CHAPTER

18 Intrusion Detection and Prevention Systems

Intrusion detection systems (IDSs) and intrusion prevention systems (IPSs) are important tools in a computer security arsenal. Often thought of as a tertiary extra after antivirus software and firewalls, an IDS is often the best way to detect a security breach. As useful as they can be, however, successfully deploying an IDS or IPS is one of the biggest challenges a security administrator can face.

This chapter will introduce IDS/IPS concepts, describe the different IDS and IPS types available, identify features to help you evaluate different solutions, and discuss real-life deployment considerations. By the end of this chapter, you should have a rich understanding of both systems and be prepared to navigate the toughest operational issues.

An IDS can be network based or host based: a network IDS is referred to as a NIDS, whereas a host-based IDS is referred to as a HIDS. Additionally, a NIDS and HIDS can *detect* traffic of interest, or if they are further configured to prevent a specific action from happening, they are referred to as intrusion prevention systems: NIPS and HIPS. Keep in mind, no matter what the form, NIDS, HIDS, NIPS, or HIPS, they are generically referred to as IDS.

IDS Concepts

Intrusion detection (ID) is the process of monitoring for and identifying specific malicious traffic. Most network administrators do ID all the time without realizing it. Security administrators are constantly checking system and security log files for something suspicious. An antivirus scanner is an ID system when it checks files and disks for known malware. Administrators use other security audit tools to look for inappropriate rights, elevated privileges, altered permissions, incorrect group memberships, unauthorized registry changes, malicious file manipulation, inactive user accounts, and unauthorized applications. An IDS is just another tool that can monitor host system changes (host based) or sniff network packets off the wire (network based) looking for signs of malicious intent.

An IDS can take the form of a software program installed on an operating system, but today's commercial network-sniffing IDS/IPS typically takes the form of a hardware appliance because of performance requirements. An IDS uses either a packet-level network interface driver to intercept packet traffic or it "hooks" the operating system to insert inspection subroutines. An IDS is a sort of virtual food-taster, deployed primarily for early detection, but increasingly used to prevent attacks.

When the IDS notices a possible malicious threat, called an *event,* it logs the transaction and takes appropriate action. The action may simply be to continue to log, send an alert, redirect the attack, or prevent the maliciousness. If the threat is high risk, the IDS will alert the appropriate people. Alerts can be sent by e-mail, Simple Network Management Protocol (SNMP), pager, SMTP to a mobile device, or console broadcast. An IDS supports the defense-in-depth security principle and can be used to detect a wide range of rogue events, including but not limited to the following:

- Impersonation attempts
- Password cracking
- Protocol attacks
- Buffer overflows
- Installation of rootkits
- Rogue commands
- Software vulnerability exploits
- Malicious code, like viruses, worms, and Trojans
- Illegal data manipulation
- Unauthorized file access
- Denial of service (DoS) attacks

Threat Types

To really understand an IDS, you must understand the security threats and exploits it can detect and prevent. Threats can be classified as attacks or misuse, and they can exploit network protocols or work as malicious content at the application layer.

Attacks or Misuse

Attacks are unauthorized activity with malicious intent using specially crafted code or techniques. Attacks include denial of service, virus or worm infections, buffer overflows, malformed requests, file corruption, malformed network packets, or unauthorized program execution. *Misuse* refers to unauthorized events without specially crafted code. In this case, the offending person used normally crafted traffic or requests and their implicit level of authorization to do something malicious. Misuse can also refer to unintended consequences, such as when a hapless new user overwrites a critical document with a blank page. Another misuse event could be a user mapping a drive to a file server share not intended by the network administrator.

Regardless of how an alert is detected, the administrator groups all alerts into one of four categories:

- True positives (correct escalation of important events)
- False positives (incorrect escalation of unimportant events)

- True negatives (correct ignorance of unimportant events)
- False negatives (incorrect ignorance of important events)

An easy way to remember these principles is by thinking about the concepts of "alert" and "condition." A simple fire alarm can serve as a good illustration:

- A true positive happens when the alert is positive (the alarm sounded) and the condition it represents is true (meaning there actually is a fire). That's a good thing—it's what the fire alarm is supposed to do.
- A false positive happens when the alert is positive (the alarm sounded) but the condition it represents is false (meaning there is no fire). That's not so great—it wastes time and annoys people.
- A true negative is when the alert is negative (the alarm is not sounding) and it is reporting a true condition (there is no fire). That's a good situation, and it's what you'd expect the majority of the time.
- A false negative is when the alert is negative (no alarm sounded), but the condition it represents is false (there is fire). This is a truly dangerous condition, whether in the case of a building fire or in an IDS.

You can also think about these examples in the context of other detection systems, such as car alarms. Some people get so annoyed by false positives (car alarms going off for no apparent reason) that they learn to ignore all car alarms, even when they are true positives (there was a reason for the alarm, but nobody paid attention). This phenomenon can also happen with an IDS. If not properly tuned, it can generate so much "noise" that it's ignored.

Most IDSs are deployed to detect intentionally malicious attacks coming from external locations, but they are also proving of value within the corporate world for monitoring behaviors and violations by internal users. Security surveys often reveal internal misuse events as a leading cause of corporate data loss, and an IDS tool can track internal maliciousness (intentional or unintentional) almost as well as external attacks. In one case, a sharp security officer working in an IT department used an IDS to catch a fellow employee cracking passwords and reading confidential e-mail.

Network Protocol Attacks

Many of the security threats detected by an IDS exploit network protocols (layers two and three of the OSI model). Network protocols such as TCP/IP define standard ways of transmitting data to facilitate open communications. The data is sent in a packet (layer three), which is then encapsulated into a layer two frame, which is then transmitted as packages of electronic bits (1s and 0s) framed in a particular format defined by a network protocol—but the protocols do not contemplate the consequences of malicious packet creation. This is because protocols are designed to perform functions, not to be secure.

When information is sent between network hosts, commands and data sent by higher-layer application processes (such as FTP clients, web servers, and IM chat programs) are placed as payload content into discrete containers (called *datagrams* or *packets*), numbered, and sent from source to destination. When the packets arrive at the destination, they are reassembled, and the content is handed off to the destination application. Network protocols define the packet's formatting and how the datagram is transmitted between source and destination. Malicious network protocol attacks interfere with the normal operation of this process.

Flag Exploits Abnormally crafted network packets are typically used for DoS attacks on host machines, to skirt past network perimeter defenses (bypassing access control devices), to impersonate another user's session (attack on integrity), or to crash a host's IP stack (DoS). Malicious network traffic works by playing tricks with the legitimate format settings of the IP protocol. For instance, using a specially crafted tool, an attacker can set incompatible sequences of TCP flags, causing destination host machines to issue responses other than the normal responses, resulting in session hijacking or more typically a DoS condition. Other examples of maliciously formed TCP traffic include an attacker setting an ACK flag in an originating session packet without sending an initial SYN packet to initiate traffic, or sending a SYN and FIN (start and stop) combination at the same time. TCP flags can be set in multiple ways and each generates a response that can either identify the target system, determine if a stateful packet-inspecting device is in front of the target, or create a no-response condition. Port scanners often use different types of scans to determine whether the destination port is open or closed, even if firewall-like blocking mechanisms are installed to stop normal port scanners.

Fragmentation and Reassembly Attacks Although not quite the security threat they once were, IP packets can be used in *fragmentation* attacks. TCP/IP fragmentation is allowed because all routers have a *maximum transmission unit (MTU)*, which is the maximum number of bytes that they can send in a single packet. A large packet can be broken down into multiple smaller packets (known as *fragments*) and sent from source to destination. A *fragment offset* value located in each fragment tells the destination IP host how to reassemble the separate packets back into the larger packet.

Attacks can use fragment offset values to cause the packets to maliciously reassemble and intentionally force the reassembly of a malicious packet. If an IDS or firewall allows fragmentation and does not reassemble the packets before inspection, an exploit may slip by. For example, suppose a firewall does not allow FTP traffic, and an attacker sends fragmented packets posing as some other allowable traffic. If the packets act as SMTP e-mail packets headed to destination port 25, they could be passed through, but after they are past the firewall, they could reassemble to overwrite the original port number and become FTP packets to destination port 21. The main advantage here for the attacker is stealth, which allows him or her to bypass the IDS.

Today, most IDSs, operating systems, and firewalls have antifragmentation defenses. By default, a Windows host will drop fragmented packets.

Application Attacks

Although network protocol attacks abound, most security threats exploit the host's application layer. In these cases, the TCP/IP packets are constructed legitimately, but their data payload contains malicious content. Application attacks can be text commands used to exploit operating system or application holes, or they can contain malicious content such as a buffer overflow exploit, a maliciously crafted command, or a computer virus. Application attacks include misappropriated passwords, cross-site scripting, malicious URLs, password-cracking attempts, rootkit software, illegal data manipulation, unauthorized file access, and every other attack that doesn't rely on malformed network packets to work.

The major problem is that the majority of these attacks are allowed by the firewall because they are carried over legitimate services like port 80 (HTTP) or port 25 (SMTP) without their contents being checked.

Content Obfuscation Most IDSs look for known malicious commands or data in a network packet's data payload. A byte-by-byte comparison is done between the payload and each potential threat signature in the IDS's database. If something matches, it's flagged as an event. This is how "signature-based" IDSs work. Someone has to have the knowledge to write the "signature."

Because byte scanning is relatively easy to do, attackers use encoding schemes to hide their malicious commands and content. *Encoding* schemes are non-plaintext character representations that eventually get converted to plaintext for processing. The flexibility of the coding for international languages on the Internet allows ASCII characters to be represented by many different encoding schemes, including hexadecimal (base 16, in which the word "Hello" looks like "48 65 6C 6C 6F"), decimal notation (where "Hello" is "72 101 108 108 111"), octal (base 8, in which "Hello" appears as "110 145 154 154 157"), Unicode (where "Hello" = "0048 0065 006C 006C 006F"), and any combination thereof. Web URLs and commands have particularly flexible syntax. Complicating the issue, most browsers encountering common syntax mistakes, like reversed slashes or incorrect case, convert them to their legitimate form. Here is an example of one URL presented in different forms with syntax mistakes and encoding. Type them into your browser and see for yourself.

- http://www.mcgraw-hill.com (normal representation)
- http:\\198.45.19.151 (IP address and wrong slashes)
- http://%77%77%77%2E%6D%63%67%72%61%77%2D%68%69%6C%6C%2E%63%6F%6D (hexadecimal encoded)

NOTE It is not unusual to see a few characters of encoding in a legitimate URL. When you see mostly character encoding, however, you should get suspicious.

Encoding can be used to obscure text and data used to create malicious commands. Attackers employ all sorts of tricks to fool IDSs, including using tabs instead of spaces, changing values from lowercase to uppercase, splitting data commands into several different packets sent over a long period of time, hiding parameters, prematurely ending requests, using excessively long URLs, and using text delimiters.

Data Normalization An IDS signature database has to consider all character encoding schemes and tricks that can end up creating the same malicious pattern. This task is usually accomplished by normalizing the data before inspection. Normalization reassembles fragments into single whole packets, converts encoded characters into plain ASCII text, fixes syntax mistakes, removes extraneous characters, converts tabs to spaces, removes common hacker tricks, and does its best to convert the data into its final intended form.

Threats an IDS Cannot Detect

IDSs excel at catching known, definitive malicious attacks. Although some experts will say that a properly defined IDS can catch any security threat, events involving misuse prove the most difficult to detect and prevent. For example, if an outside hacker uses social engineering tricks to get the CEO's password, not many IDSs will notice. If the webmaster accidentally posts a confidential document to a public directory available to the world, the IDS won't notice. If an attacker uses the default password of an administrative account that should have been changed right after the system was installed, few IDSs will notice. If a hacker gets inside the network and copies confidential files, an IDS would have trouble noticing it. That's not to say you can't use an IDS to detect each of the preceding misuse events, but they are more difficult to detect than straight-out attacks. The most effective way for an attacker to bypass the visibility of an IDS is to encrypt the traffic at many layers (layers two, three, and through seven). For example, using OpenSSH or SSL would encrypt most of the data, whereas using IPSec would encrypt the traffic in transit.

First-Generation IDS

IDS development as we know it today began in the early 1980s, but only started growing in the PC marketplace in the late 1990s. First-generation IDSs focused almost exclusively on the benefit of early warning resulting from accurate detection. This continues to be a base requirement of an IDS, and vendors frequently brag about their product's accuracy. The practical reality is that while most IDSs are considered fairly accurate, no IDS has ever been close to being perfectly accurate. Although a plethora of antivirus scanners enjoy year-after-year 95 to 99 percent accuracy rates, IDSs never get over 90 percent accuracy against a wide spectrum of real-world attack traffic. Most are in the 80 percent range. Some test results show 100 percent detection rates, but in every such instance, the IDS was tuned after several previous, less accurate rounds of testing. When an IDS misses a legitimate threat, it is called a *false negative*. Most IDS are plagued with even higher false positive rates, however.

IDSs have high false positive rates. A false positive is when the IDS says there is a security threat by "alerting," but the traffic is not malicious or was never intended to be malicious (benign condition). A common example is when an IDS flags an e-mail as infected with a particular virus because it is looking for some key text known to be in the message body of the e-mail virus (for example, the phrase "cheap pharmaceuticals"). When an e-mail intended to warn readers about the virus includes the keywords that the reader should be on the lookout for, it can also create a false positive. The IDS should be flagging the e-mail as infected only if it actually contains a virus, not just if it has the same message text.

Simply searching for text within the message body to detect malware is an immature detection choice. Many security web services that send subscribers early warning e-mails complain that nearly 10 percent of their e-mails are kicked back by overly zealous IDSs. Many of those same services have taken to misrepresenting the warning text purposely (by slightly changing the text, such as "che4p_pharmaceut1cals") in a desperate attempt to get past the subscribers' poorly configured defenses. If the measure of IDS accuracy is the number of logged security events against legitimate attacks, accuracy plummets on most IDS products. This is the biggest problem facing IDSs, and solving it is considered the holy grail for IDS vendors. If you plan to get involved with IDSs, proving out false positives will be a big part of your life.

In an effort to decrease false positives, some IDSs are tuned to be more sensitive. They will wait for a highly definitive attack within a narrow set of parameters before they alert the administrator. Although they deliver fewer false positives, they have a higher risk of missing a legitimate attack. Other IDSs go the other route and report on almost everything. Although they catch more of the legitimate threats, those legitimate warnings are buried in the logs between tons of false positives. If administrators are so overwhelmed with false positives that they don't want to read the logs, they can create a "human denial of service" attack. Some attackers attempt to do just this by generating massive numbers of false positives, hoping the one legitimate attack goes unnoticed.

Which is a better practice? Higher false positives or higher false negatives? Most IDS products err on the side of reporting more events and requiring the user to fine-tune the IDS to ignore frequent false positives. Fine-tuning an IDS means configuring sensitivity up or down to where you, the administrator, are comfortable with the number of false negatives and false positives. When you are talking with vendors or reviewing IDS products, inquire about which detection philosophy the IDS follows. If you don't know ahead of time, you'll know after you turn it on.

Second-Generation IDS

The net effect of most IDSs being fairly accurate and none being highly accurate has resulted in vendors and administrators using other IDS features for differentiation. Here are some of those other features that may be more or less useful in different circumstances:

- IDS type and detection model
- End-user interface
- IDS management
- Prevention mechanisms
- Performance
- Logging and alerting
- Reporting and analysis

All of these are discussed in this chapter.

First-generation IDSs focused on accurate attack detection. *Second-generation* IDSs do that and work to simplify the administrator's life by offering a bountiful array of back-end options. They offer intuitive end-user interfaces, intrusion prevention, centralized device management, event correlation, and data analysis. Second-generation IDSs do more than just detect attacks—they sort them, prevent them, and attempt to add as much value as they can beyond mere detection.

Experienced IDS administrators know that half of the success or failure of an IDS is determined by time consuming, and very complicated, technical work. Catching an attacker hacking in real-time is always exciting, as is snooping on the snooper, so first-time implementers often spend most of their time learning about and implementing detection patterns. In doing so, though, they often breeze through or skip the reading on setting up the management features, configuring the database, and printing reports. They turn on their IDS and are quickly overwhelmed because they didn't plan ahead.

TIP To increase your odds of a successful IDS deployment, remember this: For every hour you spend looking at cool detection signatures, spend an hour planning and configuring your logging, reporting, and analysis tools.

IDS Types and Detection Models

Depending on what assets you want to protect, an IDS can protect a host or a network. All IDSs follow one of two intrusion detection models—*anomaly* (also called *profile, behavior, heuristic,* or *statistical*) detection or *signature* (knowledge-based) detection—although some systems use parts of both when it's advantageous. Both anomaly and signature detection work by monitoring a wide population of events and triggering based on predefined behaviors.

Host-Based IDS

A *host-based IDS* (HIDS) is installed on the host it is intended to monitor. The host can be a server, workstation, or any networked device (such as a printer, router, or gateway). A HIDS installs as a service or daemon, or it modifies the underlying operating system's kernel or application to gain first inspection authority. Although a HIDS may include the ability to sniff network traffic intended for the monitored host, it excels at monitoring and reporting direct interactions at the application layer. Application attacks can include memory modifications, maliciously crafted application requests, buffer overflows, or file-modification attempts. A HIDS can inspect each incoming command, looking for signs of maliciousness, or simply track unauthorized file changes.

A *file-integrity* HIDS (sometimes called a *snapshot* or *checksum* HIDS) takes a cryptographic hash of important files in a known clean state and then checks them again later for comparison. If any changes are noted, the HIDS alerts the administrator that there may be a change in integrity.

A *behavior-monitoring* HIDS performs real-time monitoring and intercepts potentially malicious behavior. For instance, a Windows HIDS reports on attempts to modify the registry, manipulate files, access the system, change passwords, escalate privileges, and otherwise directly modify the host. On a Unix host, a behavior-monitoring HIDS may monitor attempts to access system binaries, attempts to download password files, and change permissions and scheduled jobs. A behavior-monitoring HIDS on a web server may monitor incoming requests and report maliciously crafted HTML responses, cross-site scripting attacks, or SQL injection code.

Real-Time or Snapshot?

Early warning and prevention are the greatest advantages of a *real-time* HIDS. Because a real-time HIDS is always monitoring system and application calls, it can stop potentially malicious events from happening in the first place. On the downside, real-time monitoring takes up significant CPU cycles, which may not be acceptable on a high-performance asset, like a popular web server or a large database server. Real-time behavior-monitoring only screens previously defined threats, and new attack vectors are devised several times a year, meaning that real-time monitors must be updated, much like databases for an antivirus scanner. In addition, if an intrusion successfully gets by the real-time behavior blocker, the HIDS won't be able to provide as much detailed information about what happened thereafter as a snapshot HIDS would.

Snapshot HIDSs are reactive by nature. They can only report on maliciousness, not stop it. A snapshot HIDS excels at forensic analysis. With one report, you can capture all the changes between a known good state and the corrupted state. You will not have to piece together several different progressing states to see all the changes made since the baseline. Damage assessment is significantly easier than with a real-time HIDS because a snapshot HIDS can tell you exactly what has changed. You can use comparative reports to decide whether you have to rebuild the host completely or whether a piecemeal restoration can be done safely. You can also use the before and after snapshots as forensic evidence in an investigation.

Snapshot systems are useful outside the realm of computer security, too. You can use a snapshot system for configuration and change management. A snapshot can be valuable when you have to build many different systems with the same configuration settings as a master copy. You can configure the additional systems and use snapshot comparison to see if all configurations are identical. You can also run snapshot reports later to see if anyone has made unauthorized changes to a host. The obvious disadvantage of a snapshot HIDS is that alerting and reporting is done after the fact. By then, the changes have already occurred, and the damage is done.

Network-Based IDS (NIDS)

Network-based IDSs (NIDSs) are the most popular IDSs, and they work by capturing and analyzing network packets speeding by on the wire. Unlike a HIDS, a NIDS is designed to protect more than one host. It can protect a group of computer hosts, like a server farm, or monitor an entire network. Captured traffic is compared against protocol specifications and normal traffic trends or the packet's payload data is examined for malicious content. If a security threat is noted, the event is logged and an alert is generated.

With a HIDS, you install the software on the host you want monitored and the software does all the work. Because a NIDS works by examining network packet traffic, including traffic not intended for the NIDS host on the network, it has a few extra deployment considerations. It is common for brand-new NIDS users to spend hours wondering why their IDS isn't generating any alerts. Sometimes it's because there is no threat traffic to alert on, and other times it's because the NIDS isn't set up to capture packets headed to other hosts.

A sure sign that the network layer of your NIDS is misconfigured is that it only picks up broadcast traffic and traffic headed for it specifically. Traffic doesn't start showing up at the NIDS simply because it was turned on. You must configure your NIDS and the network so the traffic you want to examine is physically passed to the NIDS. NIDSs must have promiscuous network cards with packet-level drivers, and they must be installed on each monitored network segment. Network taps, a dedicated appliance used to mirror a port or interface physically, and Switch Port Analysis (SPAN), are the two most common methods for setting up monitoring on a switched network.

Packet-Level Drivers

Network packets are captured using a packet-level software driver bound to a network interface card. Many Unix and Windows systems do not have native packet-level drivers built in, so IDS implementations commonly rely on open source packet-level drivers. Most commercial IDSs have their own packet-level drivers and packet-sniffing software.

Promiscuous Mode

For a NIDS to sniff packets, the packets have to be given to the packet-level driver by the network interface card. By default, most network cards are not *promiscuous,* meaning they only read packets off the wire that are intended for them. This typically includes *unicast* packets, meant solely for one particular workstation, *broadcast* packets, meant for every computer that can listen to them, and *multicast* traffic, meant for two or more previously defined hosts. Most networks contain unicast and broadcast traffic. Multicast traffic isn't as common, but it is gaining in popularity for web-streaming applications. By default, a network card in normal mode drops traffic destined for other computers and packets with transmission anomalies (resulting from collisions, bad cabling, and so on). If you are going to set up an IDS, make sure its network interface card has a *promiscuous mode* and is able to inspect all traffic passing by on the wire.

Sensors for Network Segments

For the purposes of this chapter, a *network segment* can be defined as a single logical packet domain. For a NIDS, this definition means that all network traffic heading to and from all computers on the same network segment can be physically monitored.

You should have at least one NIDS inspection device per network segment to monitor a network effectively. This device can be a fully operational IDS interface or, more commonly, a router or switch interface to which all network traffic is copied, known as a *span port,* or a traffic repeater device, known as a *sensor* or *tap.* One port plugs into the middle of a connection on the network segment to be monitored, and the other plugs into a cable leading to the central IDS console.

NOTE Like a tap, a span port does not readily reveal itself to attackers who might otherwise note the IDS's presence.

Routers are the edge points of network segments, and you must place at least one sensor on each segment you wish to monitor. Most of today's networks contain switch devices. With the notable exception of broadcast packets, switches only send packets to a single destination port. On a switched network, an IDS will not see its neighbor's non-broadcast traffic. Many switches support *port mirroring,* also called *port spanning* or *traffic redirection.* Port mirroring is accomplished by instructing the switch to copy all traffic to and from a specific port to another port where the IDS sits.

Anomaly-Detection (AD) Model

Anomaly detection (AD) was proposed in 1985 by noted security laureate Dr. Dorothy E. Denning, and it works by establishing accepted baselines and noting exceptional differences. Baselines can be established for a particular computer host or for a particular network segment. Some IDS vendors refer to AD systems as *behavior-based* since they look for deviating behaviors. If an IDS looks only at network packet headers for differences, it is called *protocol anomaly detection*.

Several IDSs have anomaly-based detection engines. Several massively distributed AD systems monitor the overall health of the Internet, and a handful of high-risk Internet threats have been minimized over the last few years because unusual activity was noticed by a large number of correlated AD systems.

The goal of AD is to be able to detect a wide range of malicious intrusions, including those for which no previous detection signature exists. By learning known good behaviors during a period of "profiling," in which an AD system identifies and stores all the normal activities that occur on a system or network, it can alert to everything else that doesn't fit the normal profile. Anomaly detection is statistical in nature and works on the concept of measuring the number of events happening in a given time interval for a monitored metric. A simple example is someone logging in with the incorrect password too many times, causing an account to be locked out and generating a message to the security log. Anomaly detection IDS expands the same concept to cover network traffic patterns, application events, and system utilization. Here are some other events AD systems can monitor and trigger alerts from:

- Unusual user account activity
- Excessive file and object access
- High CPU utilization
- Inappropriate protocol use
- Unusual workstation login location
- Unusual login frequency
- High number of concurrent logins
- High number of sessions
- Any code manipulation
- Unexpected privileged use or escalation attempts
- Unusual content

An accepted baseline may be that network utilization on a particular segment never rises above 20 percent and routinely only includes HTTP, FTP, and SMTP traffic. An AD baseline might be that there are no unicast packets between workstations and only unicasts between servers and workstations. If a DoS attack pegs the network utilization above 20 percent for an extended period of time, or someone tries to telnet to a server on a monitored segment, the IDS would create a security event. Excessive repetition of identical characters in an HTTP response might be indicative of a buffer overflow attempt.

When an AD system is installed, it monitors the host or network and creates a monitoring policy based on the learned baseline. The IDS or installer chooses which events to measure and how long the AD system should measure to determine a baseline. The installer must make sure that nothing unusual is happening during the sampling period that might skew the baseline.

Anomalies are empirically measured as a statistically significant change from the baseline norm. The difference can be measured as a number, a percentage, or as a number of standard deviations. In some cases, like the access of an unused system file or the use of an inactive account, one instance is enough to trigger the AD system. For normal events with ongoing activity, two or more statistical deviations from the baseline measurement creates an alert.

AD Advantages

AD systems are great at detecting a sudden high value for some metric. For example, when the SQL Slammer worm ate up all available CPU cycles and bandwidth on affected servers and networks within seconds of infection, you can bet AD systems went off. They did not need to wait until an antivirus vendor released an updated signature. As another example, if your AD system defines a buffer overflow as any traffic with over a thousand repeating characters, it will catch any buffer overflow, known or unknown, that exceeds that definition. It doesn't need to know the character used or how the buffer overflow works. If your AD system knows your network usually experiences ten FTP sessions in a day, and suddenly it experiences a thousand, it will likely catch the suspicious activity.

AD Disadvantages

Because AD systems base their detection on deviation from what's normal, they tend to work well in static environments, such as on servers that do the same thing day in and day out, or on networks where traffic patterns are consistent throughout the day. On more dynamic systems and networks that, therefore, have a wider range of normal behaviors, false positives can occur when the AD triggers on something that wasn't captured during the profiling period.

Signature-Detection Model

Signature-detection or *misuse* IDSs are the most popular type of IDS, and they work by using databases of known bad behaviors and patterns. This is nearly the exact opposite of AD systems. When you think of a signature-detection IDS, think of it as an antivirus scanner for network traffic. Signature-detection engines can query any portion of a network packet or look for a specific series of data bytes. The defined patterns of code are called *signatures,* and often they are included as part of a governing *rule* when used within an IDS.

Signatures are byte sequences that are unique to a particular malady. A byte signature may contain a sample of virus code, a malicious combination of keystrokes used in a buffer overflow, or text that indicates the attacker is looking for the presence of a particular file in a particular directory. For performance reasons, the signature must be crafted so it is the shortest possible sequence of bytes needed to detect its related threat reliably. It must be highly accurate in detecting the threat and not cause false positives. Signatures and rules can be collected together into larger sets called *signature databases* or *rule sets.*

Signature-Detection Rules

Rules are the heart of any signature-detection engine. A rule usually contains the following information as a bare minimum:

- Unique signature byte sequence
- Protocol to examine (such as TCP, UDP, ICMP)
- IP port requested
- IP addresses to inspect (destination and source)
- Action to take if a threat is detected (such as allow, deny, alert, log, disconnect)

Most IDSs come with hundreds of predefined signatures and rules. They are either all turned on automatically or you can pick and choose. Each activated rule or signature adds processing time for analyzing each event. If you were to turn on every rule and inspection option of a signature-detection IDS, you would likely find it couldn't keep up with traffic inspection. Administrators should activate the rules and options with an acceptable cost/benefit tradeoff.

Most IDSs also allow you to make custom rules and signatures, which is essential for responding immediately to new threats or for fine-tuning an IDS. Here are some hints when creating rules and signatures:

- Byte signatures should be as short as possible, but reliable, and they should not cause false positives.
- Similar rules should be near each other. Organizing your rules speeds up future maintenance tasks.
- Some IDSs and firewalls require rules that block traffic to appear before rules that allow traffic. Check with your vendor to see if rule placement matters.
- Create wide-sweeping rules that do the quickest filtering first. For example, if a network packet has a protocol anomaly, it should cause an alert event without the packet ever getting to the more processor-intensive content scanning.
- To minimize false positives, rules should be as specific as possible, including information that specifically narrows down the population of acceptable packets to be inspected.

Some threats, like polymorphic viruses or multiple-vector worms, require multiple signatures to identify the same threat. For instance, many computer worms arrive as infected executables, spread over internal drive shares, send themselves out with their own SMTP engines, drop other Trojans and viruses, and use Internet chat channels to spread. Each attack vector would require a different signature.

Advantages of Signature Detection

Signature-detection IDSs are proficient at recognizing known threats. Once a good signature is created, signature detection IDSs are great at finding patterns, and because they are popular, a signature to catch a new popular attack usually exists within hours of it first being reported. This applies to most open source and commercial vendors.

Another advantage of a signature-detection IDS is that it will specifically identify the threat, whereas an AD engine can only point out a generality. An AD IDS might alert you that a new TCP port opened on your file server, but a signature-detection IDS will tell you what exploit was used. Because a signature-detection engine can better identify specific threats, it has a better chance at providing the correct countermeasure for intrusion prevention.

Disadvantages of Signature Detection

Although signature-detection IDS are the most popular type of IDS, they have several disadvantages as compared to an AD IDS.

Cannot Recognize Unknown Attacks Just like antivirus scanners, signature-detection IDSs are not able to recognize previously unknown attacks. Attackers can change one byte in the malware program (creating a variant) to invalidate an entire signature. Hundreds of new malware threats are created every year, and signature-based IDSs are always playing catch up. To be fair, there hasn't been a significant threat in the last few years that didn't have a signature identified by the next day, but your exposure is increased in the so-called zero-hour.

Performance Suffers as Signatures or Rules Grow Because each network packet or event is compared against the signature database, or at least a subset of the signature database, performance suffers as rules increase. Most IDS administrators using signature detection usually end up only using the most common signatures and not the less common rules. The more helpful vendors rank the different rules with threat risks so the administrator can make an informed risk tradeoff decision. Although this is an efficient use of processing cycles, it does decrease detection reliability.

Some vendors are responding by including *generic* signatures that detect more than one event. To do so, their detection engines support wildcards to represent a series of bytes, like this:

Virus A has a signature of	`14 90 90 90 56 76 56 64 64`
Virus B has a signature of	`14 80 90 90 56 76 56 13 10`
A wildcard signature for viruses A and B is	`14 ? 90 90 56 76 56 * *`

Of course, the use of wildcard signatures increases the chance of false positives. Antivirus vendors faced a similar dilemma last decade and called viruses generic boot sector or generic file infectors. Some vendors went so far that they rarely identified any threat by its specific name. Security administrators were not happy with the results, and vendors had to return to using signatures that are more specific.

Because a signature is a small, unique series of bytes, all a threat coder has to do is change one byte that is identified in the signature to make the threat undetectable. Threats with small changes like these are called *variants*. Luckily, most variants share some common portion of code that is still unique to the whole class of threats, so that one appropriate signature, or the use of wildcards, can identify the whole family.

What Type of IDS Should You Use?

There are dozens of IDSs to choose from. The first thing you need to do is survey the computer assets you want to protect and identify the most valuable computer assets that should get a higher level of security assurance. These devices are usually the easiest ones to use when making an ROI case to management. New IDS administrators should start small, learn, fine-tune, and then grow. Don't try to boil the ocean. A HIDS should be used when you want to protect a specific valuable host asset. A NIDS should be used for general network awareness and as an early warning detector across multiple hosts.

You need to pick an IDS that supports your network topology, operating system platforms, budget, and experience. If you have a significant amount of wireless traffic exposed in public areas, consider investing in a wireless IPS. If you have high-speed links that you need to monitor, make sure your IDS has been rated and tested at the same traffic levels.

Should your IDS be based on anomaly or signature detection? When possible, use a product that does both. The best IDSs utilize all techniques, combining the strengths of each type to provide a greater defense strategy.

IDS Features

As discussed earlier in the chapter, IDSs are more than detection engines. Detection is their main purpose, but if you can't configure the system or get the appropriate information out of the IDS, it won't be much help. This section discusses the end-user interface, IDS management, intrusion prevention, performance, logging and alerting, and reporting and data analysis.

IDS End-User Interfaces

IDS end-user interfaces let you configure the product and see ongoing detection activities. You should be able to configure operational parameters, rules, alert events, actions, log files, and update mechanisms. IDS interfaces come in two flavors: syntactically difficult command prompts or less-functional GUIs.

Historically, IDSs are command-line beasts with user-configurable text files. Command-line consoles are available on the host computer or can be obtained by a Telnet session or proprietary administrative software. The configuration files control the operation of the IDS detection engine, define and hold the detection rules, and contain the log files and alerts. You configure the files, save them, and then run the IDS. If any runtime errors appear, you have to reconfigure and rerun. A few of the command-line IDS programs have spawned GUI consoles that hide the command-line complexities.

NOTE A frequent complaint of new GUI IDS users is that once the IDS is turned on, "nothing happens!" This is because the IDS is not detecting any defined threats, not placed appropriately in the network topology to be able to sniff traffic, or not configured to display events to the screen (because doing so wastes valuable CPU cycles).

Although text-based user interfaces may be fast and configurable, they aren't loved by the masses. Hence, more and more IDSs are coming with user-friendly GUIs that make installation a breeze and configuration a matter of point-and-click. With few exceptions, the GUIs tend to be less customizable than their text-based cousins and, if connected to the detection engine in real time, can cause slowness. Many of the GUI consoles present a pretty picture to the end-user but end up writing settings to text files, so you get the benefits of both worlds.

Intrusion-Prevention Systems (IPS)

Since the beginning, IDS developers have wanted the IDS to do more than just monitor and report maliciousness. What good is a device that only tells you you've been maligned when the real value is in preventing the intrusion? That's like a car alarm telling you that your car has been stolen, after the fact. Like intrusion detection, intrusion prevention has long been practiced by network administrators as a daily part of their routine. Setting access controls, requiring passwords, enabling real-time antivirus scanning, updating patches, and installing perimeter firewalls are all examples of common intrusion-prevention controls. Intrusion-prevention controls, as they apply to IDSs, involve real-time countermeasures taken against a specific, active threat. For example, the IDS might notice a ping flood and deny all future traffic originating from the same IP address. Alternatively, a host-based IDS might stop a malicious program from modifying system files.

Going far beyond mere monitoring and alerting, second-generation IDSs are being called *intrusion-prevention systems* (IPSs). They either stop the attack or interact with an external system to put down the threat.

If the IPS, as shown in Figure 18-1, is a mandatory inspection point with the ability to filter real-time traffic, it is considered *inline*. Inline IPSs can drop packets, reset connections, and route suspicious traffic to quarantined areas for inspection. If the IPS isn't inline and is only inspecting the traffic, it still can instruct other network perimeter systems to stop an

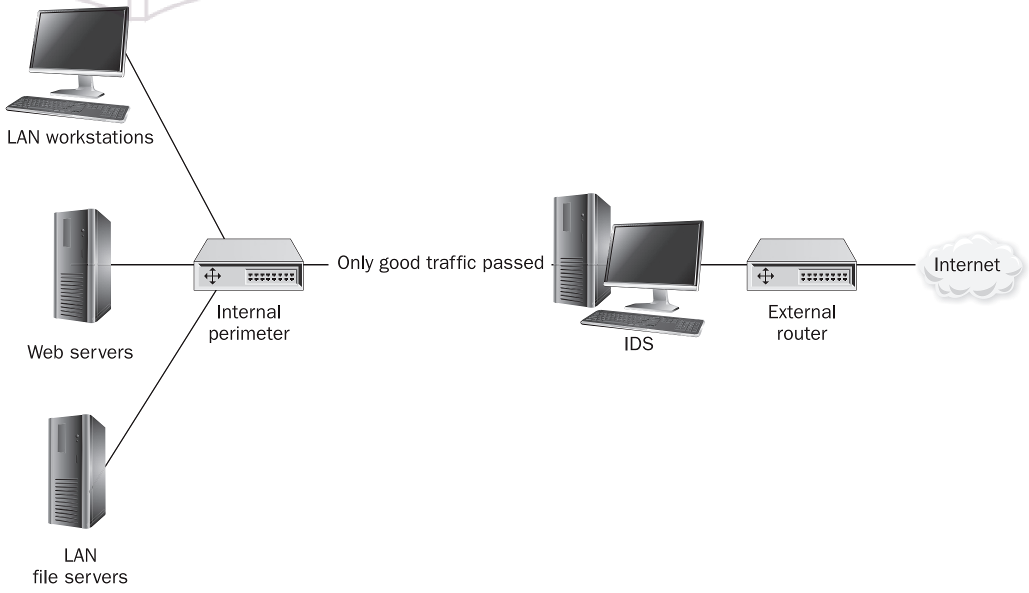

Figure 18-1 IDS placed to drop malicious packets before they can enter the network

exploit. It may do this by sending scripted commands to a firewall, instructing it to deny all traffic from the remote attacker's IP address, calling a virus scanner to clean a malicious file, or simply telling the monitored host to deny the hacker's intended modification.

For an IPS to cooperate with an external device, they must share a common scripting language, API, or some other communicating mechanism. Another common IPS method is for the IDS device to send reset (RST) packets to both sides of the connection, forcing both source and destination hosts to drop the communication. This method isn't seen as being very accurate, because often the successful exploit has happened by the time a forced reset has occurred, and the sensors themselves can get in the way and drop the RST packets.

IPS Disadvantages

A well-known consequence of IPSs is their ability to exacerbate the effects of a false positive. With an IDS, a false positive leads to wasted log space and time, as the administrator researches the threat's legitimacy. IPSs are proactive, and a false positive means a legitimate service or host is being denied. Malicious attackers have even used prevention countermeasures as a DoS attack.

Is It a Firewall or an IPS?

With the growing importance of intrusion prevention, most firewalls are beginning to look a lot like IPSs, and IPSs can look a lot like firewalls. Although there is no hard and fast rule, one way of distinguishing the two is that if the device inspects payload content to make its decision or identifies the exploit by name, it's an IPS. Historically, firewalls make decisions by IP address and port number (both source and destination) at layers three and four. IPSs can do that, but they can also identify the particular exploit if there is a previously defined pattern within layer 5 through layer 7.

An IPS can compile several different connection attempts, recognize that they were part of one port-scan event, and perhaps even identify the port-scanning tool that was used. A firewall would report each separate connection attempt as a separate event. IPSs have more expert knowledge and can identify exploits by popular name.

IDS Management

Central to the IDS field are the definitions of *management console* and *agent*. An IDS agent (which can be a *probe*, *sensor*, or *tap*) is the software process or device that does the actual data collection and inspection. If you plan to monitor more than two network segments, you can separately manage multiple sensors by connecting them to a central management console. This allows you to concentrate your IDS expertise at one location.

IDS management consoles usually fulfill two central roles: configuration and reporting. If you have multiple agents, a central console can configure and update multiple distributed agents at once. For example, if you discover a new type of attack, you can use the central console to update the attack definitions for all sensors at the same time. A central console also aids in determining agent status—active and online or otherwise.

NOTE If the management console and sensors run on different machines, traffic between the two should be protected. This is often accomplished using SSL or a proprietary vendor method.

In environments with more than one IDS agent, reporting captured events to a central console is crucial. This is known as event *aggregation*. If the central console attempts to organize seemingly distinct multiple events into a smaller subset of related attacks, it is known as event *correlation*. For example, if a remote intruder port-scans five different hosts, each running its own sensor, a central console can combine the events into one larger event. To aid in this type of correlation analysis, most consoles allow you to sort events by

- Destination IP address
- Source IP address
- Type of attack
- Type of protocol
- Time of attack

You can also customize the policy that determines whether two separate events are related. For example, you can tell the console to link all IP fragmentation attacks in the last five minutes into one event, no matter how many source IP addresses were involved. Agents are configured to report events to the central console, and then the console handles the job of alerting system administrators. This centralization of duties helps with setting useful alert thresholds and specifying who should be alerted. Changes to the alert notification list can be made on one computer instead of on numerous distributed agents.

A management console can also play the role of expert analyzer. A lightweight IDS performs the role of agent and analyzer on one machine. In larger environments with many distributed probes, agents collect data and send it to the central console without determining whether the monitored event was malicious or not. The central console manages the database, warehousing all the collected event data. As shown in Figure 18-2, the database may be maintained on a separate computer connected with a fast link.

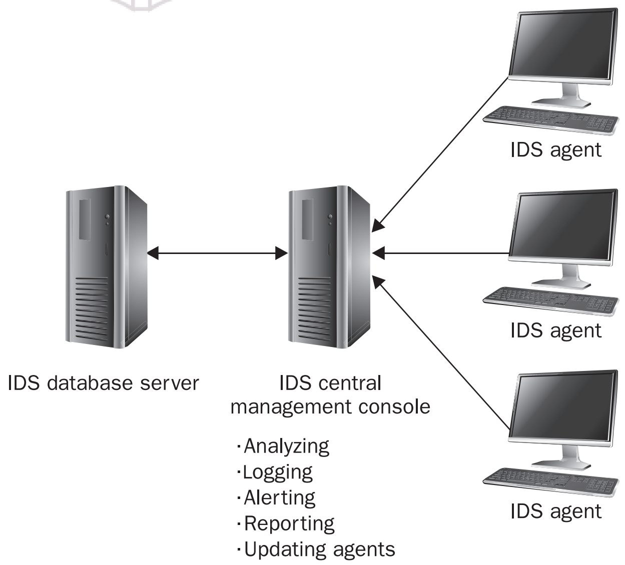

Figure 18-2 Example of a distributed IDS topology

Of course, having a central management console means having a single point of failure. If the management console goes down, alerts will not be passed on, and malicious traffic may not be recorded. Despite this risk, however, if you have more than one sensor, a management console is a necessity. Moreover, if a central console is helpful for managing multiple IDS sensors, it can also be helpful for managing information from even more computer security devices.

IDS Logging and Alerting

When security events are detected by an IDS, they generate alerts and log files.

Alerts

Alerts are high-priority events communicated to administrators in real time. The IDS's policy determines what security threats are considered high risk, and the priority level is set accordingly. Typically, you would not want an IDS administrator to respond as quickly to a NetBIOS scan against your appropriately firewalled network as you would to a successful DoS attack against your company's primary web server. When an event is considered high risk against a valuable asset, it should be communicated immediately.

Carefully contemplate what method should be used for communication. For example, most IDS alerts are sent via e-mail. In the case of a fast-spreading e-mail worm, the e-mail system will be severely taxed, and finding an alert message among thousands of other messages might be daunting. In fact, the alert may not even be delivered at all. SMS (text) messages can be a viable alternative, if they are delivered over a different path. In any case, it's a good idea to make sure alerts can get to you in more than one way.

Alerts should be quick and to the point; however, they need to contain enough information for the incident responder to track down the event. They should describe location, event, information about the source of the event and priority, and they should fit on a small display, like these two examples:

```
LAN3-1: Smurf attack-Medium
Corpweb3: DoS-High
```

More advanced IDS systems allow you to combine identical alerts occurring in a given time period as the same event. Although this might not seem important as you read this book, it becomes important to the administrator at 3 A.M. when one port scan turns into over a thousand different alerts in under a minute. Correlation thresholds allow a security administrator to be appropriately alerted for an event without feeling like the whole network is under siege.

Logs

IDS log files record all detected events regardless of priority and, after its detection engine, have the greatest influence on the speed and use of an IDS. IDS logs are used for data analysis and reporting. They can include just a barebones summary of events or a complete network packet decode. Although complete network traces are preferable for forensics, they can quickly take up a lot of hard drive space. A small network can generate hundreds of events a minute, and a mid-sized network can generate tens of thousands. If you plan to store multiple days' worth of logs with full packet decoding, make sure your IDS's hard drive is large enough.

Regardless of the log format an IDS uses, all log files must be rotated out frequently in order to maintain performance and to prevent lockups. Unfortunately, when you rotate a log file out, it complicates threat analysis, because you will have to merge multiple files to cover a greater time period.

At a minimum, a log file should record the event location, *timestamp* (date and time to the hundredth of a second, which is typically provided by your internal NTP server), description of the action attempted, criticality, and IDS response, if any. If the event was recorded using network packets, then the following additional information should be noted: source and destination IP addresses, protocol, and port number. The log should provide a short description of the attack and give links to the vendor or other vulnerability web sites for a more detailed explanation.

NOTE Reporting event timestamps in Coordinated Universal Time (UTC), also known as Greenwich Mean Time (GMT) or Zulu time, will simplify your task when reporting events to external authorities in different time zones. UTC is the worldwide standard for time reporting based on the "0" longitude meridian. All other time zones are based on adding or subtracting from UTC.

Most vulnerability databases describe the security event as if it can only be a malicious attack, when, in fact, this is often not true. IDS vendor databases should also list reasons why the reported event may be a false positive. For example, if the IDS reports an IP spoof event, it's helpful to read that IP spoofs can be created by poorly configured, but legitimate, VPN links. If you keep receiving port-scanning alerts that you trace to your ISP's DNS servers, learning that it is a normal behavior for them as they attempt to respond to misconfigured client workstations is helpful.

IDS Deployment Considerations

IDSs are beneficial tools, but they have weaknesses. They need to be fine-tuned if you want to maximize their usefulness, and if you intend to deploy one, you'll need to come up with a deployment plan to do so successfully. Creating this usually represents a substantial amount of work. This section summarizes these deployment issues.

IDS Fine-Tuning

Fine-tuning an IDS means doing three things: increasing inspection speed, decreasing false positives, and using efficient logging and alerting.

Increasing Inspection Speed

Most IDS administrators start off monitoring all packets and capturing full packet decodes. You can narrow down what packets an IDS inspects by telling it to include or ignore packets based on source and destination addresses. For example, if you are most concerned with protecting your servers, modify the IDS's packet inspection engine so it only captures packets with server destination addresses. Another common packet filter is a rule that excludes broadcast packets between routers. Routers are always busy chatting and broadcasting to learn routes and reconstruct routing tables, but if you aren't worried about internal ARP poisoning, don't capture ARP packets. The more packets the IDS can safely ignore, the faster it will be.

Another strategy is to let other faster perimeter devices do the filtering. Routers and firewalls are usually faster than IDSs, so, when possible, configure the packet filters of your routers and firewalls to deny traffic that should not be on your network in the first place. For example, tell your router to deny IP address spoofs, and tell your firewall to drop all NetBIOS traffic originating from the Internet. The more traffic that you can block with the faster device, the higher performing your IDS will be. That's the way it should be—each security device should be configured to excel at what it does best, at the layer from which it does it best.

Decreasing False Positives

Because IDS have so many false positives, the number one job of any IDS administrator is to track down and troubleshoot false positives. In most instances, false positives will outweigh all other events. Track them all down, rule out maliciousness, and then appropriately modify the source or IDS to prevent them. Often the source of the false positive is a misbehaving program or a chatty router. If you can't stop the source of the false positive, modify the IDS so it will not track the event. The key is that you want your logs to be as accurate as they can be, and they should only alert you to events that need human intervention. Don't get into the habit of ignoring the frequently occurring false positives in your logs as a way of doing business. This will quickly lead to your missing the real events buried inside all the false positives—or to the logs not being read at all.

Using Efficient Logging and Alerting

Most vendor products come with their own preset levels of event criticalities, but when setting up the IDS, take the time to customize the criticalities for your environment. For instance, if you don't have any Apache web servers, set Apache exploit notices with a low level of prioritization. Better yet, don't track or log them at all.

IPS Deployment Plan

So you want to deploy your first IPS. You've mapped your network, surveyed your needs, decided what to protect, and picked an IPS solution. Here are the steps to a successful IPS deployment:

1. Document your environment's security policy.
2. Define human roles.
3. Decide the physical location of the IPS and sensors.
4. Configure the IPS sensors and management console to support your security policy.
5. Plan and configure device management (including the update policy).
6. Review and customize your detection mechanisms.
7. Plan and configure any prevention mechanisms.
8. Plan and configure your logging, alerting, and reporting.
9. Deploy the sensors and console (do not encrypt communication between sensors and links to lessen troubleshooting).
10. Test the deployment using IPS testing tools (initially use very broad rules to make sure the sensors are working).
11. Encrypt communications between the sensors and console.

12. Test the IPS setup with actual rules.
13. Analyze the results and troubleshoot any deficiencies.
14. Fine-tune the sensors, console, logging, alerting, and reporting.
15. Implement the IPS system in the live environment in monitor-only mode.
16. Validate alerts generated from the IPS.
17. One at a time, set blocking rules for known reliable alerts that are important in your environment.
18. Continue adding blocking rules over time as your confidence in each rule increases.
19. Define continuing education plans for the IPS administrator.
20. Repeat these steps as necessary over the life of the IPS.

As you can see, installing and testing an IPS is a lot of work. The key is to take small steps in your deployment, and plan and configure all the parts of your IPS before just turning it on. The more time you spend on defining reporting and database mechanisms at the beginning, the better the deployment will go.

During the initial tests, in step 10, use a test rule that is sure to trigger the IPS sensor or console on every packet. This ensures that the physical part of the sensor is working and lets you test the logging and alerting mechanisms. Once you know the physical layer is working, you can remove that test rule (or comment it out or unselect it, in case you need it later). Do not turn on encryption, digital signing, or any other self-securing components until after you've tested the initial physical connections. This reduces troubleshooting time caused by mistyped passphrases or incorrectly configured security settings.

Finally, keep on top of your logs, and research all critical events. Quickly rule out false positives, and fine-tune your IPS on a regular basis to minimize false positives and false negatives. Once you get behind in your log duty, catching up again is tough. Successful IPS administrators track and troubleshoot everything as quickly as they can. The extra effort will pay dividends with smaller and more accurate logs.

Security Information and Event Management (SIEM)

Multiple security systems can report to a centralized *Security Information and Event Management (SIEM) system,* bringing together logs and alerts from several disparate sources. You may find different combinations of references to the acronym SIEM, owing to the evolution of capabilities and the consequent variety of names attached to SIEM products over the years, such as "Security Incident and Event Management" or "Security Incident and Event Monitoring." These are all the same thing—a technology to collect, analyze, and correlate events and alerts generated by monitoring systems.

SIEM platforms take the log files, find commonalities (such as attack types and threat origination), and summarize the results for a particular time period. For example, all logs and alerts from all IDSs, perimeter firewalls, personal firewalls, antivirus scanners, and operating systems can be tied together. Events from all logs are then gathered, analyzed, and reported on from one location. SIEMs offer the ultimate in event correlation, giving you one place to get a quick snapshot of your system's security or to get trend information. SIEMs can also coordinate signature and product updates.

SIEMs have a huge advantage over individual IDS systems because they have the capability to collect and analyze many different sources of information to determine what's really happening. As a result, the SIEM can significantly reduce false positives by verifying information based on other data. That data comes from many sources, including workstations, servers, computing infrastructure, databases, applications, network devices, and security systems. Because all those sources generate a vast amount of real-time data, SIEM products need to be fast and effective, with a significant amount of storage and computing power.

Today's network attacks are often complex—slow, multifaceted, and stealthy. Attackers use many techniques to circumvent security controls. Slow attacks can spread malicious network traffic over days, weeks, or even months, hiding inside the massive data streams experienced on any given network. Multifaceted attacks use a variety of techniques in the hope that at least one will succeed, or that the distributed nature of the attacks will distract attention away from the source. Stealthy attacks use obscure or nonstandard aspects of network technologies and protocols to slip past traditional monitoring capabilities that have been programmed based on the assumption that network traffic will always follow the normal standards. An IDS needs a SIEM to detect these advanced attacks.

A SIEM is one of the most important tools used by security operations and monitoring staff, because it provides one-stop visibility into many different areas of the information processing environment and attacks against those areas. Let's take a look at what a SIEM can do.

Data Aggregation

SIEMs collect information from every available source that is relevant to a security event. These sources take the form of alerts, real-time data, logs, and supporting data. Together, these provide the correlation engine of the SIEM with information it can use to make decisions about what to bring to the security administrator's attention. Consider the following examples of specific data sources consumed by a SIEM.

Alerts

When is an alert real, and when is it a false positive? This is the key question associated with an IDS, and a source of frustration for security administrators in charge of tuning IDSs. This is where a SIEM enters the picture. The SIEM's key function is to validate security alerts using many different sources of data to reduce false positives, so only the most reliable alerts get sent on to the security administrator. Thus, the alerts from all IDS sources as well as all other security monitoring systems should be given only to the SIEM, so it can decide which ones to pass along.

Real-Time Data

Real-time data such as network flow data (for instance, Cisco's NetFlow and similar traffic monitoring protocols from other vendors) gives the SIEM additional information to correlate. Streaming this data into the SIEM provides important information about normal and abnormal traffic patterns that can be used in conjunction with alerts to determine whether an attack is in progress. For example, an unusually high amount of SMTP traffic that accompanies several malware alerts may result in a high confidence alert that an e-mail worm is on the loose. Similarly, an abnormally high amount of inbound Internet traffic, combined with a high number of firewall deny events, can indicate a denial of service attack. Another example is fragmented or truncated network packets, which may indicate a network-based attack. Each of these real-time data elements gives the SIEM important validation data for IDS alerts.

Logs

Logs are different from events, in that they are a normal part of system activity and usually meant for debugging purposes. Logs can be an important additional data source for a SIEM, however. Logs contain valuable information about what's happening on a system, and they can give the SIEM a deeper view into what's happening. For example, login failures that may otherwise go unnoticed by a system administrator because they are buried in a system log might be of great interest to a SIEM, especially if there are many login failures for a single account (indicating a possible focused attempt to break into that account) or, similarly, if there are login failures on many different accounts, which may indicate a broad-based attempt to break into accounts using common passwords. System errors that are logged and collected by a SIEM are also a valuable source of correlating information.

In addition to providing the SIEM itself with detailed information, logs can be used to make decisions about the validity of IDS alerts and they are easier for humans to view in a SIEM. The system administrator who needs to find a particular log entry may find the SIEM is the best option for searching and finding that log entry.

Ideal log sources for any SIEM include the following:

- End-user computers
- Windows and Unix servers
- Domain controllers
- DNS and DHCP servers
- Mail servers
- Databases
- Web servers
- Applications
- Switches and routers
- VPN concentrators
- Firewalls
- Web filters and proxies
- Antivirus

Logs can be sent to the SIEM in a couple of different ways: they can be pushed to the SIEM by the individual devices that collect the logs, or they can be pulled in by the SIEM itself. The syslog protocol, which is widely used by Unix systems as well as network devices, is an example of a push technique. When the IP address of the SIEM is configured in the syslog service of a server or device, each log entry that device produces will be sent over the network to the SIEM. For systems that don't support syslog, such as Windows, third-party software can be used to collect static log information and send it to the SIEM. The third-party software agent can be installed directly on the reporting server, or on a central server built for log collection, in which case the software periodically connects to the server, grabs the latest log entries, and pushes them to the SIEM.

Whether pushed or pulled, log entries need to be parsed. Every vendor has a different format for the fields in their syslog data. Even though they all use the same protocol, the information contained within the log is not standardized. Modern SIEM products come with dozens of parsers that have been preconfigured to convert the syslog fields of different manufacturers into a format the SIEM can use. In the rare cases where a built-in parser is not available for a particular vendor's syslog format, the SIEM allows the administrator to define a custom mapping.

Supporting Data

You can enhance the quality of a SIEM's correlation even more by providing the SIEM with supporting data that has been previously collected. Data can be imported into the SIEM, and it will use that data to make comparative determinations.

For example, asset management data containing names, IP addresses, operating systems, and software versions gives the SIEM valuable information it can use to determine whether an IDS alert makes sense within the context of the software environment. Coupled with risk weighting data, the SIEM can use this information to prioritize and escalate alerts that pertain to high-risk systems. You can also use vulnerability scans to give the SIEM information it can use to compare an alert about an exploit with an associated vulnerability to determine if the exploit is real and whether it was successful. Moreover, geolocation information can be used to prioritize alerts from high-risk countries, or even local areas such as the datacenter or public hotspots in which mobile devices might be attacked.

Analysis

A SIEM takes all the data given to it and makes decisions, so the security administrator can focus on the most important alerts. For this reason, event correlation is a SIEM's most important feature. The correlation engine of every SIEM product is its most distinguishing feature. The better the analysis, the cleaner the end result. In effect, a SIEM is a sort of artificial intelligence system, working much like the human brain in putting together different elements that individually may not be important, but taken together form a picture of a critical security situation. And a SIEM does this at a much faster rate than any human possibly could, giving the security administrator a time advantage so he or she can react quickly to attacks in progress.

Real-time analysis of security events is only made possible with a SIEM. Thousands, or even millions, of events occur every second across most networks. No human can hope to see, absorb, and understand all of them at once. By comparison, forensic investigations in which the investigator looks at a few different data sources to decide who did what and when often take weeks of intense, focused effort. That's too long a timeframe for effective response to an attack. To stop an attack in progress, real-time analysis is required.

Because it collects so much data from across the enterprise, a SIEM can do more than alert. It can provide system and network administrators with advanced search capabilities they will not find on any other platform. For this reason, the SIEM represents an excellent shared platform that can make every administrator's job easier and more efficient. Thus, the SIEM is not just a security tool; it's also a valuable IT management tool.

The SIEM can also perform historical and forensic analysis based on the log information it collects. Depending on how much storage is allocated to the SIEM, either on-board or over the network, it can retain logs and alerts for a long enough period of time that it can investigate past events. Security investigators can dig into the logs to find out what happened in a prior situation, and system administrators can look at past events to troubleshoot and evaluate functional issues.

Operational Interface

For all the data collected by the SIEM and its resulting alerts to be human-readable, it must present the information in a way that an administrator can understand at a glance. SIEMs do this with a dashboard. A dashboard is a graphical and organized representation of alerts, event data, and statistical information that allows the administrator to see patterns, understand trends, identify unusual activity, and perceive the current threat landscape quickly at any point in time. The quality of a SIEM's dashboard is a key differentiator among the various SIEM products on the market.

Alerting is the other way the SIEM interacts with humans. Whereas the dashboard performs a pull type of data transfer to the administrator (because the administrator must go to the SIEM, log in, and intentionally look for the information), alerts represent a push technique that doesn't require human diligence to notice something important is happening. When a SIEM scores a series of events and the associated correlation of supporting information to be high enough, it sends an alert. The threshold for alerts should be set properly to ensure that only events that require action get the attention of the administrator, without excessive false positives. This is another reason a SIEM complements an IDS–the SIEM is more sophisticated than the IDS at creating appropriate alerts.

Additional SIEM Features

SIEMs provide additional value beyond collecting data and sending alerts. Because they collect and store so much data, SIEMs provide a natural advantage for offline log storage and retention, root cause analysis, advanced searching, and compliance reporting.

Offline log storage and retention is an important protection against tampering. Any time a system is compromised by an attacker, the attacker generally attempts to delete the traces of his or her activities by removing log entries, or even entire log files. When these logs are transmitted immediately from servers and devices to offline storage, in this case the SIEM, they cannot be tampered with because the SIEM stores them in a protected location that attackers cannot access. Log retention is also a compliance requirement for some organizations.

Analyzing the root cause of IT problems can be facilitated by all the information collected by a SIEM. Because it parses the log information into a standard format regardless of which product or technology produced the data, administrators can easily search individual data fields to find what they're looking for. In addition, because the timestamps are all normalized, you can easily see groups of events that happened together, even across disparate platforms.

The SIEM's advanced search capabilities provide another valuable advantage to system administrators. Imagine searching through individual system logs to find a particular piece of data you need on several different systems. Because system logs are generally not easily searchable, and systems don't typically provide sophisticated search capabilities, this can be a lot of work. And many technologies are standalone, without centralized log search ability. SIEMs provide administrators with that centralized ability to sift through the mountains of data produced by individual systems and devices to find what they're looking for without spending a lot of time and effort.

Finally, a SIEM can be employed to collect and report on compliance data for systems on the network. Because the SIEM has the most complete set of information about various aspects of the network that need to be monitored, reported, and audited, a SIEM's compliance reports are a great way to automate governance processes.

Summary

An intrusion detection system should be a part of every network security administrator's protection plan. An IDS provides the "detection" aspect of the three *D*s of security mentioned in Chapter 1, by providing visibility into activities, incidents, and intrusions. Along with other ID tools and methods, an IDS can monitor a host for system changes or sniff network packets off the wire, looking for malicious intent. A NIDS uses the same technology to make decisions about blocking network traffic. An IDS can be installed purely as a monitoring and detection device that sends alerts to administrators, who would then evaluate the situation and potentially take some action.

An IDS in blocking mode is known as IPS. Security administrators should consider using a combination of HIPS and NIPS, with both signature-detection and anomaly-based engines. An IPS's biggest weaknesses are the high number of false positives and the significant maintenance effort needed to keep it up to date and finely tuned so it doesn't block legitimate activities on systems and networks.

A HIPS would be appropriate on strategically valuable hosts, an IDS across the network for general early-warning detection, and an IPS for critical networks that need active protection. Central management consoles are helpful when multiple distributed agents are involved.

SIEM systems greatly enhance the accuracy, effectiveness, and completeness of IDS alerts. By themselves, individual IDS sensors can only see constrained segments of a network. Used in conjunction with a SIEM, multiple IDS sensors can provide much greater visibility. Reliability is also improved when a SIEM is used to collect and correlate alerts from IDSs and other sources, along with supporting data that has either been preconfigured into the SIEM or fed to it in real time. SIEMs also provide advanced capabilities that enhance the effectiveness of system, network, and security administrators.

References

Carter, Earl, and Jonathan Hogue. *Intrusion Prevention Fundamentals.* Cisco Press, 2006.

Fry, Chris, and Martin Nystrom. *Security Monitoring: Proven Methods for Incident Detection on Enterprise Networks.* O'Reilly Media, 2009.

Miller, David, and Shon Harris. *Security Information and Event Management (SIEM) Implementation.* McGraw-Hill, 2010.

National Institute of Standards and Technology. *Special Publication 800-94: Guide to Intrusion Detection and Prevention Systems (IDPS).* NIST, 2007. http://csrc.nist.gov/publications/nistpubs/800-94/SP800-94.pdf

Rash, Michael, and Angela Orebaugh. *Intrusion Prevention and Active Response: Deploying Network and Host IPS.* Syngress, 2005.

Trost, Ryan. *Practical Intrusion Analysis: Prevention and Detection for the Twenty-First Century.* Addison-Wesley, 2009.