Sound the Alarm!
A Survey of Modern Intrusion Detection Methodologies

Erin Jamroz University of Puget Sound
1500 N. Alder St.
Tacoma, WA
ejamroz@pugetsound.edu

Jacob Fuhrman University of Puget Sound
1500 N. Alder St.
Tacoma, WA
jfuhrman@pugetsound.edu

ABSTRACT

1. INTRODUCTION

In a world of increasing connectivity and nearly ubiquitous technology, the importance of computer security has grown with the user base it has sought to protect. As the "smartphone" has come into such prevalent use in the U.S. and abroad, millions of people use these minicomputers daily. In addition, users of modern technology interact with tens or hundreds of other devices each day, from wireless networks, to connections across their devices at home, to accessing content on the web. With daily access to these services becoming so important to such a massive user base, it is only natural that they would become attractive targets to attackers, and thus all the more important to defend. There are many approaches available to defending a given system or a network of systems, but one that has risen in scientific interest lately is the Intrusion Detection System (IDS), a software suite intended to detect attacks made against a network or system and notify those charged with protecting it.

Intrusion detection is the process of detecting unauthorized use of a system and alerting the proper authorities of such misuse. Intrusion detection systems (IDS) are the systems used to to detect these misuses and aid in their defense.

1.1 Audit Systems

This section will breakdown IDSs by their audit system, which describes the source of the data that they analyse for intrusions. There are three main types of audit systems: *Host-Based*, *Network-Based*, and *Application-Based*.

1.1.1 Host-Based

Host-Based IDSs (HIDS) are characterized by having detection sensors on each individual machine within a network. These types of systems monitor an individual computer's system logs to detect attacks. HIDSs are able to detect attacks with much more specificity than network-based systems because they are able to monitor individual processes on an particular machine for malicious activity.

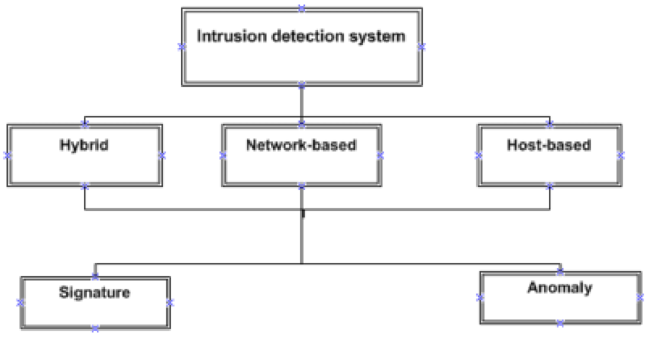


Figure 1: A general breakdown of IDSs by Audit System and Event Analysis. Image taken from [2]

There are many advantages to using a HIDS. First, and fore most, they are able to detect attacks that network-based sensors cannot, because they are running on the host machine. Next, unlike network-based sensors, they are fully functional within both switched, and encrypted networks because their audit source is independent of network traffic, and they are able to use host resources to decrypt traffic before analysis.

1.1.2 Network-Based

Network-based IDSs (NIDS) are characterized by having detection sensors placed at network hubs, such as routers, rather than on each individual machine in a network. These types of systems work by monitoring network traffic looking for patterns in flow or analyzing packet headers for trends. These are the most commonly used types of systems today. It is worth noting that this category of IDS can be further subdivided by type of network data analyzed [5].

NIDSs have many advantages over a host-based system, mainly that fewer sensors are required to cover the entire network, which also means less installation time and maintenance. These types of systems also have little to no impact on network performance, because they are sensors that traffic is simply routed through. As they are their own independent system residing on the network, they can be set up in such a way that they are invisible to outsiders. This hidden nature also makes them very insulated from attack themselves, which is a large advantage for network administrators.

However, these systems are not without their drawbacks. It has been shown that in periods of high network load, these systems performance drops due to an inability to process all incoming packets. As such, they could fail to detect an attack that was launched during one of these periods. The effectiveness of these types of systems also drops in switched networks, because switches subdivide network traffic such that a NIDS can no longer monitor all traffic on the network because much of it is hidden behind a switches. Finally, NIDSs are are unable to analyse encrypted traffic. This incapability is a major drawback, especially in this day in age, as more and more traffic is being moved to encrypted channels of communication.

1.1.3 Hybrid System

1.2 Event Analysis

This section will further subdivide and discuss IDSs by event analysis, which describes the method of detection used, given a particular audit source of data. These detection methods fall into two major categories: *Signature Detection* and *Anomaly Detection*. Signature based methods are much more common in practice, but anomaly detection is an area of much research and promise. A summary of both strategies can be found in 2.

1.2.1 Signature Detection

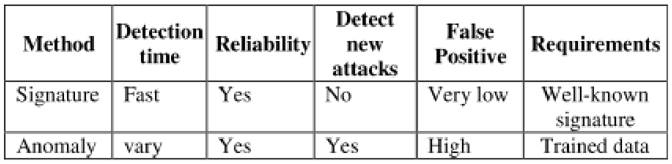
Signature Detection is a complex subject, but can be easily summarized without losing critical details. This section is largely a summary of simple descriptions of the principles of Signature Detection from [34], combined with examples for illustration from [20]. To begin, Signature detection systems operate on an extremely basic principle: attacks that have been detected before should be recognizable after initially being discovered, assuming that whatever unique feature(s) that identify them are written down and matched against. Thus, a Signature-based IDS is one that maintains a database of attack signatures and matches against said database to flag unauthorized or potentially harmful activity on a system, determining if it is an attack at all (and which one it may be of thousands) by examining it side-by-side with each signature.

As an example, consider an exploit of Suidperl, a version of Perl that allows the execution of scripts that change user IDs and/or group IDs. Early versions of the Suidperl interpreter do not properly relinquish root privileges when changing its effective user and group IDs. This allows the creation of a (two line!) script that simply presents the user with a root shell after execution. There are two main approaches to recognize this attack with a signature based IDS. The first is the more simple one, which is searching for strings that have no real use outside of such an attack, and thus can be used to uniquely identify a script that is intended to circumvent access control. The second is to analyze the system and confirm that a valid user-to-root transition has been made, as such sudden access control changes can be clearly identified in file system logs. It is worth noting at this juncture that though both of the above methods are valid to detect the example attack, and both are signature-based, deciding which approach to take is likely based on whether or not the IDS in use is a HIDS or a NIDS. The string-based identification method is better suited for a NIDS, as it can search network activity for strings such as ">= 0;<=0;' or 'exec (Ãę/bin/sh');" that in all likelihood are an exploit attempt and not valid traffic. In contrast, a HIDS could check file systems logs and note that a root shell was spawned without any type of valid user to root transition, and notify authorities accordingly.

Naturally, no detection framework is perfect, and signature detection, by design, may contain critical flaws depending on how it is implemented. As one might expect, such flaws revolve around the fact that signatures are user-defined and maintained, and firm definitions of what uniquely identifies a signature and what does not is subjective to any given user. Therefore, it is easy to imagine the creation of a signature that defines and identifies a given attack with the utmost precision, but to a fault, such that the signature is so specific that it cannot identify even slight variations of the same attack. Similarly, though probably not as common to see as the previous flaw, signatures can be defined too nebulously. If a signature is not specific enough to a certain attack, and too broadly identifies common attributes between multiple attacks, it is just as problematic as a signature that cannot detect variations of the same attacks.

1.2.2 Anomaly Detection

Anomaly detecting systems work by constructing a system profile of "normal" behavior, and use that profile to then detect abnormal behavior. The idea is that attacks are a subset of abnormal behavior, which will allow these types of systems to detect them. Most of these systems use some complex statistical analysis to determine when activity differs from the system profile. The major draw to these types of systems is that, unlike signature detection methods, these systems can detect not only variations on known attacks, but completely novel attacks as well. The problem with this functionality is that it is predicated on having an effective system profile, which is by no means trivial to produce. They require extensive training sets to learn from, of which few exist. In practice, these types of systems produce large numbers of false positives that require manual inspection to effectively classify as an attack or not. For this reason, few existing systems use this type of detection. Instead IDSs using this type of detection mechanism are more an an open area of research than production level systems.



Method	Detection time	Reliability	Detect new attacks	False Positive	Requirements
Signature	Fast	Yes	No	Very low	Well-known signature
Anomaly	vary	Yes	Yes	High	Trained data

Figure 2: Summary of Event Analysis systems. Image taken from [2]

2. RELATED WORK

In this section we present our categories of attack summarize multiple authors approaches to detecting attacks within those categories.

2.1 Probes

This section is will discuss various approaches to detecting Probe attacks. Probes have been classified as any passive information gathering intrusion. The most common form of probe attack, one which will be discussed at length, is the port scan. The motivation for detecting these types of scans is that any intelligent attacker wishing to launch a successful attack will gather background information about the victims system before launching his or her attack. Their primary means of gathering information is by first launching a port scan.

There are 65,536 standardly defined ports on a given machine. These are broken down into three large categories: *a.* Well-known ports(0-1,023), *b.*Registered ports (1,024-49,151), *c.* Dynamic and/or private ports (49,152-65,536) [Matiti, P Lecture notes from Surveying Port Scans Methodologies]. Essentially, a port scan consists of sending a message to each of these ports and analyzing the response message to gain information about what services the victim machine is running. For this reason, TCP ports are the most often scanned because TCP is a connection-oriented protocol, meaning that its response messages are more useful [29]. Added to this, it is easy to block/detect UDP traffic at the firewall level [5]. In general, like most attacks, port scans can be broken down into to high level categories: single source, and distributed source. Within these categories, [31] further divides each of these categories into horizantal, vertical, stobe, and block scans based on the number of machines and ports scanned. A summary of these categories can be found in 3.

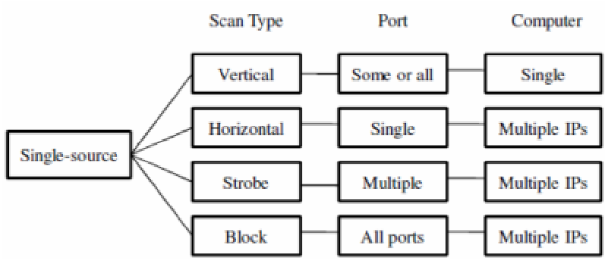


Figure 3: Summary of Port Scans. Image taken from [5]

There is a great deal of research into detecting port scans. Several prominent authors' work will be discussed in the following sub section. They have been organized into signature-based, and anomaly-based detection scheme. [5] presents a much more rigorous taxonomy of detection methodologies that we leave up to all interested parties to explore. We felt that many of the systems they discuss belong to several of their categories and that a higher level taxonomy made the distinction clearer.

2.1.1 Signature-Based Detection

Paxon: In this article, the author describes an open-source NIDS called Bro. This system is designed to be a real time signature based approach to detecting port scans. It relies on the LIBCAP library for capturing network data and is capable of both TCP and UDP processing. It functions by analyzing packet header information to generate what it calls events. These events are *connection attempt, connection established, connection rejected,* and *connection finished.* The preprocessor then passes these events to the analysis engine which alerts the administrator if these events ever cross a certain threshold. It also attempts to detect vertical scans by looking for the one source trying to contact X number of ports in Y amount of time. The main drawbacks to this system are that it drops packets in times of high network traffic load, and produces a high level of false positives.

Roesch:

Mahomey and Chan:

2.1.2 Anomaly-Based Detection

Kim *et al.*:

Ertoz *et al.*:

Streilein *et al.*:

2.2 Privilege Escalation

Privilege Escalation attacks are described in two general categories in this paper: Remote to User Attacks and User to Root Attacks. In the broadest sense, Remote to User Attacks are when an attacker seeks to gain local user access to a machine that they have network access to. A User to Root Attack, as one may infer from its name, is when an attacker who already possesses local user access seeks to escalate their privileges to those of a root user, IE gain root user access. In the following sections, both types of attacks are explained in detail.

2.2.1 Remote to User

A Remote to User attack is when an attacker who has network access to a system but does not have an account on that machine exploits a vulnerability on the system to gain unauthorized local access as a user of the target system. These types of attacks come in many forms, and can be as simple as getting valid user authentication information through guessing a user's password with a dictionary attack. However, just as common are attacks that exploit a vulnerability in a common and innocuous system service, such as FTP to gain local user access. These exploits can change system settings to allow an attacker remote access, but they can also trigger buffer overflows, which in many circumstances allow an attacker to execute arbitrary code on the remote host (which is often used to gain root access, as we will discuss later).

2.2.2 User to Root

A User to Root attack is when an attacker who has no access or local user access to a system escalates their privileges to those of a root user. As with Remote to User attacks, User to Root attacks can come in a wide variety of formats. However, most of them tend to be some form of vulnerability exploit that either allows temporary root access (which is enough for any attacker to establish a backdoor for themselves for root access in the future), or triggers a buffer overflow, which can be used by a clever attacker to execute arbitrary code to gain root access.

2.2.3 *Detection*

Currently, research on detection of Privilege Escalation attacks is fairly rudimentary. While a lot of research has been done examining

2.3 Denial of Service

3. METHODS

4. EVALUATION

5. DISCUSSION

5.1 Training Sets

5.2 Effective Defense

5.3 Open Problems

6. CONCLUSIONS

7. REFERENCES

[1] 802.11 Denial-of-Service Attacks: Vulnerabilities and Practical solutions.

[2] M. Alenezi and M. Reed. Methodologies for detecting DoS/DDoS attacks against network servers. *ICSNC 2012: The Seventh International Conference on Systems and Networks Communications*, (c):92–98, 2012.

[3] S. Alexander. Trust engineering: rejecting the tyranny of the weakest link. *Proceedings of the 28th Annual Computer Security ...*, pages 145–148, 2012.

[4] M. Bernaschi, E. Gabrielli, and L. Mancini. Operating system enhancements to prevent the misuse of system calls. *... of the 7th ACM conference on ...*, 2000.

[5] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita. Surveying Port Scans and Their Detection Methodologies. *The Computer Journal*, 54(10):1565–1581, Apr. 2011.

[6] S. Brugger and J. Chow. An assessment of the DARPA IDS Evaluation Dataset using Snort. *UCDAVIS department of Computer Science*, pages 1–19, 2007.

[7] N. Carlini, A. Felt, and D. Wagner. An evaluation of the google chrome extension security architecture. *... the 21st USENIX Conference on Security*, 2012.

[8] M. Dalton, C. Kozyrakis, and N. Zeldovich. Nemesis: preventing authentication & access control vulnerabilities in web applications. *... of the 18th conference on USENIX ...*, 2009.

[9] Y. Djemaiel and N. Boudriga. A global marking scheme for tracing cyber attacks. *Proceedings of the 2007 ACM symposium on Applied computing - SAC '07*, page 170, 2007.

[10] W. Eddy. TCP SYN flooding attacks and common mitigations. 2007.

[11] M. Gandhi and S. Srivatsa. Detecting and preventing attacks using network intrusion detection systems. *International Journal of Computer Science and ...*, (2):49–60, 2008.

[12] L. A. Gordon, M. P. Loeb, W. Lucyshyn, and R. Richardson. COMPUTER CRIME 2004 CSI / FBI Computer Crime and Security Survey. 2004.

[13] S. Hansman and R. Hunt. A taxonomy of network and computer attacks. *Computers & Security*, 24(1):31–43, Feb. 2005.

[14] F.-h. Hsu and T.-c. Chiueh. Scalable Network-based Buffer Overflow Attack. pages 163–171, 2006.

[15] A. Hussain, J. Heidemann, and C. Papadopoulos. A framework for classifying denial of service attacks. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications - SIGCOMM '03*, page 99, New York, New York, USA, 2003. ACM Press.

[16] S. Jana, Suman; Kasera. On Fast and Accurate Detection of Unauthorized Wireless Access Points Using Clock Skews âLŮ Categories and Subject Descriptors. pages 104–115.

[17] M. Kirkpatrick and S. Kerr. Enforcing physically restricted access control for remote data. *Proceedings of the first ACM conference on Data ...*, pages 203–212, 2011.

[18] B. Kuperman and C. Brodley. Detection and prevention of stack buffer overflow attacks. *Communications of the ...*, 48(11), 2005.

[19] K. Labib and V. Vemuri. Detecting and visualizing denial-of-service and network probe attacks using principal component analysis. *Proc. SAR*, 2004.

[20] M. Labs. MIT Lincoln Laboratory Communication Systems and Cyber Security Cyber Systems and Technology DARPA Intrusion Detection Evaluation, 1999.

[21] U. Larson and D. Nilsson. Securing vehicles against cyber attacks. *... of the 4th annual workshop on Cyber security ...*, page 1, 2008.

[22] V. Livshits and M. Lam. Finding security vulnerabilities in Java applications with static analysis. *... of the 14th conference on USENIX Security ...*, 2005.

[23] M. McDowell and This. Security Tip (ST04 Â■ 015) Understanding Denial Â■ of Â■ Service Attacks. page 2013, 2013.

[24] D. Moore, C. Shannon, D. J. Brown, G. M. Voelker, and S. Savage. Inferring Internet denial-of-service activity. *ACM Transactions on Computer Systems*, 24(2):115–139, May 2006.

[25] R. Newman. Cybercrime, identity theft, and fraud: practicing safe internet-network security threats and vulnerabilities. *... of the 3rd annual conference on Information security ...*, 2006.

[26] D. K. Nilsson and U. E. Larson. Conducting Forensic Investigations of Cyber Attacks on Automobile In-Vehicle Networks. *International Journal of Digital Crime and Forensics*, 1(2):28–41, 2009.

[27] R. Richardson and C. Director. CSI computer crime and security survey. *Computer Security Institute*, 2008.

[28] R. Sailer, T. Jaeger, X. Zhang, and L. van Doorn. Attestation-based policy enforcement for remote access. *Proceedings of the 11th ACM conference on Computer and communications security - CCS '04*, page 308, 2004.

[29] E. Silenok, Elena; Roedel, Chris; Silenok. Detection and Characterization of Port Scan Attacks Cynthia Bailey Lee Chris Roedel Classification Methodology.

[30] S. Specht and R. Lee. Distributed denial of service: Taxonomies of attacks, tools, and countermeasures. *... on Parallel and Distributed ...*, (September):543–550, 2004.

[31] S. Staniford, J. Hoagland, and J. McAlerney. Practical automated detection of stealthy portscans. *Journal of Computer Security*, 10:105–136, 2002.

[32] A. Stock, J. Williams, and D. Wichers. OWASP top 10. *OWASP Foundation, July*, 2013.

[33] F. Sun, L. Xu, and Z. Su. Static detection of access control vulnerabilities in web applications. *Proceedings of the 20th USENIX conference on ...*, 2011.

[34] P. Taylor and P. Mell. UNDERSTANDING INTRUSION DETECTION SYSTEMS NEWSLETTER. Technical Report April 2013, The EDP Audit, Control, ans Security, 2006.

[35] G. Vigna and R. Kemmerer. NetSTAT: a network-based intrusion detection approach. *Proceedings 14th Annual Computer Security Applications Conference (Cat. No.98EX217)*, pages 25–34, 1998.

[36] D. Wagner and R. Dean. Intrusion detection via static analysis. *Security and Privacy, 2001. S&P 2001. ...*, 2001.