However, it also puts more responsibility on the security administrator responsible for maintaining the tool. The success of the Snort IDS is directly

proportional to the amount of planning, design, and configuration that the administrator is capable and willing to do.

While Snort is an incredibly powerful tool, it has some significant drawbacks. First and foremost is Snort's complexity in rulesets and plug-in/library options. In this case a blessing can be a curse if security administrator is not entirely sure which services should be run and which ones he can cut. The difficulty is especially apparent in the installation process. Each device must be configured with a certain meticulousity that can cost an organization a sizeable price in man hours as well as leave the system open to a potential vulnerability that a human is quite likely to leave unnoticed. Thankfully there is a substantial amount of documentation available either for free or for a maximum price of \$30 dollars (the going rate for [11] is \$28.50).

Secondly, despite the heuristic help that SPADE provides in anomaly-based detection, "Snort is primarily a signature-matching IDS, and consequently falls victim to the false positive quandary." [11] While steps can be made to reduce the quantity of these pesky false alarms, there are an inevitable evil that must be dealt with. Additionally, because Snort is predominantly signature-based, it will repeatedly be susceptible to newly crafted attacks. Despite how quickly the open source community is capable of reacting, its members obviously can't write rules in real time to catch up with the rate of new attacks.

Finally, Snort's status as an open-source tool prevents it from being used by enterprize-class applications. The GNU Public License may not be enough for the necessary contractual vendor agreement to prove the product's legitimacy in the court and in the board room. An IDS is an integral part of the protection of valuable company assets and certain liability issues are often only handled best through a more proven framework than the relatively young, liberally minded Open Source Initiative and the Free Software Foundation. However, based on recent corporate decisions to invest in Linux boxes for production servers, corporations have been visibly migrating toward the open source solution.

All shortcomings aside, Snort is commercial friendly while retaining a vendor-free characteristic that lends to it an unbiased level of respect. Pride in a truly successful product in addition to a recurrent level of communal interest in solving a plentiful supply of intrusion detection challenges will continue to drive the development of this tool while sustaining its popularity both in the office building and on the home pc.

8 Recent Advances in IDS Technology

The following papers are all recent attempts at solving some of the aforementioned problems with conventional IDSs. Each of the following contributions have been made within the last two years. They demonstrate that there is still potential for improving and extending the capabilities of an intrusion detection system.

8.1 Detecting Service Violations and DoS Attacks

This paper proposes a method for detecting the Denial of Service attacks mentioned in **Section 6** and another attack of similar consequence known a quality of service (QoS) attack. The difference between a quality of service attack and a denial of service (DoS) attack is in the aim of the attacker. In a network where a QoS scheme is present, users have varied service rates based on a payment plan or special privilege. For example, if a user pays an extra monthly fee to his internet service provider (ISP) for an extra service (i.e. streaming video), he is entitled to a bigger and faster share of the available bandwidth. The motive of a QoS attack is to gain access to a higher class of service for recreational use or, worse yet, for orchestrating an attack on a larger network that would require more efficient bandwidth capabilities.

The authors of [12] believe that either attack can be quickly detected and possibly prevented by their proposed scheme. Contrary to basic detection schemes that simply wait until network performance drops abnormally, the new method presented in this paper attempts to prevent this situation from even occurring by focusing on early detection of such attacks. Additionally, the developers of this detection scheme aim to incur retribution on those responsible for the attacks.

The contributions made by this paper are techniques for monitoring network behavior in order to detect DoS and QoS attacks early enough to avoid the detrimental effects of a network outage. An outage in this sense refers to the inability of a network entity or service to function correctly. The three parameters in the decision making process of the presented techniques are delay, packet loss ratio, and throughput. All three service level agreement (SLA) parameters define the characteristics of service level behavior. Although it is not the original work of [12], the first technique for network monitoring is known as core-based monitoring. A core router is otherwise known as an ingress router or one that is placed within a network whereas an edge router, also referred to as an egress router, is one placed on the outside. Core-based monitoring involves aggregating SLA statistics from core routers. These statistics come from randomly selected packet headers saved by a core router, where random is defined by a given probability function. From the saved packet headers, ingress routers form special packets known as probe packets that are sent to egress routers, which compute the network delay. Core routers compute the other SLA parameters in a similar fashion.

The significant drawback in this solution is the amount of overhead required to generate accurate statistics. The overhead hit comes from the stain placed on the core routers in having to compute SLA statistics through marking random packets. Since the focus of this paper is quick detection of a potential DoS attack, the authors introduce distributed monitoring, which fits under the umbrella of another known technique for SLA statistic computation: edge-based monitoring, a method similar to core routing except in the way it measures the packet-loss ratio. Another approach that fits in the category of edge-based monitoring is stripe-based monitoring. Stripe-based edge monitoring involves sending a series of consecutive packets known as a stripe to various nodes on a network to compute SLA parameters in a more direct manner. In monitoring the packet drop rate of the stripe, a network monitor can make quick and accurate conclusions in regards to the proper behavior of a network. Stripe-based monitoring accomplishes all of

this through inferences made by the results of the stripes it sends to edge routers and not core routers thereby reducing overhead costs.

In distributed monitoring, overhead is further reduced by placing monitors on an overlay network which resides over the physical network. In this environment, the monitors have direct access to each other. They regularly probe the underlying network in sending stripes similar to those sent in stripe-based monitoring. However, these monitors are primarily interested in finding congested edges, or links, as opposed to exact service level agreement values. If a monitor finds that one of its edges to another node is congested, it will trigger other monitor nodes to test that link for congestion. In short, this scheme reduces to number of node comparisons to $O(n)$ whereas striped-based requires comparisons on the order of $O(n^2)$. This is because of the direct connection of edge routers in the distributed scheme which allows for detecting congestion in two directions as opposed to the restriction found in stripe-based monitoring where only malicious traffic travelling in the same direction of the stripe can be detected, potentially requiring $O(n^2)$ in a worst case scenario.

Each of these schemes incurs overhead as a result of its implementation, however, in theory they should still be able to balance this overhead with SLA parameter computation to effectively uncover a potential DoS or QoS attack scenario before it materializes. In a core-based solution, because each monitor incurs high levels of individual overhead, it is not a scalable solution unlike the route-based approach featured in stripe-based and distributed monitoring. Regardless, each monitoring scheme can in theory uncover DoS and QoS attacks. The promising arguments made in this paper suggest that research on the practical implementation of these monitoring schemes should continue and that DoS and QoS attacks are detectable at their early stages and potentially preventable.

8.2 A Virtual Machine Introspection Based Architecture for Intrusion Detection

Another recent development is [7] a paper that describes a unique approach to intrusion detection, one that attempts to solve two reoccurring problems with traditional host-based and network-based intrusion detection systems, namely evasion and direct attack. To combat these vulnerabilities, the authors of this paper develop a unique framework for an IDS built with Virtual Machine Monitor (VMM) technology. One of the best ways to prevent evasion from obstructing the view of an IDS monitor is to ensure good visibility. In other words, by increasing the range of analyzable events, an IDS monitor will have a better perception of individual host activity. However, this approach leaves the IDS monitor more vulnerable to direct attack. That is, without any redundancy, if a monitor is compromised, intrusion detection is no longer possible until the monitor is restored to a working state. This leaves a system in a fail open state in which an attacker's actions will not be recorded for post incident response analysis assuming the attack is eventually detected.

Both of the dominant intrusion detection frameworks suffer from poor visibility (NIDS) and susceptibility to direct attack (HIDS). As such, it is necessary to develop a system that can

remedy these problems while retaining the desirable properties of either framework. Virtual Machine Monitor technology accomplishes this by using an introspective approach that the authors of [7] deem Virtual Machine Introspection (VMI). VMI does not require an IDS monitor to physically reside on a host in order to analyze states and events. Instead, because a virtual machine is software that abstracts the hardware resources of a host, a monitor running on a virtual machine is isolated from the actual hardware itself. This method of isolation prevents attackers from taking advantage of the vulnerability of IDS monitors residing at the same level as host's operating system. Because the IDS monitor is no longer directly tied into the host's operating system, it can exist in a network of distributed virtual monitors while retaining a high degree of visibility.

Additional isolation between host and monitor comes from the difference in programming languages in the implementation of the virtual monitor and the actual operating system. For example, a monitor can provide surveillance for a host running a Linux operating system even if it is uniquely programmed by a different language because of the level of abstraction in the interface between a virtual and a physical machine. Since prior research of virtual machines exists, a virtual machine API is accessible.

VMM merely leverages a third-party host OS to provide drivers, bootstrapping code, and other functionality common to VMMs and traditional operating systems, instead of being forced to implement all of its functionality from scratch.

The virtual machine monitor is the entity responsible for the virtualization of system resources for virtual machines, and therefore is the most important entity within the IDS. It communicates with the VMI IDS, which is the virtual machine introspection intrusion detection system, through a protocol that further secures the VMM from attack. Even an attack on the VMI IDS is tolerable as long as the VMM retains visibility. All traffic into the VMM is assumed to be malicious and is therefore processed by the VMI IDS, just a host running in promiscuous mode would see all traffic on an Ethernet network. In addition to isolation, two other properties that the VMM approach provides and IDS with are inspection and interposition. Inspection refers to the VMM's ability to know the exact state of a virtual machine, which makes it difficult to evade the IDS. The property known as interposition allows the VMI IDS to utilize the VMM's ability to intervene in the execution of privileged instructions of its virtual machines in order to detect illegal actions. [7] gives an example of such an action as an illegal modification of a given register.

There are potential drawbacks to this system. The first is in modifying the virtual machine monitor in order to extend the functionality for the VMI IDS. If the VMM is modified incorrectly, the overall system is prone to a security breach.

In confronting this issue in our prototype system, we provided additional functionality by leveraging existing VMM mechanisms.

A second drawback is deciding which machine events to handle based on their potential for incurring overhead. For example, some events require a minimal step such as copying or logging,

whereas other events, often involving memory access, require that the VMM take a performance hit which has a direct effect on the rest of the network. The developers handle this drawback by predefining events that they know will decrease performance but must be monitored because of their potential to be malicious. A final drawback is in the potential exposure of the VMI IDS because of its relationship to the VMM and a given virtual machine. The main idea is that since the developers allow the VMI IDS to have access to a given virtual machine's state, it is exposed to attack via a successful compromise of a virtual machine. And, because the VMI IDS is allowed close access to the VMM's code, the VMM can also be exposed to attack via a successful compromise of the VMI IDS. Thus, the VMI IDS and the VMM have their own secure method for transportation to provided added security to the VMM, an entity that must be defended above all other entities.

The policy engine responsible for recognizing and responding to malicious behavior is built into the VMI IDS. Through an interface provided by the VMM, the VMI IDS can request machine state information and then utilize its policy engine to conduct analysis on the information. The engine includes a framework for developing high-level intrusion detection APIs and a module for acting on the APIs. An example of such a module is a polling-module that checks a virtual machine, via the VMM interface, for malicious behavior on a periodic basis, where the policy APIs specify what should be checked.

There are additional weaknesses with this solution that deal with flaws in the VMM. These include an exploitable network stack, the detectable presence of a VMI IDS in noticing differences between execution times of virtual, (VMM calls) and non-virtual processes, errors in the actual code of the VMM, and insecure communication between the VMI IDS and the VMM. Attacks on the IDS itself include modifying the operating system interface library of the OS running an IDS monitor, or directly attacking it through a buffer overflow or inducing resource exhaustion. However, the fact that this framework utilizes a third-party OS to run the virtual machines makes it less susceptible to attack.

An important point about the discoveries and contributions made in [7] is that other IDSs may also lend themselves to a virtual machine introspection based architecture, establishing a new approach toward hardening all IDSs. If the developers of this architecture allowed the failures of some commercial IDSs to discourage their research, they made not have made these breakthroughs. Yet, in leveraging previous research regarding virtual machines, they were able to devise an improved IDS scheme that has some very desirable features.

8.3 Global Intrusion Detection in the DOMINO Overlay System

Briefly mentioned in Section 4, distributed intrusion detection systems (DIDS) are IDS hybrids that combine the four classifications of intrusion detection: misuse detection, anomaly detection, host-based intrusion detection systems, and network-based intrusion detection systems. Naturally, the nodes that comprise a distributed intrusion detection system must be able to communicate with each other so it is a given that a DIDS will be comprised of at least NIDS monitors.

There are several papers [2, 20, 21] that introduce unique approaches toward implementing DIDSs. At the 2004 Network and Distributed System Security Symposium (NDSS '04), authors Vinod Yegneswaran, Paul Barford, and Somesh Jha presented their creation: Distributed Overlay for Monitoring InterNet Outbreaks (DOMINO)[18]. The contributions provided in this paper include a new approach toward distributed intrusion detection, a method for monitoring unused IP addresses to prevent spoofing, and a thorough evaluation of an IDS through use of extensive test data (four months of data from 1600 different networks).

DOMINO is built on top of an overlay network, much like the distributed scheme for detecting DoS and QoS attacks in [12], which means it has several desirable features. The topology of the network under surveillance is assumed to be heterogenous giving added flexibility to the DIDS. It also provides decentralization which increases the reliability of the IDS in that if one monitor is compromised, other monitors should still be able to function normally. Another important feature of DOMINO is that it can aggregate global attack information increasing the overall perception of the IDS. As is the case with most forms of DIDSs, DOMINO has a hierarchical structure comprised of layers whose nodes have intrusion detection responsibilities different than that of other nodes elsewhere in the hierarchy. In the case of DOMINO, the nodes that make up the three tier hierarchy are termed axis overlays, satellite communications, and terrestrial contributors.

Axis overlay nodes do a majority of the information sharing in the DOMINO intrusion detection system. These nodes are assumed to be continually joining and leaving a network in the event of failure or individual configuration. Each axis node is comprised of several components. The first among these is a activity database that stores global and local intrusion detection data. Data found in these decentralized databases is transferred from node to node in order to ensure that all nodes on the network can be alerted in the event of an attack. Another component to this node is an active-sink, which is a sniffer that specifically monitors unused IP addresses. In preparing for and executing an attack, an attacker may search for an unused IP address on a target network that they can spoof. However, if an IP address is deemed unused, DOMINO monitors will trigger an alert if it generates any traffic. Other components that make up axis overlay nodes are NIDS/Firewall rulesets, a query engine for gleaning real-time data from the individual host if necessary, and a communication protocol for the periodic exchange of intrusion detection data.

Satellite contributors are nodes that are responsible for a given locality in a network. Because of their limitation to a specific area on a network, their information is not necessary reliable in predicting the current situation of the entire network. Instead, they are important in contributing the incident response phase of an IDS where in the event that an axis overlay node detects malicious behavior, satellite contributors can provide more specific information about the area supposedly under attack.

The least trustworthy source of network data comes from the terrestrial contributors. Their primary task is to collect raw network data and to pass it to DOMINO nodes higher in the hierarchy. These nodes are not considered intrusion detection monitors. Instead, they are thought of as simple packet loggers that can store all kinds of traffic information.

To test their DIDS solution, the developers of DOMINO simulate two major networks attacks: the SQL-Snake worm and SQL-Sapphire worm (also known as the SQL-Slammer). The desired result of the SQL-Snake simulation was to measure DOMINO's effectiveness in reaction time and alarm rate. The method used for generating alarms was a voting scheme where based on the following metric:

A node votes for an alarm if the following holds:

- 1) 200% increase in number scans from the hourly average, and
- 2) 100% increase in the number of sources from hourly average, and
- 3) number of sources is greater than five.

Since this attack leverages multiple sources in order to further propagate itself, training nodes to recognize an increase in this kind of behavior as potentially dangerous is an efficient way of identifying the location of an attack. As a result of DOMINO's thorough communication scheme, a security administrator will have an accurate view of the network behavior and will be able to react quickly to remedy the situation. After running this simulation, the developers of DOMINO concluded that in adding enough nodes to their system, the outbreak of the SQL-Snake could be quickly detected with a low false positive rate. The SQL-Slammer simulation allowed them to come to similar conclusions in attack detection.

Another nice feature offered by the DOMINO IDS is known as IP blacklisting, a preventative measure for identifying reconnaissance on a network. The developers of DOMINO found in their research that a small number of IP addresses are often the source of a majority of scans and attacks. Therefore, in keeping an active list of these IP addresses, during the incident response phase security administrators can better narrow down the source a potential attacker.

In the end of their paper, the authors highlight usual IDS vulnerabilities and offer their approach toward solving them. In terms of combating a denial of service attack, DOMINO can remedy two potential situations. When a participant of the DOMINO IDS is under a DoS attack from an exterior location, the authors believe that the distributed nature of DOMINO allows for nodes to be compromised. The authors claim that preventing DOMINO nodes from being compromised is a "non-goal." In the event that the DoS attack is coming from a compromised node, filters that reside on DOMINO nodes have thresholds set that prevent it from becoming overwhelmed by any one source that either resides on or off a network.

The second category of IDS vulnerability that [18] addresses is infiltration, which is another term for masquerading as an IDS monitor. An authentication protocol that DOMINO requires its axis nodes to participate in requires that an attacker know how to compromise both the axis node and the authentication scheme they use to communicate.

Finally, several features of DOMINO can potentially remedy IDS monitor obfuscation. For example, the aforementioned authentication scheme between DOMINO includes adding SHA-1 digests (a hash algorithm) to ensure that tampered packets are easily recognizable. Additionally, the distributed nature of DOMINO and the filters on its nodes prevent individual nodes from

obfuscating network sensors through increasing data volume. The fact that DOMINO is a collection of multiple localities and not just one entire network gives it more perceptibility and granularity against stealthy reconnaissance attacks. Lastly, because DOMINO pays attention to both live and unused IP address, there is no part of the network that goes undetected.

DOMINO is another example of IDS research that has real promise. A potential issue could be the expense of the system given that it is such a large-scale solution. What is interesting about DOMINO is that it combines the power of network firewalls with intrusion detection to provide and optimal security environment in the form of an IDS, which is opposite to the predictions made by Gartner's Richard Stiennon. We feel it should be interesting to monitor the progress of DOMINO and its potential for practical use in a commercial setting.

8.4 Building Attack Scenarios through Integration of Complementary Alert Correlation Methods

Another paper presented at the NDSS 2004 conference presents a method for aggregating attack scenarios in order to provide a more accurate depiction of the combination of events that comprise an overall attack. The authors of [19] found that individual attacks can be correlated through integrating causal relationships and identifying similarities in attack attributes. In recognizing the nature of these relationships, they can construct a high-level representation of the probable sequence of events in a large-scale attack.

The first method for correlating attack sequences examines prerequisite and consequential data surrounding individual attack scenarios, where prerequisite data refers to system conditions necessary for an attack to occur and consequential refers to system conditions that result from a given attack. To further emphasize on how one would link attacks based on predicates (prerequisites and consequences), the authors develop a series of logical formulas. They define a hyper-alert type as a triple made up of a fact, prerequisite and a consequence. Hyper-alerts are grouped based on similarity in facts where the prerequisite condition is what must come true for the consequence to hold. The idea of this model is to establish a chain of prerequisites and consequences such that a later hyper-alert can be traced back as a consequence of an earlier alert. Utilizing some basic set theory, the authors define the following collection of sets for any hyper-alert type T : $\text{Prereq}(T)$, $\text{Conseq}(T)$, and $\text{ExpConseq}(T)$ (expanded consequence set of T which includes all predicates implied by the set of consequences for T). The IDS correlates hyper-alerts based on the relationships between these sets. To clarify, a hyper-alert h_1 is said to prepare for h_2 if for some prerequisite p of h_2 , there exists a corresponding predicate c in the set of expanded consequences of h_1 that occurs before p . In other words, a predicate that is a consequence of one alert can be an indirect cause or prerequisite for a following alert. The most appropriate method for representing these relationships is through acyclically graphing nodes connected by edges where the nodes represent individual attacks and the edges represent causal relationships between nodes.

In the event that an IDS fails to detect an individual attack, two clusters of correlated alerts are linked together in the same manner that the IDS correlates individual hyper-alerts. Therefore,

while several clusters may appear disjoint, there remains the possibility that one of these clusters prepares for the other. Augmenting correlation graphs is also possible by matching attack sequences with similar attributes. For example, if there exists two attack sequences that share an similar prerequisite and consequence, even though the sequences in between both predicates may differ for either attack sequence, they may be combined in the overall correlation graph as both sequences are equally likely.

The major contribution of [19] is in its method for providing security administrators and forensics experts with a more intuitive engine for hypothesizing the specifics of an intrusion, rather than requiring them to sift through tcpdump logs or other potentially large network data repositories. Even if the IDS misses a step in the attack sequence, given that enough the IDS collects enough data to provide some level of correlation, hypotheses based on these correlations may help investigators uncover the missed step in other event logs. This greatly increases the efficiency of the investigation and the probability of apprehending and prosecuting the responsible individual.

9 Conclusion

In this paper, we provide an introduction to intrusion detection systems by charting their history and explaining their various classifications. Additionally, we present some of their vulnerabilities and how they can be avoided. Finally, we document current research that attempts to remedy these vulnerabilities to demonstrate the potential for hardening these systems. To conclude this paper, we give three reasons as to why we believe that use and research of these devices should continue.

Reason 1: Preventing the Repetition of History

The importance of intrusion detection systems such as Snort, SHADOW, and others that base their methodology on matching attack signatures against previous attacks is that they prevent history from repeating itself. After Code Red, Nimda, SQL Slammer, SQL Snake, and other large scale, media-hyped attacks ravaged the globe, the need to patch other potentially vulnerable systems became suddenly imminent. In order to prevent carelessly written and rushed-to-production proprietary code from being repeatedly exploited, it is necessary that researchers continue to examine the attack patterns and design an IDS solution, which sometimes is as easy as posting a new rule to a message board, to ensure that those attacks do not reoccur. With so much at stake in the compromise of current networks, a solution in the form of a rule-based IDS appears to be the easiest and most supported.

Reason 2: Aiding the Post-Attack/Forensic Effort

With computer crime increasing on an annual basis, the probability that an organization will suffer an attack also rises. It is almost a given that most companies, large and small, will experience some sort of system breach ranging from the theft of a personnel phone book to 50,000 client credit card numbers stored on a centralized database server. Therefore, careful preparation is vital to maintaining a reliable level of business survivability. When the inevitable breach occurs, the IDS can often be the most reliable, and sometimes the only thorough depiction of what actually happened. When the forensics experts are summoned to the scene of a cyber-crime,

their first sources are the IDS logs, provided the attacked network had an IDS in place. A good example of how advanced IDS systems can really aid in the investigation of a break-in comes from the paper mentioned in [19] that introduced a method of associating network events as members in a set of causal or correlative relationships thereby depicting a possible step-by-step reenactment of the attack(s). This paper proposes an framework where security administrators can graphically piece together probable attack sequences to nail down the specifics of an attack and hopefully use that information to find and prosecute those responsible. Pursuant to Reason 1, a good IDS can allow the real security experts to further assess the current network configuration (access policy, monitor placement, etc.) to prevent a repeat occurrence.

Reason 3: Adding Another Measure of Security

No successful security system is comprised of only one security device, no matter how inclusive that device is. The main contenders in the argument that renders IDSs as obsolete devices are known as firewalls. Firewalls come in both hardware and software form and are usually placed at the gateway of a network in order to have control of which packets may enter a network or on individual hosts to ensure a higher granularity of security. While these devices have certainly proven their worth, even more so than intrusion detection systems in many regards, they too are vulnerable to evasion with more serious consequences. Firewalls are said to be figuratively comparable to a candy bar: crunchy on the outside but chewy on the inside. In other words, firewalls are very tough to beat and are becoming more resilient with each version. However, once they are surpassed, if there is no other form of defense within network they protect, a potential attacker can open a hole that will allow him to have considerable access to network resources (the chewy inside) without having to worry about being detected by any other device except the actual users utilizing the system. Thus, if Tsutomu Shimomura only employed a firewall on his personal system and decided to leave out the IDS that mailed him his activity logs, catching Kevin Mitnick would have been much more difficult, if not impossible as Shimomura was on vacation at the time of the attack and Mitnick had been known for his slippery escapes. Additionally, as was mentioned previously, internal attacks make up a sound majority of successful computer theft, and according to [11],

proxy-aware instant messengers, such as AOL Instant Messenger, can be used to slice through any open port on a corporate firewall...Most non-technical users may be unaware that they are creating a gaping security hose by going about their daily activity.

The point is that a sound security system needs intrusion detection as part of the solution. Encryption, cryptography, access policies, firewalls, and honeypots are individually not enough to provide effectual system security. The flaws in IDSs should invigorate those involved in the research effort to improve its technology and to continue developing unique approaches at combatting computer and network breaches. However, just as relying solely on a firewall for overall system security, relying on an IDS in a similar manner is also unwise as they are not yet capable of being a network wide solution. Network traffic along with new attack methods are too unpredictable and their scale of intensity increases with each passing year. The fact that a

successful IDS can be downloaded and distributed freely under the Open Source Initiative should be enough to convince organizations that the manual labor cost of installing and maintaining an IDS is not only sensible but necessary. Until firewalls reach the point where they can accomplish what current IDSs can through combining the two technologies in a hybrid device, intrusion detection systems are necessary components in ensuring the best solution in securing a network. Only when each of these elements are tactfully combined, can a system be truly secure. Koziol puts it well in [11] :

An IDS is a critical component in a defense-in-depth information security strategy. Defense in depth is the method of protecting information resources with a series of overlapping defensive mechanisms. The thought is that if one defense should somehow fail, others will be in line to thwart an attack.

Perhaps it is the intimidating forms or complexities that characterize various designs of intrusion detection systems that make them unmanageable and therefore ineffective. It could also be the fact that some commercial IDS vendors distribute a product that is not rigorously tested. These vendors market a dangerous sense of security that potential buyers may not recognize. On top of that, closed source solutions are expensive to purchase and tricky to install. However, intrusion detection is a technology that continues to challenge both attacker and defender alike to theorize, practicalize, analyze, and experiment. What is exciting about this technology is that it will never be perfect. As long as the human race relies on a digital medium to reposit vital assets, there will always be an infallible race condition between the next attack and the next intrusion detection solution. The results of recent breakthroughs are promising enough to justify the need to continue the development of intrusion detection systems.

References

- [1] James Anderson *Computer Security Threat Monitoring and Surveillance*. James P. Anderson Co. April 15, 1980. Contract 79F296400.
- [2] Phillip A. Porras and Peter G. Neumann *EMERALD: Event Monitoring Enabling Response to Anomalous Live Disturbances*. Computer Science Laboratory. SRI International, 1997.
- [3] Stephen Northcutt and Judy Novak. *Network Intrusion Detection* New Riders Publishing. 2003.
- [4] Wayne Jansen, Peter Mell, Tom Karygiannis, Don Marks. *Applying Mobile Agents to Intrusion Detection and Response*. National Institute of Standards and Technology. October 1999.
- [5] W. Jansen, P. Mell, T. Karygiannis, D. Marks. *Mobile Agents in Intrusion Detection Response*. National Institute for Standards and Technology, 2000.

- [6] Seth Robertson, Eric V. Siegel, Matt Miller, and Salvatore J. Stolfo. *Surveillance Detection in High Bandwidth Environments*. In Proceedings of the 2003 DARPA DISCEX III Conference. April, 2003.
- [7] Tal Garfinkel and Mendel Rosenblum. *A Virtual Machine Introspection Based Architecture for Intrusion Detection*. Network and Distributed System Security Symposium. February 6-7, 2003.
- [8] Thomas H. Ptacek and Timothy N. Newsham. *Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection*. Secure Networks, Inc. January, 1998.
- [9] Yu-Xi Lim, Varun Kanotra, Nitin Namjoshi, and Seng Oon Toh. *Wireless Intrusion Detection and Response*. Georgia Institute of Technology School of Electrical and Computer Engineering. December, 2002.
- [10] Tsutomu Shimomura. *Technical details of the attack described by Markoff in NYT*. Wed, 25 Jan 1995 12:36:45 GMT. comp.security.misc,comp.protocols.tcp-ip,alt.security.
- [11] Jack Koziol. *Intrusion Detection with Snort*. SAMS Publishing. Indianapolis, May 2003.
- [12] Ahsan Habib, Mohamed M. Hefeeda, and Bharat K. Bhargava *Detecting Service Violations and DoS Attacks* NDSS 2003. CERIAS and Department of Computer Sciences Purdue University.
- [13] Dorothy E. Denning. *An Intrusion-Detection Model*. IEEE Transactions on Software Engineering, Vol. SE-13, NO. 2, February, 1987, 222-232.
- [14] Steven A. Hofmeyr and Stephanie Forrest. *Immunity by Design: An Artificial Immune System*. Proceedings of GECCO, 1999, 1289-1296.
- [15] The Shadow Team. *SHADOW Version 1.8 Installation Manual*. Naval Surface Warfare Center Dahlgren Division. 25 April 2003.
- [16] Mark Cooper, Stephen Northcutt, Matt Fearnow, Karen Frederick. *Intrusion Signatures and Analysis*. New Riders Publishing. 2003.
- [17] John Bashor. *Software That Detects Hackers Helps Catch Big Leage Intruder*. Science Beat. Berkeley Lab. March 29, 2000.
- [18] Vinod Yegneswaran, Paul Barford, and Somesh Jha. *Global Intrusion Detection in the DOMINO Overlay System*. Computer Sciences Department, University of Wisconsin, Madison. NDSS 2004.
- [19] Peng Ning, Dingbang Xu, Christopher G. Healey, and Robert St. Amant. *Building Attack Scenarios through Integration of Complementary Alert Correlation Methods*. Cyber Defense Laboratory. Department of Computer Science. NDSS 2004.