CHAPTER 8

Intrusion Detection

Learning Objectives

After studying this chapter, you should be able to:

- ◆ Distinguish among various types of intruder behavior patterns.
- ◆ Understand the basic principles of and requirements for intrusion detection.
- ◆ Discuss the key features of host-based intrusion detection.
- ◆ Explain the concept of distributed host-based intrusion detection.
- ◆ Discuss the key features of network-based intrusion detection.
- ◆ Define the intrustion detection exchange format.
- ◆ Explain the purpose of honeypots.
- ◆ Present an overview of Snort.

A significant security problem for networked systems is hostile, or at least unwanted, trespass by users or software. User trespass can take the form of unauthorized logon or other access to a machine or, in the case of an authorized user, acquisition of privileges or performance of actions beyond those that have been authorized. Software trespass includes a range of malware variants as we discuss in Chapter 6.

This chapter covers the subject of intrusions. First, we examine the nature of intruders and how they attack, then look at strategies for detecting intrusions.

8.1 INTRUDERS

One of the key threats to security is the use of some form of hacking by an intruder, often referred to as a hacker or cracker. Verizon [VERI16] indicates that 92% of the breaches they investigated were by outsiders, with 14% by insiders, and with some breaches involving both outsiders and insiders. They also noted that insiders were responsible for a small number of very large dataset compromises. Both Symantec [SYMA16] and Verizon [VERI16] also comment that not only is there a general increase in malicious hacking activity, but also an increase in attacks specifically targeted at individuals in organizations and the IT systems they use. This trend emphasizes the need to use defense-in-depth strategies, since such targeted attacks may be designed to bypass perimeter defenses such as firewalls and network-based Intrusion detection systems (IDSs).

As with any defense strategy, an understanding of possible motivations of the attackers can assist in designing a suitable defensive strategy. Again, both Symantec [SYMA16] and Verizon [VERI16] comment on the following broad classes of intruders:

- **Cyber criminals:** Are either individuals or members of an organized crime group with a goal of financial reward. To achieve this, their activities may include identity theft, theft of financial credentials, corporate espionage, data theft, or data ransoming. Typically, they are young, often Eastern European, Russian, or southeast Asian hackers, who do business on the Web [ANTE06]. They meet in underground forums with names such as DarkMarket.org and theftservices.com to trade tips and data and coordinate attacks. For some years, reports such as [SYMA16] have quoted very large and increasing costs resulting from cyber-crime activities, and hence the need to take steps to mitigate this threat.

- **Activists:** Are either individuals working as insiders, or members of a larger group of outsider attackers, who are motivated by social or political causes. They are also known as hacktivists, and their skill level may be quite low. The aim of their attacks is often to promote and publicize their cause, typically through website defacement, denial of service attacks, or the theft and distribution of data that results in negative publicity or compromise of their targets. Well-known recent examples include the activities of the groups Anonymous and LulzSec, and the actions of Chelsea (born Bradley) Manning and Edward Snowden.
- **State-sponsored organizations:** Are groups of hackers sponsored by governments to conduct espionage or sabotage activities. They are also known as Advanced Persistent Threats (APTs), due to the covert nature and persistence over extended periods involved with many attacks in this class. Recent reports such as [MAND13], and information revealed by Edward Snowden, indicate the widespread nature and scope of these activities by a wide range of countries from China and Russia to the USA, UK, and their intelligence allies.
- **Others:** Are hackers with motivations other than those listed above, including classic hackers or crackers who are motivated by technical challenge or by peer-group esteem and reputation. Many of those responsible for discovering new categories of buffer overflow vulnerabilities [MEER10] could be regarded as members of this class. In addition, given the wide availability of attack toolkits, there is a pool of "hobby hackers" using them to explore system and network security, who could potentially become recruits for the above classes.

Across these classes of intruders, there is also a range of skill levels seen. These can be broadly classified as:

- **Apprentice:** Hackers with minimal technical skill who primarily use existing attack toolkits. They likely comprise the largest number of attackers, including many criminal and activist attackers. Given their use of existing known tools, these attackers are the easiest to defend against. They are also known as "script-kiddies" due to their use of existing scripts (tools).
- **Journeyman:** Hackers with sufficient technical skills to modify and extend attack toolkits to use newly discovered, or purchased, vulnerabilities; or to focus on different target groups. They may also be able to locate new vulnerabilities to exploit that are similar to some already known. A number of hackers with such skills are likely found in all intruder classes listed above, adapting tools for use by others. The changes in attack tools make identifying and defending against such attacks harder.
- **Master:** Hackers with high-level technical skills capable of discovering brand new categories of vulnerabilities, or writing new powerful attack toolkits. Some of the better-known classical hackers are of this level, as clearly are some of

those employed by some state-sponsored organizations, as the designation APT suggests. This makes defending against these attackers of the highest difficulty.

Intruder attacks range from the benign to the serious. At the benign end of the scale, there are people who simply wish to explore the Internet and see what is out there. At the serious end are individuals or groups that attempt to read privileged data, perform unauthorized modifications to data, or disrupt systems.

NIST SP 800-61 (*Computer Security Incident Handling Guide*, August 2012) lists the following examples of intrusion:

- Performing a remote root compromise of an e-mail server
- Defacing a Web server
- Guessing and cracking passwords
- Copying a database containing credit card numbers
- Viewing sensitive data, including payroll records and medical information, without authorization
- Running a packet sniffer on a workstation to capture usernames and passwords
- Using a permission error on an anonymous FTP server to distribute pirated software and music files
- Dialing into an unsecured modem and gaining internal network access
- Posing as an executive, calling the help desk, resetting the executive's e-mail password, and learning the new password
- Using an unattended, logged-in workstation without permission

Intrusion detection systems (IDSs) and intrusion prevention systems (IPSs), of the type described in this chapter and Chapter 9 respectively, are designed to aid countering these types of threats. They can be reasonably effective against known, less sophisticated attacks, such as those by activist groups or large-scale e-mail scams. They are likely less effective against the more sophisticated, targeted attacks by some criminal or state-sponsored intruders, since these attackers are more likely to use new, zero-day exploits, and to better obscure their activities on the targeted system. Hence they need to be part of a defense-in-depth strategy that may also include encryption of sensitive information, detailed audit trails, strong authentication and authorization controls, and active management of operating system and application security.

Intruder Behavior

The techniques and behavior patterns of intruders are constantly shifting to exploit newly discovered weaknesses and to evade detection and countermeasures. However, intruders typically use steps from a common attack methodology. [VERI16] in their "Wrap up" section illustrate a typical sequence of actions, starting with a phishing attack that results in the installation of malware that steals login credentials that eventually result in the compromise of a Point-of-Sale terminal. They note that while this is one specific incident scenario, the components are commonly seen in many attacks. [MCCL12] discuss in detail a wider range of activities associated with the following steps:

- **Target Acquisition and Information Gathering:** Where the attacker identifies and characterizes the target systems using publicly available information, both technical and non technical, and the use of network exploration tools to map target resources.

- **Initial Access:** The initial access to a target system, typically by exploiting a remote network vulnerability as we will discuss in Chapters 10 and 11, by guessing weak authentication credentials used in a remote service as we discussed in Chapter 3, or via the installation of malware on the system using some form of social engineering or drive-by-download attack as we discussed in Chapter 6.
- **Privilege Escalation:** Actions taken on the system, typically via a local access vulnerability as we will discuss in Chapters 10 and 11, to increase the privileges available to the attacker to enable their desired goals on the target system.
- **Information Gathering or System Exploit:** Actions by the attacker to access or modify information or resources on the system, or to navigate to another target system.
- **Maintaining Access:** Actions such as the installation of backdoors or other malicious software as we discussed in Chapter 6, or through the addition of covert authentication credentials or other configuration changes to the system, to enable continued access by the attacker after the initial attack.
- **Covering Tracks:** Where the attacker disables or edits audit logs such as we will discuss in Chapter 18, to remove evidence of attack activity, and uses rootkits and other measures to hide covertly installed files or code as we discussed in Chapter 6.

Table 8.1 lists examples of activities associated with the above steps.

Table 8.1 Examples of Intruder Behavior

(a) Target Acquisition and Information Gathering
• Explore corporate website for information on corporate structure, personnel, key systems, as well as details of specific Web server and OS used.
• Gather information on target network using DNS lookup tools such as dig, host, and others; and query WHOIS database.
• Map network for accessible services using tools such as NMAP.
• Send query e-mail to customer service contact, review response for information on mail client, server, and OS used, and also details of person responding.
• Identify potentially vulnerable services, for example, vulnerable Web CMS.

(b) Initial Access
• Brute force (guess) a user's Web content management system (CMS) password.
• Exploit vulnerability in Web CMS plugin to gain system access.
• Send spear-phishing e-mail with link to Web browser exploit to key people.

(c) Privilege Escalation
• Scan system for applications with local exploit.
• Exploit any vulnerable application to gain elevated privileges.
• Install sniffers to capture administrator passwords.
• Use captured administrator password to access privileged information.

(Continued)

Table 8.1 *(Continued)*

(d) Information Gathering or System Exploit

- Scan files for desired information.
- Transfer large numbers of documents to external repository.
- Use guessed or captured passwords to access other servers on network.

(e) Maintaining Access

- Install remote administration tool or rootkit with backdoor for later access.
- Use administrator password to later access network.
- Modify or disable anti-virus or IDS programs running on system.

(f) Covering Tracks

- Use rootkit to hide files installed on system.
- Edit logfiles to remove entries generated during the intrusion.

8.2 INTRUSION DETECTION

The following terms are relevant to our discussion:

security intrusion: Unauthorized act of bypassing the security mechanisms of a system.

intrusion detection: A hardware or software function that gathers and analyzes information from various areas within a computer or a network to identify possible security intrusions.

An IDS comprises three logical components:

- **Sensors:** Sensors are responsible for collecting data. The input for a sensor may be any part of a system that could contain evidence of an intrusion. Types of input to a sensor includes network packets, log files, and system call traces. Sensors collect and forward this information to the analyzer.
- **Analyzers:** Analyzers receive input from one or more sensors or from other analyzers. The analyzer is responsible for determining if an intrusion has occurred. The output of this component is an indication that an intrusion has occurred. The output may include evidence supporting the conclusion that an intrusion occurred. The analyzer may provide guidance about what actions to take as a result of the intrusion. The sensor inputs may also be stored for future analysis and review in a storage or database component.

- **User interface:** The user interface to an IDS enables a user to view output from the system or control the behavior of the system. In some systems, the user interface may equate to a manager, director, or console component.

An IDS may use a single sensor and analyzer, such as a classic HIDS on a host or NIDS in a firewall device. More sophisticated IDSs can use multiple sensors, across a range of host and network devices, sending information to a centralized analyzer and user interface in a distributed architecture.

IDSs are often classified based on the source and type of data analyzed, as:

- **Host-based IDS (HIDS)**: Monitors the characteristics of a single host and the events occurring within that host, such as process identifiers and the system calls they make, for evidence of suspicious activity.
- **Network-based IDS (NIDS)**: Monitors network traffic for particular network segments or devices and analyzes network, transport, and application protocols to identify suspicious activity.
- **Distributed or hybrid IDS:** Combines information from a number of sensors, often both host and network-based, in a central analyzer that is able to better identify and respond to intrusion activity.

Basic Principles

Authentication facilities, access control facilities, and firewalls all play a role in countering intrusions. Another line of defense is intrusion detection, and this has been the focus of much research in recent years. This interest is motivated by a number of considerations, including the following:

1. If an intrusion is detected quickly enough, the intruder can be identified and ejected from the system before any damage is done or any data are compromised. Even if the detection is not sufficiently timely to preempt the intruder, the sooner that the intrusion is detected, the less the amount of damage and the more quickly that recovery can be achieved.
2. An effective IDS can serve as a deterrent, thus acting to prevent intrusions.
3. Intrusion detection enables the collection of information about intrusion techniques that can be used to strengthen intrusion prevention measures.

Intrusion detection is based on the assumption that the behavior of the intruder differs from that of a legitimate user in ways that can be quantified. Of course, we cannot expect that there will be a crisp, exact distinction between an attack by an intruder and the normal use of resources by an authorized user. Rather, we must expect that there will be some overlap.

Figure 8.1 suggests, in abstract terms, the nature of the task confronting the designer of an IDS. Although the typical behavior of an intruder differs from the typical behavior of an authorized user, there is an overlap in these behaviors. Thus, a loose interpretation of intruder behavior, which will catch more intruders, will also lead to a number of **false positives**, or false alarms, where authorized users are identified as intruders. On the other hand, an attempt to limit false positives by a tight interpretation of intruder behavior will lead to an increase in **false negatives**, or intruders not

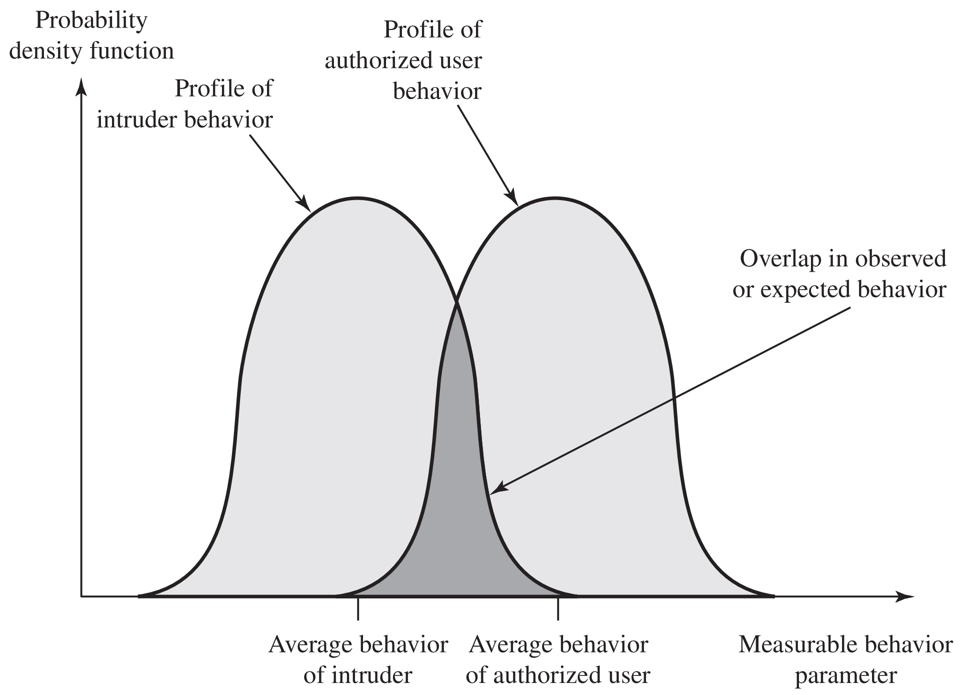


Figure 8.1 Profiles of Behavior of Intruders and Authorized Users

identified as intruders. Thus, there is an element of compromise and art in the practice of intrusion detection. Ideally, you want an IDS to have a high detection rate, that is, the ratio of detected to total attacks, while minimizing the false alarm rate, the ratio of incorrectly classified to total normal usage [LAZA05].

In an important early study of intrusion [ANDE80], Anderson postulated that one could, with reasonable confidence, distinguish between an outside attacker and a legitimate user. Patterns of legitimate user behavior can be established by observing past history, and significant deviation from such patterns can be detected. Anderson suggests the task of detecting an inside attacker (a legitimate user acting in an unauthorized fashion) is more difficult, in that the distinction between abnormal and normal behavior may be small. Anderson concluded that such violations would be undetectable solely through the search for anomalous behavior. However, insider behavior might nevertheless be detectable by intelligent definition of the class of conditions that suggest unauthorized use. These observations, which were made in 1980, remain true today.

The Base-Rate Fallacy

To be of practical use, an IDS should detect a substantial percentage of intrusions while keeping the false alarm rate at an acceptable level. If only a modest percentage of actual intrusions are detected, the system provides a false sense of security. On the other hand, if the system frequently triggers an alert when there is no intrusion (a false alarm), then either system managers will begin to ignore the alarms, or much time will be wasted analyzing the false alarms.

Unfortunately, because of the nature of the probabilities involved, it is very difficult to meet the standard of high rate of detections with a low rate of false alarms.

In general, if the actual numbers of intrusions is low compared to the number of legitimate uses of a system, then the false alarm rate will be high unless the test is extremely discriminating. This is an example of a phenomenon known as the *base-rate fallacy*. A study of existing IDSs, reported in [AXEL00], indicated that current systems have not overcome the problem of the base-rate fallacy. See Appendix I for a brief background on the mathematics of this problem.

Requirements

[BALA98] lists the following as desirable for an IDS. It must:

- Run continually with minimal human supervision.
- Be fault tolerant in the sense that it must be able to recover from system crashes and reinitializations.
- Resist subversion. The IDS must be able to monitor itself and detect if it has been modified by an attacker.
- Impose a minimal overhead on the system where it is running.
- Be able to be configured according to the security policies of the system that is being monitored.
- Be able to adapt to changes in system and user behavior over time.
- Be able to scale to monitor a large number of hosts.
- Provide graceful degradation of service in the sense that if some components of the IDS stop working for any reason, the rest of them should be affected as little as possible.
- Allow dynamic reconfiguration; that is, the ability to reconfigure the IDS without having to restart it.

8.3 ANALYSIS APPROACHES

IDSs typically use one of the following alternative approaches to analyze sensor data to detect intrusions:

1. **Anomaly detection**: Involves the collection of data relating to the behavior of legitimate users over a period of time. Then, current observed behavior is analyzed to determine with a high level of confidence whether this behavior is that of a legitimate user or alternatively that of an intruder.
2. **Signature or Heuristic detection**: Uses a set of known malicious data patterns (signatures) or attack rules (heuristics) that are compared with current behavior to decide if it is that of an intruder. It is also known as misuse detection. This approach can only identify known attacks for which it has patterns or rules.

In essence, anomaly approaches aim to define normal, or expected, behavior, in order to identify malicious or unauthorized behavior. Signature or heuristic-based approaches directly define malicious or unauthorized behavior. They can quickly and efficiently identify known attacks. However, only anomaly detection is able to detect unknown, zero-day attacks, as it starts with known good behavior and identifies anomalies to it. Given this advantage, clearly anomaly detection would be the preferred approach, were it not for the difficulty in collecting and analyzing the data required, and the high level of false alarms, as we will discuss in the following sections.

Anomaly Detection

The anomaly detection approach involves first developing a model of legitimate user behavior by collecting and processing sensor data from the normal operation of the monitored system in a training phase. This may occur at distinct times, or there may be a continuous process of monitoring and evolving the model over time. Once this model exists, current observed behavior is compared with the model in order to classify it as either legitimate or anomalous activity in a detection phase.

A variety of classification approaches are used, which [GARC09] broadly categorized as:

- **Statistical:** Analysis of the observed behavior using univariate, multivariate, or time-series models of observed metrics.
- **Knowledge based:** Approaches use an expert system that classifies observed behavior according to a set of rules that model legitimate behavior.
- **Machine-learning:** Approaches automatically determine a suitable classification model from the training data using data mining techniques.

They also note two key issues that affect the relative performance of these alternatives, being the efficiency and cost of the detection process.

The monitored data is first parameterized into desired standard metrics that will then be analyzed. This step ensures that data gathered from a variety of possible sources is provided in standard form for analysis.

Statistical approaches use the captured sensor data to develop a statistical profile of the observed metrics. The earliest approaches used univariate models, where each metric was treated as an independent random variable. However, this was too crude to effectively identify intruder behavior. Later, multivariate models considered correlations between the metrics, with better levels of discrimination observed. Time-series models use the order and time between observed events to better classify the behavior. The advantages of these statistical approaches include their relative simplicity and low computation cost, and lack of assumptions about behavior expected. Their disadvantages include the difficulty in selecting suitable metrics to obtain a reasonable balance between false positives and false negatives, and that not all behaviors can be modeled using these approaches.

Knowledge-based approaches classify the observed data using a set of rules. These rules are developed during the training phase, usually manually, to characterize the observed training data into distinct classes. Formal tools may be used to describe these rules, such as a finite-state machine or a standard description language. They are then used to classify the observed data in the detection phase. The advantages of knowledge-based approaches include their robustness and flexibility. Their main disadvantage is the difficulty and time required to develop high-quality knowledge from the data, and the need for human experts to assist with this process.

Machine-learning approaches use data mining techniques to automatically develop a model using the labeled normal training data. This model is then able to classify subsequently observed data as either normal or anomalous. A key disadvantage is that this process typically requires significant time and computational resources. Once the model is generated however, subsequent analysis is generally fairly efficient.

A variety of machine-learning approaches have been tried, with varying success. These include:

- **Bayesian networks:** Encode probabilistic relationships among observed metrics.
- **Markov models:** Develop a model with sets of states, some possibly hidden, interconnected by transition probabilities.
- **Neural networks:** Simulate human brain operation with neurons and synapse between them, that classify observed data.
- **Fuzzy logic:** Uses fuzzy set theory where reasoning is approximate, and can accommodate uncertainty.
- **Genetic algorithms:** Uses techniques inspired by evolutionary biology, including inheritance, mutation, selection and recombination, to develop classification rules.
- **Clustering and outlier detection:** Group the observed data into clusters based on some similarity or distance measure, and then identify subsequent data as either belonging to a cluster or as an outlier.

The advantages of the machine-learning approaches include their flexibility, adaptability, and ability to capture interdependencies between the observed metrics. Their disadvantages include their dependency on assumptions about accepted behavior for a system, their currently unacceptably high false alarm rate, and their high resource cost.

A key limitation of anomaly detection approaches used by IDSs, particularly the machine-learning approaches, is that they are generally only trained with legitimate data, unlike many of the other applications surveyed in [CHAN09] where both legitimate and anomalous training data is used. The lack of anomalous training data, which occurs given the desire to detect currently unknown future attacks, limits the effectiveness of some of the techniques listed above.

Signature or Heuristic Detection

Signature or heuristic techniques detect intrusion by observing events in the system and applying either a set of signature patterns to the data, or a set of rules that characterize the data, leading to a decision regarding whether the observed data indicates normal or anomalous behavior.

Signature approaches match a large collection of known patterns of malicious data against data stored on a system or in transit over a network. The signatures need to be large enough to minimize the false alarm rate, while still detecting a sufficiently large fraction of malicious data. This approach is widely used in anti virus products, in network traffic scanning proxies, and in NIDS. The advantages of this approach include the relatively low cost in time and resource use, and its wide acceptance. Disadvantages include the significant effort required to constantly identify and review new malware to create signatures able to identify it, and the inability to detect zero-day attacks for which no signatures exist.

Rule-based heuristic identification involves the use of rules for identifying known penetrations or penetrations that would exploit known weaknesses. Rules can also be defined that identify suspicious behavior, even when the behavior is within the bounds of established patterns of usage. Typically, the rules used in these systems are specific to the machine and operating system. The most fruitful approach to developing such rules is to analyze attack tools and scripts collected on the Internet. These rules can be supplemented with rules generated by knowledgeable security personnel. In this latter case, the normal procedure is to interview system administrators and security analysts to collect a suite of known penetration scenarios and key events that threaten the security of the target system.

The SNORT system, which we will discuss later in Section 8.9, is an example of a rule-based NIDS. A large collection of rules exists for it to detect a wide variety of network attacks.

8.4 HOST-BASED INTRUSION DETECTION

Host-based IDSs (HIDSs) add a specialized layer of security software to vulnerable or sensitive systems; such as database servers and administrative systems. The HIDS monitors activity on the system in a variety of ways to detect suspicious behavior. In some cases, an IDS can halt an attack before any damage is done, as we will discuss in Section 9.6, but its main purpose is to detect intrusions, log suspicious events, and send alerts.

The primary benefit of a HIDS is that it can detect both external and internal intrusions, something that is not possible either with network-based IDSs or firewalls. As we discussed in the previous section, host-based IDSs can use either anomaly or signature and heuristic approaches to detect unauthorized behavior on the monitored host. We now review some common data sources and sensors used in HIDS, continue with a discussion of how the anomaly, signature and heuristic approaches are used in HIDS, then consider distributed HIDS.

Data Sources and Sensors

As noted previously, a fundamental component of intrusion detection is the sensor that collects data. Some record of ongoing activity by users must be provided as input to the analysis component of the IDS. Common data sources include:

- **System call traces:** A record of the sequence of systems calls by processes on a system, is widely acknowledged as the preferred data source for HIDS since the pioneering work of Forrest [CREE13]. While these work well on Unix and Linux systems, they are problematic on Windows systems due to the extensive use of DLLs that obscure which processes use specific system calls.
- **Audit (log file) records**[1]**:** Most modern operating systems include accounting software that collects information on user activity. The advantage of using this information is that no additional collection software is needed.

[1] Audit records play a more general role in computer security than just intrusion detection. See Chapter 18 for a full discussion.

The disadvantages are that the audit records may not contain the needed information or may not contain it in a convenient form, and that intruders may attempt to manipulate these records to hide their actions.

- **File integrity checksums:** A common approach to detecting intruder activity on a system is to periodically scan critical files for changes from the desired baseline, by comparing a current cryptographic checksums for these files, with a record of known good values. Disadvantages include the need to generate and protect the checksums using known good files, and the difficulty monitoring changing files. Tripwire is a well-known system using this approach.
- **Registry access:** An approach used on Windows systems is to monitor access to the registry, given the amount of information and access to it used by programs on these systems. However, this source is very Windows specific, and has recorded limited success.

The sensor gathers data from the chosen source, filters the gathered data to remove any unwanted information and to standardize the information format, and forwards the result to the IDS analyzer, which may be local or remote.

Anomaly HIDS

The majority of work on anomaly-based HIDS has been done on UNIX and Linux systems, given the ease of gathering suitable data for this work. While some earlier work used audit or accounting records, the majority is based on system call traces. System calls are the means by which programs access core kernel functions, providing a wide range of interactions with the low-level operating system functions. Hence they provide detailed information on process activity that can be used to classify it as normal or anomalous. Table 8.2a lists the system calls used in current Ubuntu Linux systems as an example. This data is typically gathered using an OS hook, such as the BSM audit module. Most modern operating systems have highly reliable options for collecting this type of information.

The system call traces are then analyzed by a suitable decision engine. [CREE13] notes that the original work by Forrest et al. introduced the Sequence Time-Delay Embedding (STIDE) algorithm, based on artificial immune system approaches, that compares observed sequences of system calls with sequences from the training phase to obtain a mismatch ratio that determines whether the sequence is normal or not. Later work has used alternatives, such as Hidden Markov Models (HMM), Artificial Neural Networks (ANN), Support Vector Machines (SVM), or Extreme Learning Machines (ELM) to make this classification.

[CREE13] notes that these approaches all report providing reasonable intruder detection rates of 95–99% while having false positive rates of less than 5%, though on older test datasets. He updates these results using recent contemporary data and example attacks, with a more extensive feature extraction process from the system call traces and an ELM decision engine capable of a very high detection rate while maintaining reasonable false positive rates. This approach should lead to even more effective production HIDS products in the near future.

Windows systems have traditionally not used anomaly-based HIDS, as the wide usage of Dynamic Link Libraries (DLLs) as an intermediary between process requests for operating system functions and the actual system call interface has

Table 8.2 Linux System Calls and Windows DLLs Monitored

(a) Ubuntu Linux System Calls

accept, access, acct, adjtime, aiocancel, aioread, aiowait, aiowrite, alarm, async_daemon, auditsys, bind, chdir, chmod, chown, chroot, close, connect, creat, dup, dup2, execv, execve, exit, exportfs, fchdir, fchmod, fchown, fchroot, fcntl, flock, fork, fpathconf, fstat, fstat, fstatfs, fsync, ftime, ftruncate, getdents, getdirentries, getdomainname, getdopt, getdtablesize, getfh, getgid, getgroups, gethostid, gethostname, getitimer, getmsg, getpagesize, getpeername, getpgrp, getpid, getpriority, getrlimit, getrusage, getsockname, getsockopt, gettimeofday, getuid, gtty, ioctl, kill, killpg, link, listen, lseek, lstat, madvise, mctl, mincore, mkdir, mknod, mmap, mount, mount, mprotect, mpxchan, msgsys, msync, munmap, nfs_mount, nfssvc, nice, open, pathconf, pause, pcfs_mount, phys, pipe, poll, profil, ptrace, putmsg, quota, quotactl, read, readlink, readv, reboot, recv, recvfrom, recvmsg, rename, resuba, rfssys, rmdir, sbreak, sbrk, select, semsys, send, sendmsg, sendto, setdomainname, setdopt, setgid, setgroups, sethostid, sethostname, setitimer, setpgid, setpgrp, setpgrp, setpriority, setquota, setregid, setreuid, setrlimit, setsid, setsockopt, settimeofday, setuid, shmsys, shutdown, sigblock, sigpause, sigpending, sigsetmask, sigstack, sigsys, sigvec, socket, socketaddr, socketpair, sstk, stat, stat, statfs, stime, stty, swapon, symlink, sync, sysconf, time, times, truncate, umask, umount, uname, unlink, unmount, ustat, utime, utimes, vadvise, vfork, vhangup, vlimit, vpixsys, vread, vtimes, vtrace, vwrite, wait, wait3, wait4, write, writev

(b) Key Windows DLLs and Executables

comctl32
kernel32
msvcpp
msvcrt
mswsock
ntdll
ntoskrnl
user32
ws2_32

hindered the effective use of system call traces to classify process behavior. Some work was done using either audit log entries, or registry file updates as a data source, but neither approach was very successful. [CREE13] reports a new approach that uses traces of key DLL function calls as an alternative data source, with results comparable to that found with Linux system call trace HIDS. Table 8.2b lists the key DLLs and executables monitored. Note that all of the distinct functions within these DLLs, numbering in their thousands, are monitored, forming the equivalent to the system call list presented in Table 8.2a. The adoption of this approach should lead to the development of more effective Windows HIDS, capable of detecting zero-day attacks, unlike the current generation of signature and heuristic Windows HIDS that we will discuss later.

While using system call traces provides arguably the richest information source for a HIDS, it does impose a moderate load on the monitored system to gather and classify this data. And as we noted earlier, the training phase for many of the decision engines requires very significant time and computational resources. Hence, others have trialed approaches based on audit (log) records. However, these both have a lower detection rate than the system call trace approaches (80% reported), and are more susceptible to intruder manipulation.

A further alternative to examining current process behavior is to look for changes to important files on the monitored host. This uses a cryptographic checksum to check for any changes from the known good baseline for the monitored files. Typically, all program binaries, scripts, and configuration files are monitored, either on each access, or on a periodic scan of the file system. The tripwire system is a widely used implementation of this approach, and is available for all major operating systems including Linux, Mac OS, and Windows. This approach is very sensitive to changes in the monitored files, as a result of intruder activity or for any other reason. However, it cannot detect changes made to processes once they are running on the system. Other difficulties include determining which files to monitor, since a surprising number of files change in an operational system, having access to a known good copy of each monitored file to establish the baseline value, and protecting the database of file signatures.

Signature or Heuristic HIDS

The alternative of signature or heuristic-based HIDS is widely used, particularly as seen in anti virus (A/V), more correctly viewed as anti malware, products. These are very commonly used on client systems and increasingly on mobile devices, and also incorporated into mail and Web application proxies on firewalls and in network-based IDSs. They use either a database of file signatures, which are patterns of data found in known malicious software, or heuristic rules that characterize known malicious behavior.

These products are quite efficient at detecting known malware, however they are not capable of detecting zero-day attacks that do not correspond to the known signatures or heuristic rules. They are widely used, particularly on Windows systems, which continue to be targeted by intruders, as we discussed in Section 6.9.

Distributed HIDS

Traditionally, work on host-based IDSs focused on single-system stand-alone operation. The typical organization, however, needs to defend a distributed collection of hosts supported by a LAN or internetwork. Although it is possible to mount a defense by using stand-alone IDSs on each host, a more effective defense can be achieved by coordination and cooperation among IDSs across the network.

Porras points out the following major issues in the design of a distributed IDS [PORR92]:

- A distributed IDS may need to deal with different sensor data formats. In a heterogeneous environment, different systems may use different sensors and approaches to gathering data for intrusion detection use.
- One or more nodes in the network will serve as collection and analysis points for the data from the systems on the network. Thus, either raw sensor data or summary data must be transmitted across the network. Therefore, there is a requirement to assure the integrity and confidentiality of these data. Integrity is required to prevent an intruder from masking his or her activities by altering the transmitted audit information. Confidentiality is required because the transmitted audit information could be valuable.

- Either a centralized or decentralized architecture can be used. With a centralized architecture, there is a single central point of collection and analysis of all sensor data. This eases the task of correlating incoming reports but creates a potential bottleneck and single point of failure. With a decentralized architecture, there is more than one analysis center, but these must coordinate their activities and exchange information.

A good example of a distributed IDS is one developed at the University of California at Davis [HEBE92, SNAP91]; a similar approach has been taken for a project at Purdue University [SPAF00, BALA98]. Figure 8.2 shows the overall architecture, which consists of three main components:

1. **Host agent module:** An audit collection module operating as a background process on a monitored system. Its purpose is to collect data on security-related events on the host and transmit these to the central manager. Figure 8.3 shows details of the agent module architecture.
2. **LAN monitor agent module:** Operates in the same fashion as a host agent module except that it analyzes LAN traffic and reports the results to the central manager.
3. **Central manager module:** Receives reports from LAN monitor and host agents and processes and correlates these reports to detect intrusion.

The scheme is designed to be independent of any operating system or system auditing implementation. Figure 8.3 shows the general approach that is taken. The agent captures each audit record produced by the native audit collection system. A filter is applied that retains only those records that are of security interest. These records are then reformatted into a standardized format referred to as the host

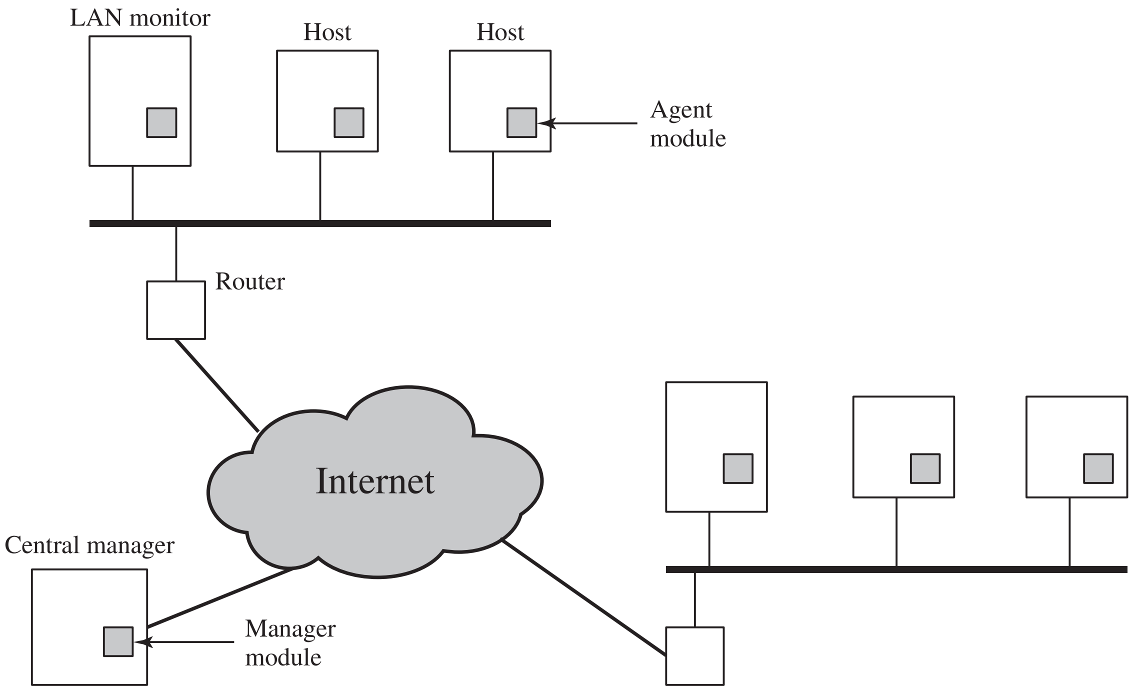


Figure 8.2 **Architecture for Distributed Intrusion Detection**

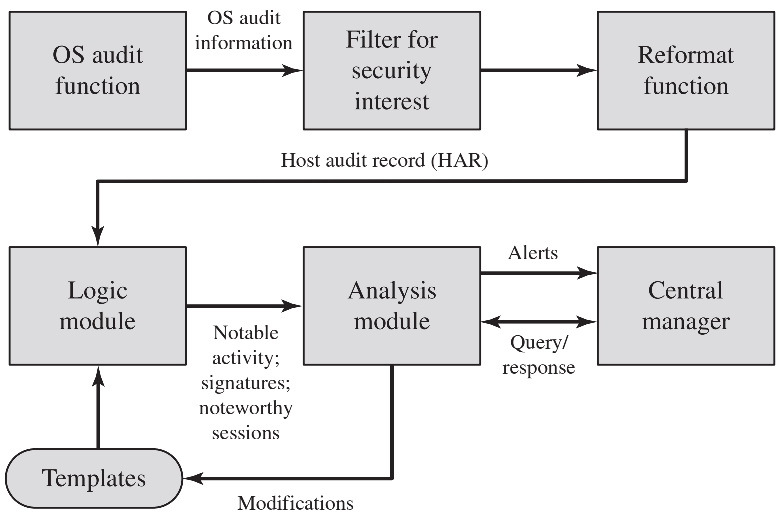


Figure 8.3 Agent Architecture

audit record (HAR). Next, a template-driven logic module analyzes the records for suspicious activity. At the lowest level, the agent scans for notable events that are of interest independent of any past events. Examples include failed files, accessing system files, and changing a file's access control. At the next higher level, the agent looks for sequences of events, such as known attack patterns (signatures). Finally, the agent looks for anomalous behavior of an individual user based on a historical profile of that user, such as number of programs executed, number of files accessed, and the like.

When suspicious activity is detected, an alert is sent to the central manager. The central manager includes an expert system that can draw inferences from received data. The manager may also query individual systems for copies of HARs to correlate with those from other agents.

The LAN monitor agent also supplies information to the central manager. The LAN monitor agent audits host-host connections, services used, and volume of traffic. It searches for significant events, such as sudden changes in network load, the use of security-related services, and suspicious network activities.

The architecture depicted in Figures 8.2 and 8.3 is quite general and flexible. It offers a foundation for a machine-independent approach that can expand from stand-alone intrusion detection to a system that is able to correlate activity from a number of sites and networks to detect suspicious activity that would otherwise remain undetected.

8.5 NETWORK-BASED INTRUSION DETECTION

A network-based IDS (NIDS) monitors traffic at selected points on a network or interconnected set of networks. The NIDS examines the traffic packet by packet in real time, or close to real time, to attempt to detect intrusion patterns. The NIDS may examine network-, transport-, and/or application-level protocol activity. Note the contrast with a host-based IDS; a NIDS examines packet traffic directed toward potentially vulnerable computer systems on a network. A host-based system examines user and software activity on a host.

NIDS are typically included in the perimeter security infrastructure of an organization, either incorporated into, or associated with, the firewall. They typically focus on monitoring for external intrusion attempts, by analyzing both traffic patterns and traffic content for malicious activity. With the increasing use of encryption though, NIDS have lost access to significant content, hindering their ability to function well. Thus, while they have an important role to play, they can only form part of the solution. A typical NIDS facility includes a number of sensors to monitor packet traffic, one or more servers for NIDS management functions, and one or more management consoles for the human interface. The analysis of traffic patterns to detect intrusions may be done at the sensor, at the management server, or some combination of the two.

Types of Network Sensors

Sensors can be deployed in one of two modes: inline and passive. An **inline sensor** is inserted into a network segment so the traffic that it is monitoring must pass through the sensor. One way to achieve an inline sensor is to combine NIDS sensor logic with another network device, such as a firewall or a LAN switch. This approach has the advantage that no additional separate hardware devices are needed; all that is required is NIDS sensor software. An alternative is a stand-alone inline NIDS sensor. The primary motivation for the use of inline sensors is to enable them to block an attack when one is detected. In this case, the device is performing both intrusion detection and intrusion prevention functions.

More commonly, **passive sensors** are used. A passive sensor monitors a copy of network traffic; the actual traffic does not pass through the device. From the point of view of traffic flow, the passive sensor is more efficient than the inline sensor, because it does not add an extra handling step that contributes to packet delay.

Figure 8.4 illustrates a typical passive sensor configuration. The sensor connects to the network transmission medium, such as a fiber optic cable, by a direct physical tap. The tap provides the sensor with a copy of all network traffic being carried by the medium. The network interface card (NIC) for this tap usually does not have an IP address configured for it. All traffic into this NIC is simply collected with no protocol interaction with the network. The sensor has a second NIC that connects to the network with an IP address and enables the sensor to communicate with a NIDS management server.

Another distinction is whether the sensor is monitoring a wired or wireless network. A wireless network sensor may either be inline, incorporated into a wireless access point (AP), or a passive wireless traffic monitor. Only these sensors can gather and analyze wireless protocol traffic, and hence detect attacks against those protocols. Such attacks include wireless denial-of-service, session hijack, or AP impersonation. A NIDS focussed exclusively on a wireless network is known as a Wireless IDS (WIDS). Alternatively, wireless sensors may be a component of a more general NIDS gathering data from both wired and wireless network traffic, or even of a distributed IDS combining host and network sensor data.

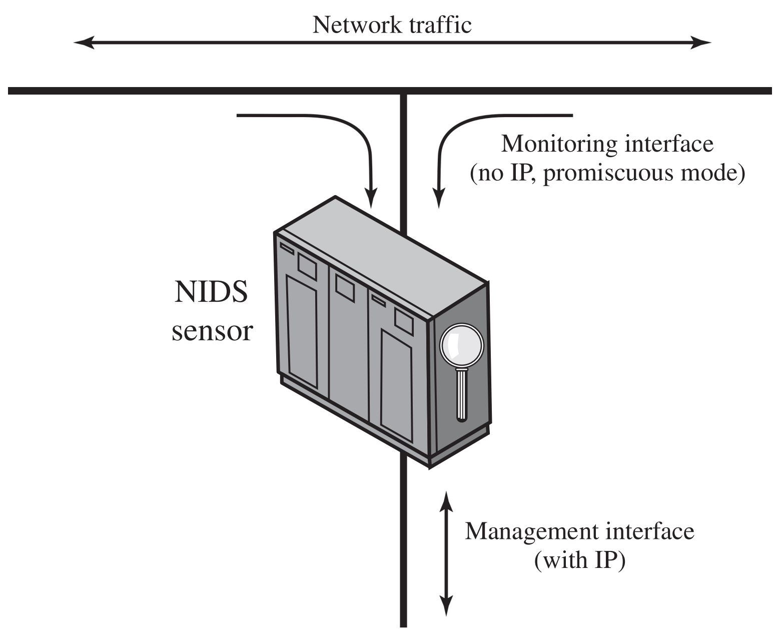


Figure 8.4 Passive NIDS Sensor
Source: Based on [CREM06].

NIDS Sensor Deployment

Consider an organization with multiple sites, each of which has one or more LANs, with all of the networks interconnected via the Internet or some other WAN technology. For a comprehensive NIDS strategy, one or more sensors are needed at each site. Within a single site, a key decision for the security administrator is the placement of the sensors.

Figure 8.5 illustrates a number of possibilities. In general terms, this configuration is typical of larger organizations. All Internet traffic passes through an external firewall that protects the entire facility.[2] Traffic from the outside world, such as customers and vendors that need access to public services, such as Web and mail, is monitored. The external firewall also provides a degree of protection for those parts of the network that should only be accessible by users from other corporate sites. Internal firewalls may also be used to provide more specific protection to certain parts of the network.

A common location for a NIDS sensor is just inside the external firewall (location 1 in the figure). This position has a number of advantages:

- Sees attacks, originating from the outside world, that penetrate the network's perimeter defenses (external firewall).
- Highlights problems with the network firewall policy or performance.
- Sees attacks that might target the Web server or ftp server.
- Even if the incoming attack is not recognized, the IDS can sometimes recognize the outgoing traffic that results from the compromised server.

[2]Firewalls will be discussed in detail in Chapter 9. In essence, a firewall is designed to protect one or a connected set of networks on the inside of the firewall from Internet and other traffic from outside the firewall. The firewall does this by restricting traffic, rejecting potentially threatening packets.

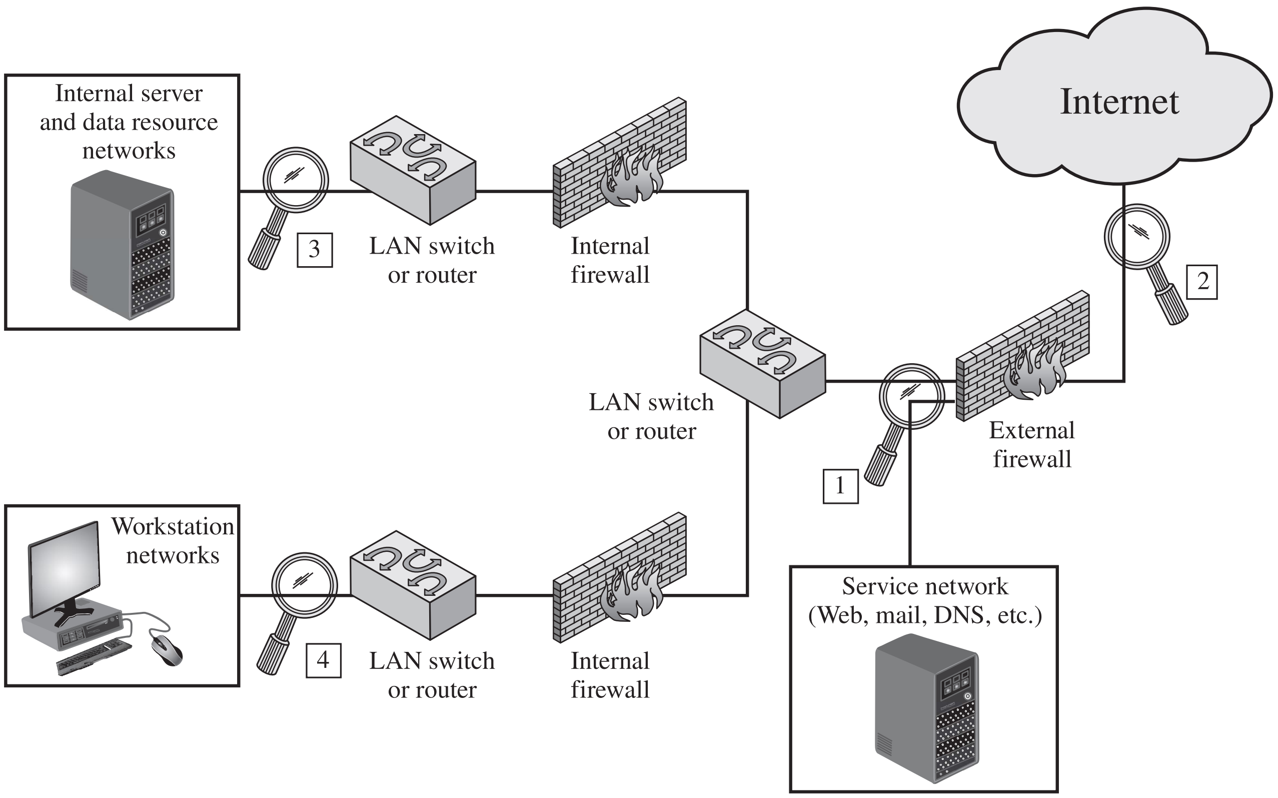


Figure 8.5 **Example of NIDS Sensor Deployment**

Instead of placing a NIDS sensor inside the external firewall, the security administrator may choose to place a NIDS sensor between the external firewall and the Internet or WAN (location 2). In this position, the sensor can monitor all network traffic, unfiltered. The advantages of this approach are as follows:

- Documents number of attacks originating on the Internet that target the network.
- Documents types of attacks originating on the Internet that target the network.

A sensor at location 2 has a higher processing burden than any sensor located elsewhere on the site network.

In addition to a sensor at the boundary of the network, on either side of the external firewall, the administrator may configure a firewall and one or more sensors to protect major backbone networks, such as those that support internal servers and database resources (location 3). The benefits of this placement include the following:

- Monitors a large amount of a network's traffic, thus increasing the possibility of spotting attacks.
- Detects unauthorized activity by authorized users within the organization's security perimeter.

Thus, a sensor at location 3 is able to monitor for both internal and external attacks. Because the sensor monitors traffic to only a subset of devices at the site, it can be tuned to specific protocols and attack types, thus reducing the processing burden.

Finally, the network facilities at a site may include separate LANs that support user workstations and servers specific to a single department. The administrator could configure a firewall and NIDS sensor to provide additional protection for all of these networks or target the protection to critical subsystems, such as personnel and financial networks (location 4). A sensor used in this latter fashion provides the following benefits:

- Detects attacks targeting critical systems and resources.
- Allows focusing of limited resources to the network assets considered of greatest value.

As with a sensor at location 3, a sensor at location 4 can be tuned to specific protocols and attack types, thus reducing the processing burden.

Intrusion Detection Techniques

As with host-based intrusion detection, network-based intrusion detection makes use of signature detection and anomaly detection. Unlike the case with HIDS, a number of commercial anomaly NIDS products are available [GARC09]. One of the best known is the Statistical Packet Anomaly Detection Engine (SPADE), available as a plug-in for the Snort system that we will discuss later.

SIGNATURE DETECTION NIST SP 800-94 (*Guide to Intrusion Detection and Prevention Systems*, July 2012) lists the following as examples of that types of attacks that are suitable for signature detection:

- **Application layer reconnaissance and attacks:** Most NIDS technologies analyze several dozen application protocols. Commonly analyzed ones include Dynamic Host Configuration Protocol (DHCP), DNS, Finger, FTP, HTTP, Internet Message Access Protocol (IMAP), Internet Relay Chat (IRC), Network File System (NFS), Post Office Protocol (POP), rlogin/rsh, Remote Procedure Call (RPC), Session Initiation Protocol (SIP), Server Message Block (SMB), SMTP, SNMP, Telnet, and Trivial File Transfer Protocol (TFTP), as well as database protocols, instant messaging applications, and peer-to-peer file sharing software. The NIDS is looking for attack patterns that have been identified as targeting these protocols. Examples of attack include buffer overflows, password guessing, and malware transmission.
- **Transport layer reconnaissance and attacks:** NIDSs analyze TCP and UDP traffic and perhaps other transport layer protocols. Examples of attacks are unusual packet fragmentation, scans for vulnerable ports, and TCP-specific attacks such as SYN floods.
- **Network layer reconnaissance and attacks:** NIDSs typically analyze IPv4, IPv6, ICMP, and IGMP at this level. Examples of attacks are spoofed IP addresses and illegal IP header values.
- **Unexpected application services:** The NIDS attempts to determine if the activity on a transport connection is consistent with the expected application protocol. An example is a host running an unauthorized application service.
- **Policy violations:** Examples include use of inappropriate websites and use of forbidden application protocols.

Anomaly Detection Techniques NIST SP 800-94 lists the following as examples of the types of attacks that are suitable for anomaly detection:

- **Denial-of-service (DoS) attacks:** Such attacks involve either significantly increased packet traffic or significantly increase connection attempts, in an attempt to overwhelm the target system. These attacks are analyzed in Chapter 7. Anomaly detection is well-suited to such attacks.
- **Scanning:** A scanning attack occurs when an attacker probes a target network or system by sending different kinds of packets. Using the responses received from the target, the attacker can learn many of the system's characteristics and vulnerabilities. Thus, a scanning attack acts as a target identification tool for an attacker. Scanning can be detected by atypical flow patterns at the application layer (e.g., banner grabbing[3]), transport layer (e.g., TCP and UDP port scanning), and network layer (e.g., ICMP scanning).
- **Worms:** Worms[4] spreading among hosts can be detected in more than one way. Some worms propagate quickly and use large amounts of bandwidth. Worms can also be detected because they can cause hosts to communicate with each other that typically do not, and they can also cause hosts to use ports that they normally do not use. Many worms also perform scanning. Chapter 6 discusses worms in detail.

Stateful Protocol Analysis (SPA) NIST SP 800-94 details this subset of anomaly detection that compares observed network traffic against predetermined universal vendor supplied profiles of benign protocol traffic. This distinguishes it from anomaly techniques trained with organization specific traffic profiles. SPA understands and tracks network, transport, and application protocol states to ensure they progress as expected. A key disadvantage of SPA is the high resource use it requires.

Logging of Alerts

When a sensor detects a potential violation, it sends an alert and logs information related to the event. The NIDS analysis module can use this information to refine intrusion detection parameters and algorithms. The security administrator can use this information to design prevention techniques. Typical information logged by a NIDS sensor includes the following:

- Timestamp (usually date and time)
- Connection or session ID (typically a consecutive or unique number assigned to each TCP connection or to like groups of packets for connectionless protocols)
- Event or alert type

[3]Typically, banner grabbing consists of initiating a connection to a network server and recording the data that is returned at the beginning of the session. This information can specify the name of the application, version number, and even the operating system that is running the server [DAMR03].

[4]A worm is a program that can replicate itself and send copies from computer to computer across network connections. Upon arrival, the worm may be activated to replicate and propagate again. In addition to propagation, the worm usually performs some unwanted function.

- Rating (e.g., priority, severity, impact, confidence)
- Network, transport, and application layer protocols
- Source and destination IP addresses
- Source and destination TCP or UDP ports, or ICMP types and codes
- Number of bytes transmitted over the connection
- Decoded payload data, such as application requests and responses
- State-related information (e.g., authenticated username)

8.6 DISTRIBUTED OR HYBRID INTRUSION DETECTION

In recent years, the concept of communicating IDSs has evolved to schemes that involve distributed systems that cooperate to identify intrusions and to adapt to changing attack profiles. These combine in a central IDS, the complementary information sources used by HIDS with host-based process and data details, and NIDS with network events and data, to manage and coordinate intrusion detection and response in an organization's IT infrastructure. Two key problems have always confronted systems such as IDSs, firewalls, virus and worm detectors, and so on. First, these tools may not recognize new threats or radical modifications of existing threats. And second, it is difficult to update schemes rapidly enough to deal with quickly spreading attacks. A separate problem for perimeter defenses, such as firewalls, is that the modern enterprise has loosely defined boundaries, and hosts are generally able to move in and out. Examples are hosts that communicate using wireless technology and employee laptops that can be plugged into network ports.

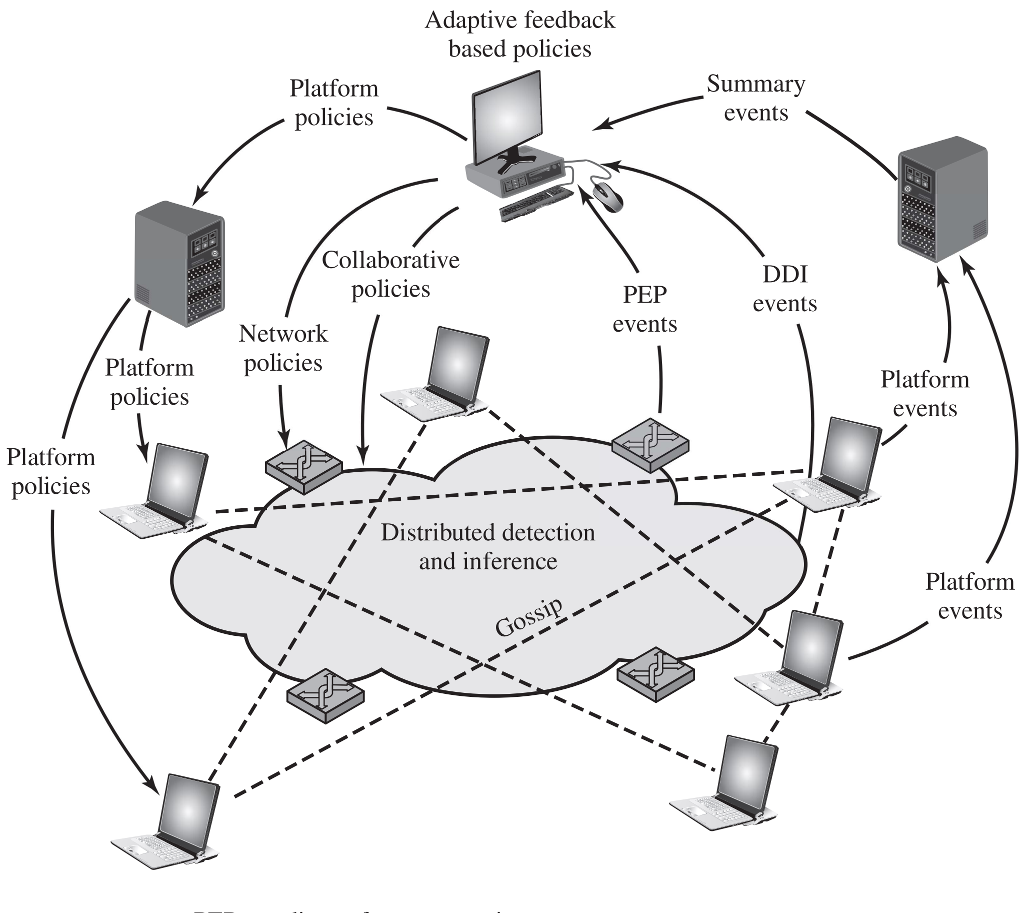
Attackers have exploited these problems in several ways. The more traditional attack approach is to develop worms and other malicious software that spreads ever more rapidly and to develop other attacks (such as DoS attacks) that strike with overwhelming force before a defense can be mounted. This style of attack is still prevalent. But more recently, attackers have added a quite different approach: Slow the spread of the attack so it will be more difficult to detect by conventional algorithms [ANTH07].

A way to counter such attacks is to develop cooperated systems that can recognize attacks based on more subtle clues then adapt quickly. In this approach, anomaly detectors at local nodes look for evidence of unusual activity. For example, a machine that normally makes just a few network connections might suspect that an attack is under way if it is suddenly instructed to make connections at a higher rate. With only this evidence, the local system risks a false positive if it reacts to the suspected attack (say by disconnecting from the network and issuing an alert) but it risks a false negative if it ignores the attack or waits for further evidence. In an adaptive, cooperative system, the local node instead uses a peer-to-peer "gossip" protocol to inform other machines of its suspicion, in the form of a probability that the network is under attack. If a machine receives enough of these messages so a threshold is exceeded, the machine assumes an attack is under way and responds. The machine may respond locally to defend itself and also send an alert to a central system.

An example of this approach is a scheme developed by Intel and referred to as autonomic enterprise security [AGOS06]. Figure 8.6 illustrates the approach. This approach does not rely solely on perimeter defense mechanisms, such as firewalls, or on individual host-based defenses. Instead, each end host and each network device (e.g., routers) is considered to be a potential sensor and may have the sensor software module installed. The sensors in this distributed configuration can exchange information to corroborate the state of the network (i.e., whether an attack is under way).

The Intel designers provide the following motivation for this approach:

1. IDSs deployed selectively may miss a network-based attack or may be slow to recognize that an attack is under way. The use of multiple IDSs that share information has been shown to provide greater coverage and more rapid response to attacks, especially slowly growing attacks (e.g., [BAIL05], [RAJA05]).
2. Analysis of network traffic at the host level provides an environment in which there is much less network traffic than found at a network device such as a router. Thus, attack patterns will stand out more, providing in effect a higher signal-to-noise ratio.
3. Host-based detectors can make use of a richer set of data, possibly using application data from the host as input into the local classifier.



PEP = policy enforcement point
DDI = distributed detection and inference

Figure 8.6 Overall Architecture of an Autonomic Enterprise Security System

NIST SP 800-94 notes that a distributed or hybrid IDS can be constructed using multiple products from a single vendor, designed to share and exchange data. This is clearly an easier, but may not be the most cost-effective or comprehensive solution. Alternatively, specialized security information and event management (SIEM) software exists that can import and analyze data from a variety of sources, sensors, and products. Such software may well rely on standardized protocols, such as Intrusion Detection Exchange Format we will discuss in the next section. An analogy may help clarify the advantage of this distributed approach. Suppose a single host is subject to a prolonged attack, and the host is configured to minimize false positives. Early on in the attack, no alert is sounded because the risk of false positive is high. If the attack persists, the evidence that an attack is under way becomes stronger and the risk of false positive decreases. However, much time has passed. Now, consider many local sensors, each of which suspect the onset of an attack and all of which collaborate. Because numerous systems see the same evidence, an alert can be issued with a low false positive risk. Thus, instead of a long period of time, we use a large number of sensors to reduce false positives and still detect attacks. A number of vendors now offer this type of product.

We now summarize the principal elements of this approach, illustrated in Figure 8.6. A central system is configured with a default set of security policies. Based on input from distributed sensors, these policies are adapted and specific actions are communicated to the various platforms in the distributed system. The device-specific policies may include immediate actions to take or parameter settings to be adjusted. The central system also communicates collaborative policies to all platforms that adjust the timing and content of collaborative gossip messages. Three types of input guide the actions of the central system:

- **Summary events:** Events from various sources are collected by intermediate collection points such as firewalls, IDSs, or servers that serve a specific segment of the enterprise network. These events are summarized for delivery to the central policy system.
- **DDI events:** Distributed detection and inference (DDI) events are alerts that are generated when the gossip traffic enables a platform to conclude that an attack is under way.
- **PEP events:** Policy enforcement points (PEPs) reside on trusted, self-defending platforms and intelligent IDSs. These systems correlate distributed information, local decisions, and individual device actions to detect intrusions that may not be evident at the host level.

8.7 INTRUSION DETECTION EXCHANGE FORMAT

To facilitate the development of distributed IDSs that can function across a wide range of platforms and environments, standards are needed to support interoperability. Such standards are the focus of the IETF Intrusion Detection Working Group. The purpose of the working group is to define data formats and exchange procedures for sharing information of interest to intrusion detection and response systems and to