Intrusion Detection: Challenges and Current Solutions

Ibraheem Alhashim
Department of Computer Science
Portland State University
Portland, Oregon
ibraheem@pdx.edu

Abstract: The problem of intrusion has been a long standing computer security challenge. With governments and institutes relying more on computer networks it is becoming essential to arm networks with defenses against intrusion and misuse. Such intrusions include installing harmful viruses, spreading worms, initiating distributed attacks, and misuse by authorized users. To defend against these attacks, systems can utilize anomaly detection and pattern matching against known attacks to detect intrusions. These intrusion detection systems have rapidly evolved since their introduction in the 1980s. This paper discusses some of the challenges and threats that face intrusion detection systems followed by a survey of several systems currently available.

Introduction

In all the major public and private sectors, people rely on computers and networking to accomplish the simplest and most complex tasks of their lives. Computers record and handle important information from online game accounts to bank accounts, from tracking personal packages to even tracking military fighters. Such important resources are faced with constant attacks from different types of individuals with the intention of damaging or stealing such valuable informations. These attacks raised serious concerns in the domain of security and data protection.

Experts in the field of computer security have worked on developing security models that protect systems from attacks by shielding. Shielding requires users to identify and authenticate themselves. However, these models are limited by a number of constrains that do not allow for a flexible system. Such limitations include the improbability of building a perfectly secure system, the slow evolution of shielding systems, and constrains on the user's environment and abilities. Perhaps a more significant limitation is the fact that a shielded system is still vulnerable to insider attacks [1]. During the mid 1980s, security experts proposed an alternative approach to security that involved auditing user activities and reporting anomalous behavior [2]. Research on this approach resulted in what is now called an intrusion detection system (IDS).

Intrusion detection deals with the problem of identifying unauthorized access or abusive usage by authorized users. Examples of such intrusions include unauthorized alteration of files and unauthorized use of the system resources such as user account creation or deletion [1]. For nearly thirty years of research and development, the field of intrusion detection is still maturing [2]. The threats and attacks on computer networks today increase at a rate that makes it hard on IDS developers to create sold intrusion detection solutions. Despite these challenges, today's systems are composed of a highly sophisticated set of hardware and software solutions that help secure valuable information.

Attacks and threats

Malwares are "softwares that are designed to infiltrate or damage a computer system without the owner's informed consent" [3]. A software is considered to be a malware based on the intentions of its developer. Generally, the effects of malware range from simple nuisance adwares to hard drive wiping viruses. Other malwares are used to regain access to unauthorized data by executing their payloads, such malwares are called Trojan horses.

One of the dominant types of malware are **computer viruses**. Computer viruses act in a similar manner as biological viruses. They infiltrate the system, destroy data, replicate and spread. The life cycle of viruses start with the execution of an infected code. The code then starts to spread to other applications on the system by copying it self and hiding in other executable files. Each time an infected application is executed the virus code executes as well. Subramanya categorized viruses into seven types in his paper on computer viruses [4]. Such categories include file infecters, boot record infecters, encrypted viruses, and polymorphic viruses.

An example of such viruses is the Nimda virus that was active in the year 2002 . This virus has a mailing worm code that work on spreading the virus through an e-mail attachment named "Readme.exe". The virus work on infecting executable files in the windows environment. The virus also search for vulnerable web servers and work on infecting them [4]. Such viruses require system calls that fit the profile of a malicious anomaly. Therefore, such attack patterns should be included in the detection rules of an intrusion detection systems.

With the advancements in the field of anti-virus development, a new breed of malware, called **computer worms**, gained popularity with the help of large scale networks. A computer worm is "a program that self-propagates across a network exploiting security or policy flaws in widely used services" [5]. The difference between computer viruses and computer worms is that viruses need to be executed by the user for propagation where as worms doesn't dependent on existing executables for survival. Worms are also different from viruses in the targets they attack. Often, a worm's main goal is to spread throughout a network without necessarily destroying data or infecting other files on the same system.

Weaver et al., in their paper "A taxonomy of computer worms", categorized worms based on five criteria [5]. First is target discovery which represent the technique in which the worm finds new systems to infiltrate. The second criteria is the carrier which represent how a worm travels to its new target. Perhaps the most important of all is the creator's intentions. The way that a worm gets activated is the fourth criteria. Another important criteria is the worm's payload.

The payload of a computer worm is a separate code from the propagation code. Different payloads have been implemented for different attack intentions. The most common payload in worms is a nonfunctional payload in which the worm's sole purpose is to spread. An Internet remote control payload allows attackers the ability to execute commands on victim machines without their consent and it does so by exploiting back doors on that system. An example of such payload is the Code Red II worm that was active in 2001 [5]. A spam-relay payload helps spammers avoid IP filtering mechanisms especially since worms spread to different machines all over the world. HTML proxies payloads aid attackers in phishing for sensitive users data which can be done by redirecting web requests to some random proxy machines. Denial of Service (DOS) attacks can also be included with worm's payloads.

Other payloads include data collection, data damage, physical-world control or damage, and worm maintenance that works on updating the worm's code from a covert server. Similar to computer viruses, these attack follow a pattern of system calls, disk operations, and network activities that can be modeled and monitored by IDS applications.

Other harmful intrusions on computer networks are **Distributed denial-of-service** (DDoS) attacks. Denial-of-service attacks are “explicit attempts to prevent the legitimate use of a service” [6]. The distributed version of the attack employs multiple sources to achieve the DoS goal. The result of such attacks often cause the victim to receive destructive traffic that would inflict damage on the system. The attacks often starts when the attacker sends a stream of data packets to the victim that would then consume an important resource. When that resource is consumed, the system can no longer serve legitimate clients requesting its services. Another approach for DDoS attacks is made by sending special kinds of packets that would cause an application or a protocol to freeze or shutdown.

DDoS attacks are usually carried out in several stages. Attackers start by searching the network, often the Internet, for vulnerable systems and work on recruiting them for an attack. These systems, commonly called **bots**, get broken into and then infected by the attacker's code. This code can also help in searching and recruiting more systems. Another way for recruiting that attackers use, is by disguising their code in what seems to be useful software. This software can be distributed by e-mail attachments as a fake update or what would seems to be a fun or entertaining application. These types of softwares could contain other malicious code, similar to what viruses or worms would normally do, and are called Trojans. Once a system gets infected it would start sending the harmful packets, as a part of a large botnet, throughout the internet.

The reason why DDoS attacks are possible and effective is that the current Internet infrastructure allows for an effective mechanism to transfer packets from source to destination. This leaves the responsibility of security and protection on the destination system rather than inspecting the current packet that is being forwarded by the intermediate network. Thus, if a system is compromised it would efficiently work on affecting other connected systems in its network. Another reason for why DDoS are successful is the fact that most Internet hosts and services are at risk of being overwhelmed with too many user requests. It is very difficult and highly impractical to limit the number of requests a server should expect at any point. For intrusion detection systems to be able to detect these attacks it would require a highly sophisticated approach to isolate the malicious packets from the legitimate traffic.

Another serious intrusion threat is the **misuse of authorized users**. These attacks are challenging for intrusion detectors since authorized users, especially maintenance and administrative personnels, retain rights to modify and delete important data on a system. An IDS is forced to follow the security policies and anomaly models for what to report and what to allow. Thus, in the event of a misuse by a network administrator, or a hijacked account, the IDS would report a modification misuse with a lower priority since that user has been granted the leverage to perform such modifications [9]. To overcome this problem IDS designers need to incorporate the primary principles of multi-level security systems and separation of duties in their intrusion detection solutions.

Threats are still possible in some cases in which attackers can escape detection [9]. This can be achieved when the attacking code follows a slower routine of modifications than a known malicious behavior. Also, the attacker's code might take advantage of low level features of the system that the IDS do not usually log which is executed by infiltrated accounts with a low security level.

Intrusion Detection Systems (IDS)

An intrusion detection system is a “set of security tools deployed throughout a network that work on detecting intrusions” [7]. The two common types of IDS are the Network Intrusion Detection Systems and the Host Intrusion Detection Systems.

Network Intrusion Detection Systems (NIDS) are commonly installed as a dedicated part of the network. Multiple NIDS are often used to detect and report malicious behaviors or files that conform with a known model of attacks. Such incident reports would then be logged and sent to a security administrator. There are several advantages when using network based detection systems. A single NIDS subnet could cover a large range of systems. NIDS allow network administrators to easily install and update the IDS software that would in turn affect a large segment of the network. Network based setups would also be immune to DoS attacks that commonly target host machines. Another significant advantage is that NIDS are usually platform independent and hence securing multiple systems and running on different operating environments all on the same network. There are, however, some disadvantages to NIDS [7].

These network based intrusion detection systems are known to produce false positives due to the complexity of distinguishing between a legitimate traffic and a malicious one. Another obstacle for NIDS is their inability to analyze encrypted traffic or handle certain high speed networks [8]. IDS relay on known threat models to detect intrusion which limit its ability to detect new variations of similar threats. Similar to any network based service a NIDS is subject to a latency effect that would render the system useless if an attack inflect a damage before the IDS sends an alert to the security administrator.

The second type of IDS is the **Host Intrusion Detection System** (HIDS). HIDS are designed to monitor, detect, and respond to user and system activity and attacks on a host [8]. Unlike the network based IDS that deal with network packets, this setup allows for a better deal of detection on a single system. HIDS work by supervising modifications to important variables on the host system and compare them to known attack models. An advantage of using host based IDS is the system's ability to associate a user to an event [7]. In some cases, these systems are able to detect threats that aren't detectable by network based IDS. Unlike NIDS, a host based system can detect encrypted traffic since it resides on the destination system that would decrypt the encrypted packet. Host-based intrusion detection is also best suited to combat internal threats and misuse by authorized users [8]. This is due to the fact that a HIDS can monitor and react to specific user actions on the same host.

Limitations of HIDS exist and in some cases are closely related to essentials of the intrusion detection process. Such limitation is that intrusion reports are rendered as unrealizable once an attack is successful on a host. Another limitation is that these systems would normally run under the host's operating system, and therefore vulnerable to the same attacks on the OS. Also, HIDS are required to be installed on each system in the network which is not practical during the process of upgrading the IDS software. Host based IDS require resources to run on the host and thus threatened by DoS attacks [7].

To overcome the shortages of both system, a third type of IDS can be setup on a network. Such system is called the **Hybrid Intrusion Detection System**. Hybrid systems “offer management of and alert notification from both network and host-based intrusion detection devices” [8].

Current IDS solutions

Network security and intrusion detection have been an important issues since the early use of computer networks in the military. Research in the field of intrusion detection extended to include players from the Department of Defense, the UC Davis Computer Security Lab and other academic institutes, and eventually commercial developers [8]. Current IDS products can be categorized into research, government sponsored, commercial, and open source products. Examples of these products are discussed below.

Research Systems

EMERALD

The Event Monitoring Enabling Responses to Anomalous Live Disturbances (EMERALD) is an IDS research tool that was developed and maintained by the Stanford Research Institute, also known as SRI International [10]. The project began in 1983 following the introduction of a multivariate statistical algorithm that help separate different user behaviors. Later on the project included the use of a signature analysis subsystem and eventually resulted into a real time system that monitors activities on multiple hosts. EMERALD was the end product of these projects with the objective of creating a large network intrusion detector. The system structures users into domains that are then analyzed by three monitors. Each monitor work on anomaly detection and signature analysis. Another part of the monitors, the resolver, works on integrating these results.

For its statistical profiling, EMERALD uses an abstract notion of a profile that separates management from analysis. For its signature analysis a layer called the service-layer works on detecting abnormalities that correspond with known threats. These threat reports get asserted by a higher level monitor that work on identifying the extent of the threat. The results of these reports can then be passed by the resolver to third party tools that provide an interface for network administrators and could execute counter measurement procedures [10].

Research on EMERALD is still active especially with the increased complexity of attacks posed on today's networks. Most of these attack require IDS to collect and analysis data from multiple networks, all in real time. EMERALD offer a flexible and well established intrusion detection architecture that represents the future of IDS.

Bro

Bro is an open source Unix based NIDS that is developed at the Lawrence Berkeley National Laboratory, and lead by Vern Paxson. It works by passively watching network traffic for suspicious activities [11]. Bro's detection process starts by extracting application level semantics from network traffic. The system then works on analyzing and comparing activities with known threat patterns. Bro detects both specific attacks defined by signatures or special events and detects unusual activities.

Each event captured by Bro is checked through policy scripts that Bro administrators supply or by the default scripts that ship with Bro. The scripts are written in a specialized language that is designed for network analysis and security analysis. Each script contains several event handlers specifying particular actions for each event. These event handlers maintain and update global state information, generate other new events, generate alerts that produce log messages, and can also invoke shell commands [11].

There are several design goals for the Bro IDS [10]. A major goal of Bro is its ability to handle high speed and high volume traffic with no packet dropping. Bro also features a real time notification system that ensure timely response to threats. Another design goal is the separation between mechanism and policy that results in easier implementations and straightforward maintenance. Bro's scripts feature allows for extensibility which is crucial for any IDS since attacks today are evolving very rapidly. Since Bro's source code is open source, it is actively evolving by the contributions from computer security specialists which has an advantage of being up to date on current threats.

Government sponsored Systems

CIDDS

The Common Intrusion Detection Director System (CIDDS) is a dedicated collection of software and hardware that supports the Air Force Information Warfare Center’s (AFIWC) Intrusion Detection Tools (IDT) program [10]. AFIWC has an Air Force Computer Emergency Response Team (AFCERT) that is responsible for administration and network security operations involving the IDT program.

CIDDS obtain connections data from Automated Security Incident Measurement (ASIM) Sensor host machines and stores this data on a local database. ASIM is the first solution “to incorporate both a hardware and software solution to network intrusion detection” [8]. The data stored at the local Oracle databases is examined for detailed correlation and analysis by security analysts and automated tools. The system has a graphical user interface that allow users “to correlate, study, and analyze indicators of potentially intrusive, malicious or unauthorized events occurring on Air Force networks” [10]. The CIDDS and ASIM Sensors provide the capability to enforce Air Force network security policies and help protect Air Force networks from threats.

Commercial Systems

SecureNet IDS/IPS

The SecureNet IDS is developed by the software company Intrusion. Intrusion claims that its software provide packet analysis and application awareness, and it can be deployed passively for ID or actively for intrusion prevention [12]. SecurNet can be work on networks with speed ranges upto 1 Gbit/s. The software allow for tweaking and creating pattern-matching and protocol-decode signatures.

The system uses a hybrid detection model that applies pattern matching for performance and protocol decoding for intentional evasion and network anomalies. SecureNet has a scripting language and graphical interface for adjustments and for creating specific protocol decode detection signatures.

Dragon IDS/IPS

Enterasys provide a range of network security solutions. Dragon is their IDS and Intrusion Prevention System (IPS). The system can be deployed as both host based or network based detection system. Its ID and IP systems use signature-based, protocol-based, anomaly-based, and behavior-based models simultaneously to detect intrusion [13]. Dragon features a multi-threaded architecture and a virtual sensor that allow for scalability.

The IP and ID systems are the core components which can be combined with other products from Enterasys such as the Dragon Security Command Console (DSCC) and NetSight Automated Security Manager (ASM). These tools work on identifying and locating intrusions and can automatically execute appropriate commands in case of serious threats. The Enterasys Network Access Control (NAC) tool can be used for monitoring behaviors of authorized users. The IDS integrates vulnerability pattern matching, protocol analysis and anomaly based detection and also applies application-based event detection that detects unknown signature attacks against common applications such as FTP. The IPS component work on attack alerts, dropping malicious packets and terminating sessions. It also establish firewall rules that block the source of the attack for a period of time [13].

Open Source Systems

OSSEC

OSSEC is an open source host based intrusion detection system. The system was first authored and is lead by Daniel B. Cid. OSSEC features a variety of IDS tasks such as log analysis, registry monitoring, integrity checks, rootkit detection, time-based alerting and active response [14]. OSSEC runs on various operating systems including Windows, Mac OS X 10, Linux, and other POSIX compliant systems.

System integrity checking is a key feature of OSSEC's architecture. OSSEC can run as a standalone process or as a client-sever model. The client side compress and encrypts the data then uses UDP to communicate with the server. Authentication with the server is done by using a pre-shared key which eliminate the need for an integrity database on the server. The client process sends copies of the data to the server where the integrity delta is calculated. Newer versions of OSSEC support reading database logs and storing alerts on PostgreSQL and MySQL

OSSEC log analyzer contains a large number of predefined log rules. While running, the analyzer monitors a list of system logs that can be configured to include specifically important logs. These logs get processed by the an engine that attempts to match rules stored inside XML documents. These XML documents contain definitions that describe the level of alert, regular expression matching, process lookup, and other declarations.

Snort

Snort is a cross-platform, lightweight network intrusion detection tool that can be deployed to monitor small TCP/IP networks and detect a wide variety of suspicious network traffic as well as outright attacks [15]. It was first developed by Martin Roesch and now owned and developed by Sourcefire. Snort's design goal is to have a small system footprint, and be easy to configure by system administrators who need to quickly develop security solutions for a new threat.

Snort is based on the “libpcap” packet sniffer and logger. It performs content pattern matching to detect attacks using rules based logging. Such attacks include buffer overflows, port scans, CGI script attacks, and Server Message Block (SMB) probes. Snort features a real-time alerting feature that send alerts to syslog, SMB WinPopup messages, or separate alert files. The system is configured by command line switches and Berkeley Packet Filter commands. The system's detection engine interprets a simple programming language that is used to describe per packet tests and actions.

This IDS has three primary subsystems. The first subsystem is the the packet decoder. Each subroutine in the decoder order packets data by layering data structures on the network's traffic. These routines are called in order from the data link layer up through the application layer. The second subsystem is the the detection engine. Detection rules are stored in a two dimensional linked list of chain headers and chain options. Chain headers include common attributes and the chain options contain the detection modifier options. This technique helps in speeding up the detection process since the common attributes are compressed into one Chain header and the individual detection signatures are stored in the Chain option. Rule chains are searched recursively and when a rule matches the current packet in the detection engine it triggers the action assigned in the rule definition.

The third subsystem of Snort is the logging and alerting subsystem. The logging system logs packets in a decoded format or in a tcpdump binary format which is saved to a single log file. Other logging options include shutting down the logging system that would result in performance improvements. Alerts options in Snort include sending alerts to syslog, logging to a text file, or sending WinPopup messages via the SMBclient program. The alerting system has two options when sending alerts to plain text files. A fast alert option writes a short list of information to the alert file which in turn allows for greater performance. The full alerting option writes alert messages and packet headers information through the transport layer protocol. As in logging, alerts can be turned off for performance reasons.

Snort's uses a command line interface as its basic interface, however it can be combined with other software like OSSIM for a graphical user interface and for generating graphs and charts. Other proprietary versions which might include integrated hardware and software support are sold by the current owner of the Snort IDS.

Challenges facing IDS

The field of intrusion detection is still maturing with many challenges facing ID developers [10]. Such challenges include the increased number of intruders and intrusion techniques that uses sophisticated and complex methods. Another issue is the use of encryption when the malicious code spread through the connected networks. The use of different operating systems and technologies makes the job of an IDS harder since it needs to correlate data coming from different environments. An important requirement for a sold IDS is the ability to handle alerts in real-time. This is becoming much more difficult with the increase of network traffic and speeds.

Other challenges include the lack of standards in the field of intrusion detection with multiple structures for each system. This strategy slows down the progress of intrusion detection since each system would have to create its own set of rules and definitions, different from other systems that already established similar attack signatures. IDS are also prone to high levels of false positives and false negatives that are unacceptable. IDS face threats and attacks to the system itself that are similar to attacks on anti-virus softwares and firewalls. IDS also face challenges of limited traffic visibility due to switched LANs [10].

Conclusions

ID systems try to solve the problem of identifying unauthorized access or abusive usage by authorized users. Unauthorized access is used to gain access to important information, to spread worms, to recruit bots, to initiate DDoS attacks, or to be used for other malicious reasons. Different types of IDS include network based systems and host based systems. Other hybrid models exist and work by collecting data from HIDS and NIDS softwares. Most of these systems use techniques that identify threats using anomaly detection, signature analysis, or both. There are different options available for network administrators that include open source software and proprietary software. The new approach in the field of intrusion detection relies on the usage of abstract rules and real-time responses combined with the use of special networking hardware to optimize the intrusion detection process. These ID systems are rapidly evolving due to the increased challenges that face the network security domain.

References

[1] Network intrusion detection, Mukherjee, B.; Heberlein, L.T.; Levitt, K.N. Network, IEEE, Volume: 8 Issue: 3 May/Jun 1994. Pages: 26-41.

[2] Defending yourself: The role of intrusion detection systems. McHugh, J., Christie, A., Allen, J., 2000. IEEE Software (Sept./Oct.), Pages: 42–51.

[3] Malware. Wikipedia, The Free Encyclopedia. Wikimedia Foundation, Inc. 12 Nov 2007 http://en.wikipedia.org/w/index.php?title=Malware&oldid=170410498

[4] Computer viruses. Subramanya, S.R.; Lakshminarasimhan, N.Potentials, IEEE Volume 20, Issue 4, Oct-Nov 2001 Pages: 16 - 19.

[5] A taxonomy of computer worms. Weaver, N.; Paxson, V; Staniford, S; and Cunningham, R. Proceedings of the 2003 ACM workshop on Rapid Malcode.

[6] A taxonomy of DDoS attack and DDoS defense mechanisms. Jelena Mirkovic , Peter Reiher,, ACM SIGCOMM Computer Communication Review, v.34 n.2, April 2004.

[7] IDS - Today and Tomorrow. Goeldenitz, T. The SANS Institute. January 22, 2002. From http://www.sans.org/reading_room/whitepapers/detection/351.php?id=351&cat=detection.

[8] The Evolution of Intrusion Detection Systems. Paul Innella, Tetrad Digital Integrity, LLC. November 16, 2001. From http://www.securityfocus.com/infocus/1514

[9] An intrusion-detection model. D. E. Denning. IEEE Transactions on Software Engineering, Vol. SE-13(No. 2):222-232, Feb. 1987.

[10] State of the practice of intrusion detection technologies. Allen, J., Christie, A., Fithen, W., McHugh, J., Pickel, J., and Stoner, E. 2000. Technical Report CMU/SEI-99TR -028, CMU/SEI.

[11] Bro Intrusion Detection System. Lawrence Berkeley National Laboratory. Accessed November 17, 2007 from http://www.bro-ids.org

[12] SecureNet Product Brief. Intrusion, Inc. Accessed November 18th, 2007 from http://www.intrusion.com/Default.aspx?DN=bee1192e-5a5b-4a44-b653-efce9f846523

[13] Enterasys Dragon Intrusion Detection and Prevention. November 19th, 2007 from http://www.enterasys.com/products/advanced-security-apps/dragon-intrusion-detection-protection.aspx

[14] Free Lunch: OSSEC. Storms, A. 360 Security, nCircle.com. November 20th, 2007 from

http://blog.ncircle.com/blogs/sync/archives/2007/04/free_lunch_ossec.html

[15] Snort: Lightweight intrusion detection for networks. M. Roesch.. In Proc. 13th Systems Administration Conference (LISA), pages 229-238. USENIX Association, November 1999.