CHAPTER 6

Intrusion Detection

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 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 can take the form of a virus, worm, or Trojan horse.

This chapter covers the subject of intruders. We discuss other forms of attack in subsequent chapters. First, we examine the nature of the intrusion attack and then look at strategies detecting intrusions.

6.1 INTRUDERS

One of the two most publicized threats to security is the intruder (the other is viruses), generally referred to as a hacker or cracker. In an important early study of intrusion, Anderson [ANDE80] identified three classes of intruders:

- **Masquerader:** An individual who is not authorized to use the computer and who penetrates a system's access controls to exploit a legitimate user's account
- **Misfeasor:** A legitimate user who accesses data, programs, or resources for which such access is not authorized, or who is authorized for such access but misuses his or her privileges
- **Clandestine user:** An individual who seizes supervisory control of the system and uses this control to evade auditing and access controls or to suppress audit collection

The masquerader is likely to be an outsider; the misfeasor generally is an insider; and the clandestine user can be either an outsider or an insider.

Intruder attacks range from the benign to the serious. At the benign end of the scale, there are many people who simply wish to explore internets and see what is out there. At the serious end are individuals who are attempting to read privileged data, perform unauthorized modifications to data, or disrupt the system.

[GRAN04] 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

Intruder Behavior Patterns

The techniques and behavior patterns of intruders are constantly shifting, to exploit newly discovered weaknesses and to evade detection and countermeasures. Even so, intruders typically follow one of a number of recognizable behavior patterns, and these patterns typically differ from those of ordinary users. In the following, we look at three broad examples of intruder behavior patterns, to give the reader some feel for the challenge facing the security administrator. Table 6.1, based on [RADC04], summarizes the behavior.

Hackers Traditionally, those who hack into computers do so for the thrill of it or for status. The hacking community is a strong meritocracy in which status is determined by level of competence. Thus, attackers often look for targets of opportunity and then share the information with others. A typical example is a break-in at a large financial institution reported in [RADC04]. The intruder took advantage of the fact

Table 6.1 Some Examples of Intruder Patterns of Behavior

(a) Hacker

1.	Select the target using IP lookup tools such as NSLookup, Dig, and others.
2.	Map network for accessible services using tools such as NMAP.
3.	Identify potentially vulnerable services (in this case, pcAnywhere).
4.	Brute force (guess) pcAnywhere password.
5.	Install remote administration tool called DameWare.
6.	Wait for administrator to log on and capture his password.
7.	Use that password to access remainder of network.

(b) Criminal Enterprise

1.	Act quickly and precisely to make their activities harder to detect.
2.	Exploit perimeter through vulnerable ports.
3.	Use Trojan horses (hidden software) to leave back doors for reentry.
4.	Use sniffers to capture passwords.
5.	Do not stick around until noticed.
6.	Make few or no mistakes.

(c) Internal Threat

1.	Create network accounts for themselves and their friends.
2.	Access accounts and applications they wouldn't normally use for their daily jobs.
3.	E-mail former and prospective employers.
4.	Conduct furtive instant-messaging chats.
5.	Visit Web sites that cater to disgruntled employees, such as f'dcompany.com.
6.	Perform large downloads and file copying.
7.	Access the network during off hours.

that the corporate network was running unprotected services, some of which were not even needed. In this case, the key to the break-in was the pcAnywhere application. The manufacturer, Symantec, advertises this program as a remote control solution that enables secure connection to remote devices. But the attacker had an easy time gaining access to pcAnywhere; the administrator used the same three-letter username and password for the program. In this case, there was no intrusion detection system on the 700-node corporate network. The intruder was only discovered when a vice president walked into her office and saw the cursor moving files around on her Windows workstation.

Benign intruders might be tolerable, although they do consume resources and may slow performance for legitimate users. However, there is no way in advance to know whether an intruder will be benign or malign. Consequently, even for systems with no particularly sensitive resources, there is a motivation to control this problem.

Intrusion detection systems (IDSs) and intrusion prevention systems (IPSs), of the type described in this chapter and Chapter 9, respectively, are designed to counter this type of hacker threat. In addition to using such systems, organizations can consider restricting remote logons to specific IP addresses and/or use virtual private network technology.

One of the results of the growing awareness of the intruder problem has been the establishment of a number of computer emergency response teams (CERTs). These cooperative ventures collect information about system vulnerabilities and disseminate it to systems managers. Hackers also routinely CERT reports. Thus, it is important for system administrators to quickly insert all software patches to discovered vulnerabilities. Unfortunately, given the complexity of many IT systems, and the rate at which patches are released, this is increasingly difficult to achieve without automated updating. Even then, there are problems caused by incompatibilities resulting from the updated software. Hence the need for multiple layers of defense in managing security threats to IT systems.

Criminals Organized groups of hackers have become a widespread and common threat to Internet-based systems. These groups can be in the employ of a corporation or government but often are loosely affiliated gangs of hackers. Typically, these gangs are young, often Eastern European, Russian, or southeast Asian hackers who do business on the Web [ANTE06]. They meet in underground forums with names like DarkMarket.org and theftservices.com to trade tips and data and coordinate attacks. A common target is a credit card file at an e-commerce server. Attackers attempt to gain root access. The card numbers are used by organized crime gangs to purchase expensive items and are then posted to carder sites, where others can access and use the account numbers; this obscures usage patterns and complicates investigation.

Whereas traditional hackers look for targets of opportunity, criminal hackers usually have specific targets, or at least classes of targets in mind. Once a site is penetrated, the attacker acts quickly, scooping up as much valuable information as possible and exiting.

IDSs and IPSs can also be used for these types of attackers but may be less effective because of the quick in-and-out nature of the attack. For e-commerce sites, database encryption should be used for sensitive customer information, especially credit cards. For hosted e-commerce sites (provided by an outsider service), the e-commerce organization should make use of a dedicated server (not used to support multiple customers) and closely monitor the provider's security services.

Insider Attacks Insider attacks are among the most difficult to detect and prevent. Employees already have access and knowledge about the structure and content of corporate databases. Insider attacks can be motivated by revenge of simply a feeling of entitlement. An example of the former is the case of Kenneth Patterson, fired from his position as data communications manager for American Eagle Outfitters. Patterson disabled the company's ability to process credit card purchases during five days of the holiday season of 2002. As for a sense of entitlement, there have always been many employees who felt entitled to take extra office supplies for home use, but this now extends to corporate data. An example is that of a vice president of sales for a stock analysis firm who quit to go to a competitor. Before she left, she copied the customer database to take with her. The offender reported feeling no animus toward her former employee; she simply wanted the data because it would be useful to her.

Although IDS and IPS facilities can be useful in countering insider attacks, other more direct approaches are of higher priority. Examples include the following:

- Enforce least privilege, only allowing access to the resources employees need to do their job.
- Set logs to see what users access and what commands they are entering.
- Protect sensitive resources with strong authentication.
- Upon termination, delete employee's computer and network access.
- Upon termination, make a mirror image of employee's hard drive before reissuing it. That evidence might be needed if your company information turns up at a competitor.

In this section, we look at the techniques used for intrusion. Then we examine ways to detect intrusion.

Intrusion Techniques

The objective of the intruder is to gain access to a system or to increase the range of privileges accessible on a system. Most initial attacks use system or software vulnerabilities that allow a user to execute code that opens a back door into the system. Intruders can get access to a system by exploiting attacks such as buffer overflows on a program that runs with certain privileges. We examine such software vulnerabilities in Part Two.

Alternatively, the intruder attempts to acquire information that should have been protected. In some cases, this information is in the form of a user password. With knowledge of some other user's password, an intruder can log in to a system and exercise all the privileges accorded to the legitimate user. Password guessing and password acquisition techniques are discussed in Chapter 3.

6.2 INTRUSION DETECTION

The following definitions from RFC 2828 (Internet Security Glossary) are relevant to our discussion:

Security Intrusion: A security event, or a combination of multiple security events, that constitutes a security incident in which an intruder gains, or attempts to gain, access to a system (or system resource) without having authorization to do so.

Intrusion Detection: A security service that monitors and analyzes system events for the purpose of finding, and providing real-time or near real-time warning of, attempts to access system resources in an unauthorized manner.

IDSs can be classified as follows:

- **Host-based IDS:** Monitors the characteristics of a single host and the events occurring within that host for suspicious activity
- **Network-based IDS:** Monitors network traffic for particular network segments or devices and analyzes network, transport, and application protocols to identify suspicious activity

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.
- **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.

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 6.1 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 authorized users 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 identified as intruders. Thus, there is an element of compromise and art in the practice of intrusion detection.

In Anderson's study [ANDE80], it was postulated that one could, with reasonable confidence, distinguish between a masquerader 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 that the task of detecting a misfeasor (legitimate user performing 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, misfeasor behavior might nevertheless be detectable by intelligent definition of the class of conditions that

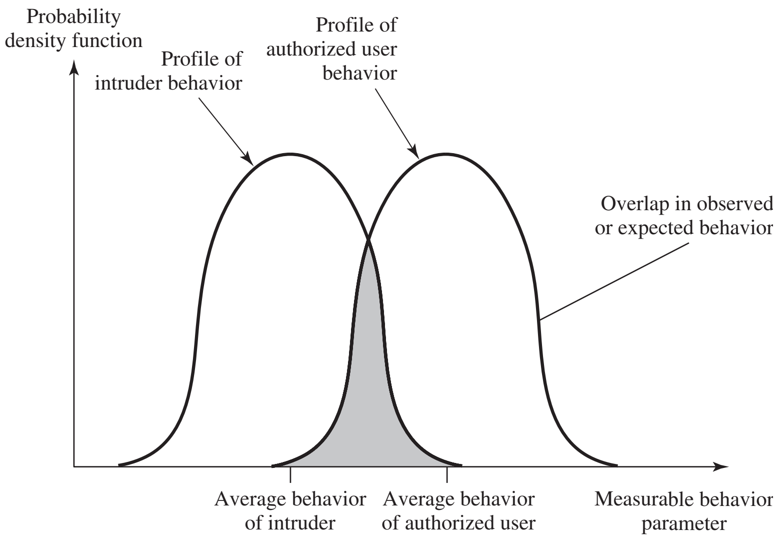

Probability density function
Profile of intruder behavior
Profile of authorized user behavior
Overlap in observed or expected behavior
Average behavior of intruder
Average behavior of authorized user
Measurable behavior parameter

Figure 6.1 **Profiles of Behavior of Intruders and Authorized Users**

suggest unauthorized use. Finally, the detection of the clandestine user was felt to be beyond the scope of purely automated techniques. These observations, which were made in 1980, remain true today.

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.

6.3 HOST-BASED INTRUSION DETECTION

Host-based IDSs add a specialized layer of security software to vulnerable or sensitive systems; examples include database servers and administrative systems. The host-based IDS 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, but its primary purpose is to detect intrusions, log suspicious events, and send alerts.

The primary benefit of a host-based IDS is that it can detect both external and internal intrusions, something that is not possible either with network-based IDSs or firewalls.

Host-based IDSs follow one of two general approaches to intrusion detection:

1. **Anomaly detection:** Involves the collection of data relating to the behavior of legitimate users over a period of time. Then statistical tests are applied to observed behavior to determine with a high level of confidence whether that behavior is not legitimate user behavior. The following are two approaches to statistical anomaly detection:
   a. **Threshold detection:** This approach involves defining thresholds, independent of user, for the frequency of occurrence of various events.
   b. **Profile based:** A profile of the activity of each user is developed and used to detect changes in the behavior of individual accounts.

2. **Signature detection:** Involves an attempt to define a set of rules or attack patterns that can be used to decide that a given behavior is that of an intruder.

In essence, anomaly approaches attempt to define normal, or expected, behavior, whereas signature-based approaches attempt to define proper behavior.

In terms of the types of attackers listed earlier, anomaly detection is effective against masqueraders, who are unlikely to mimic the behavior patterns of the accounts they appropriate. On the other hand, such techniques may be unable to deal with misfeasors. For such attacks, signature-based approaches may be able to recognize events and sequences that, in context, reveal penetration. In practice, a system may employ a combination of both approaches to be effective against a broad range of attacks.

Audit Records

A fundamental tool for intrusion detection is the audit record.[1] Some record of ongoing activity by users must be maintained as input to an IDS. Basically, two plans are used:

- **Native audit records:** Virtually all multiuser 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. The disadvantage is that the native audit records may not contain the needed information or may not contain it in a convenient form.
- **Detection-specific audit records:** A collection facility can be implemented that generates audit records containing only that information required by the IDS. One advantage of such an approach is that it could be made vendor independent and ported to a variety of systems. The disadvantage is the extra overhead involved in having, in effect, two accounting packages running on a machine.

A good example of detection-specific audit records is one developed by Dorothy Denning [DENN87]. Each audit record contains the following fields:

- **Subject:** Initiators of actions. A subject is typically a terminal user but might also be a process acting on behalf of users or groups of users. All activity arises through commands issued by subjects. Subjects may be grouped into different access classes, and these classes may overlap.
- **Action:** Operation performed by the subject on or with an object; for example, login, read, perform I/O, execute.
- **Object:** Receptors of actions. Examples include files, programs, messages, records, terminals, printers, and user- or program-created structures. When a subject is the recipient of an action, such as electronic mail, then that subject is considered an object. Objects may be grouped by type. Object granularity may vary by object type and by environment. For example, database actions may be audited for the database as a whole or at the record level.
- **Exception-Condition:** Denotes which, if any, exception condition is raised on return.

[1]Audit records play a more general role in computer security than just intrusion detection. See Chapter 15 for a full discussion.

- **Resource-Usage:** A list of quantitative elements in which each element gives the amount used of some resource (e.g., number of lines printed or displayed, number of records read or written, processor time, I/O units used, session elapsed time).
- **Time-Stamp:** Unique time-and-date stamp identifying when the action took place.

Most user operations are made up of a number of elementary actions. For example, a file copy involves the execution of the user command, which includes doing access validation and setting up the copy, plus the read from one file, plus the write to another file. Consider the command

```
COPY GAME.EXE TO <Library>GAME.EXE
```

issued by Smith to copy an executable file GAME from the current directory to the directory. The following audit records may be generated:

Smith	execute	<Library>COPY.EXE	0	CPU = 00002	11058721678
Smith	read	<Smith>GAME.EXE	0	RECORDS = 0	11058721679
Smith	execute	<Library>COPY.EXE	write-viol	RECORDS = 0	11058721680

In this case, the copy is aborted because Smith does not have write permission to <Library> .

The decomposition of a user operation into elementary actions has three advantages:

1. Because objects are the protectable entities in a system, the use of elementary actions enables an audit of all behavior affecting an object. Thus, the system can detect attempted subversions of access controls (by noting an abnormality in the number of exception conditions returned) and can detect successful subversions by noting an abnormality in the set of objects accessible to the subject.
2. Single-object, single-action audit records simplify the model and the implementation.
3. Because of the simple, uniform structure of the detection-specific audit records, it may be relatively easy to obtain this information or at least part of it by a straightforward mapping from existing native audit records to the detection-specific audit records.

Anomaly Detection

As was mentioned, anomaly detection techniques fall into two broad categories: threshold detection and profile-based systems. Threshold detection involves counting the number of occurrences of a specific event type over an interval of time. If the count surpasses what is considered a reasonable number that one might expect to occur, then intrusion is assumed.

Threshold analysis, by itself, is a crude and ineffective detector of even moderately sophisticated attacks. Both the threshold and the time interval must be determined. Because of the variability across users, such thresholds are likely to generate either a lot of false positives or a lot of false negatives. However, simple threshold detectors may be useful in conjunction with more sophisticated techniques.

Profile-based anomaly detection focuses on characterizing the past behavior of individual users or related groups of users and then detecting significant deviations. A profile may consist of a set of parameters, so that deviation on just a single parameter may not be sufficient in itself to signal an alert.

The foundation of this approach is an analysis of audit records. The audit records provide input to the intrusion detection function in two ways. First, the designer must decide on a number of quantitative metrics that can be used to measure user behavior. An analysis of audit records over a period of time can be used to determine the activity profile of the average user. Thus, the audit records serve to define typical behavior. Second, current audit records are the input used to detect intrusion. That is, the intrusion detection model analyzes incoming audit records to determine deviation from average behavior.

Examples of metrics that are useful for profile-based intrusion detection are the following:

- **Counter:** A nonnegative integer that may be incremented but not decremented until it is reset by management action. Typically, a count of certain event types is kept over a particular period of time. Examples include the number of logins by a single user during an hour, the number of times a given command is executed during a single user session, and the number of password failures during a minute.
- **Gauge:** A nonnegative integer that may be incremented or decremented. Typically, a gauge is used to measure the current value of some entity. Examples include the number of logical connections assigned to a user application and the number of outgoing messages queued for a user process.
- **Interval timer:** The length of time between two related events. An example is the length of time between successive logins to an account.
- **Resource utilization:** Quantity of resources consumed during a specified period. Examples include the number of pages printed during a user session and total time consumed by a program execution.

Given these general metrics, various tests can be performed to determine whether current activity fits within acceptable limits. [DENN87] lists the following approaches that may be taken:

- Mean and standard deviation
- Multivariate
- Markov process
- Time series
- Operational

The simplest statistical test is to measure the **mean and standard deviation** of a parameter over some historical period. This gives a reflection of the average behavior and its variability. The use of mean and standard deviation is applicable to a wide variety of counters, timers, and resource measures. But these measures, by themselves, are typically too crude for intrusion detection purposes.

A **multivariate** model is based on correlations between two or more variables. Intruder behavior may be characterized with greater confidence by considering such correlations (for example, processor time and resource usage, or login frequency and session elapsed time).

A **Markov process** model is used to establish transition probabilities among various states. As an example, this model might be used to look at transitions between certain commands.

A **time series** model focuses on time intervals, looking for sequences of events that happen too rapidly or too slowly. A variety of statistical tests can be applied to characterize abnormal timing.

Finally, an **operational model** is based on a judgment of what is considered abnormal, rather than an automated analysis of past audit records. Typically, fixed limits are defined and intrusion is suspected for an observation that is outside the limits. This type of approach works best where intruder behavior can be deduced from certain types of activities. For example, a large number of login attempts over a short period suggests an attempted intrusion.

As an example of the use of these various metrics and models, Table 6.2 shows various measures used for the Stanford Research Institute (SRI) IDS (IDES) [ANDE95, JAVI91] and the follow-on program Emerald [NEUM99].

The main advantage of the use of statistical profiles is that a prior knowledge of security flaws is not required. The detector program learns what is "normal" behavior and then looks for deviations. The approach is not based on system-dependent characteristics and vulnerabilities. Thus, it should be readily portable among a variety of systems.

Signature Detection

Signature techniques detect intrusion by observing events in the system and applying a set of rules that lead to a decision regarding whether a given pattern of activity is or is not suspicious. In very general terms, we can characterize all approaches as focusing on either anomaly detection or penetration identification, although there is some overlap in these approaches.

Rule-based anomaly detection is similar in terms of its approach and strengths to statistical anomaly detection. With the rule-based approach, historical audit records are analyzed to identify usage patterns and to generate automatically rules that describe those patterns. Rules may represent past behavior patterns of users, programs, privileges, time slots, terminals, and so on. Current behavior is then observed, and each transaction is matched against the set of rules to determine if it conforms to any historically observed pattern of behavior.

As with statistical anomaly detection, rule-based anomaly detection does not require knowledge of security vulnerabilities within the system. Rather, the scheme is based on observing past behavior and, in effect, assuming that the future will be

Table 6.2 Measures That May Be Used for Intrusion Detection

Measure	Model	Type of Intrusion Detected
Login and Session Activity		
Login frequency by day and time	Mean and standard deviation	Intruders may be likely to log in during off hours.
Frequency of login at different locations	Mean and standard deviation	Intruders may log in from a location that a particular user rarely or never uses.
Time since last login	Operational	Break-in on a "dead" account.
Elapsed time per session	Mean and standard deviation	Significant deviations might indicate masquerader.
Quantity of output to location	Mean and standard deviation	Excessive amounts of data transmitted to remote locations could signify leakage of sensitive data.
Session resource utilization	Mean and standard deviation	Unusual processor or I/O levels could signal an intruder.
Password failures at login	Operational	Attempted break-in by password guessing.
Failures to login from specified terminals	Operational	Attempted break-in.
Command or Program Execution Activity		
Execution frequency	Mean and standard deviation	May detect intruders, who are likely to use different commands, or a successful penetration by a legitimate user, who has gained access to privileged commands.
Program resource utilization	Mean and standard deviation	An abnormal value might suggest injection of a virus or Trojan horse, which performs side effects that increase I/O or processor utilization.
Execution denials	Operational model	May detect penetration attempt by individual user who seeks higher privileges.
Fil Access Activity		
Read, write, create, delete frequency	Mean and standard deviation	Abnormalities for read and write access for individual users may signify masquerading or browsing.
Records read, written	Mean and standard deviation	Abnormality could signify an attempt to obtain sensitive data by inference and aggregation.
Failure count for read, write, create, delete	Operational	May detect users who persistently attempt to access unauthorized files.

like the past. In order for this approach to be effective, a rather large database of rules will be needed. For example, a scheme described in [VACC89] contains anywhere from 10^4 to 10^6 rules.

Rule-based penetration identification takes a very different approach to intrusion detection. The key feature of such systems is 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.

A simple example of the type of rules that can be used is found in NIDX, an early system that used heuristic rules that can be used to assign degrees of suspicion to activities [BAUE88]. Example heuristics are the following:

1. Users should not read files in other users' personal directories.
2. Users must not write other users' files.
3. Users who log in after hours often access the same files they used earlier.
4. Users do not generally open disk devices directly but rely on higher-level operating system utilities.
5. Users should not be logged in more than once to the same system.
6. Users do not make copies of system programs.

The penetration identification scheme used in IDES is representative of the strategy followed. Audit records are examined as they are generated, and they are matched against the rule base. If a match is found, then the user's *suspicion rating* is increased. If enough rules are matched, then the rating will pass a threshold that results in the reporting of an anomaly.

The IDES approach is based on an examination of audit records. A weakness of this plan is its lack of flexibility. For a given penetration scenario, there may be a number of alternative audit record sequences that could be produced, each varying from the others slightly or in subtle ways. It may be difficult to pin down all these variations in explicit rules. Another method is to develop a higher-level model independent of specific audit records. An example of this is a state transition model known as USTAT [VIGN02, ILGU95]. USTAT deals in general actions rather than the detailed specific actions recorded by the UNIX auditing mechanism. USTAT is implemented on a SunOS system that provides audit records on 239 events. Of these, only 28 are used by a preprocessor, which maps these onto 10 general actions (Table 6.3). Using just these actions and the parameters that are invoked with each action, a state transition diagram is developed that characterizes suspicious activity. Because a number of different auditable events map into a smaller number of actions, the rule-creation process is simpler. Furthermore, the state transition diagram model is easily modified to accommodate newly learned intrusion behaviors.

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

Table 6.3 USTAT Actions versus SunOS Event Types

USTAT Action	SunOS Event Type
Read	open_r, open_rc, open_rtc, open_rwc, open rwtc, open_rt, open_rw, open_rwt
Write	truncate, ftruncate, creat, open_rtc, open_rwc, open_rwtc, open_rt, open_rw, open_rwt, open_w, open_wt, open_wc, open_wct
Create	mkdir, creat, open_rc, open_rtc, open_rwc, open_rwtc, open_wc, open_wtc, mknod
Delete	rmdir, unlink
Execute	exec, execve
Exit	exit
Modify_Owner	chown, fchown
Modify_Perm	chmod, fchmod
Rename	rename
Hardlink	link

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 6A for a brief background on the mathematics of this problem.

6.4 DISTRIBUTED HOST-BASED INTRUSION DETECTION

Traditionally, work on host-based IDSs focused on single-system stand-alone facilities. 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 audit record formats. In a heterogeneous environment, different systems will employ different native audit collection systems and, if using intrusion detection, may employ different formats for security-related audit records.
- 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 audit 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 audit 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 [SPAF00, BALA98]. Figure 6.2 shows the overall architecture, which consists of three main components:

- **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.
- **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.
- **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 6.3 [SNAP91] shows the general approach that is taken. The agent captures each audit record produced by the native audit collection

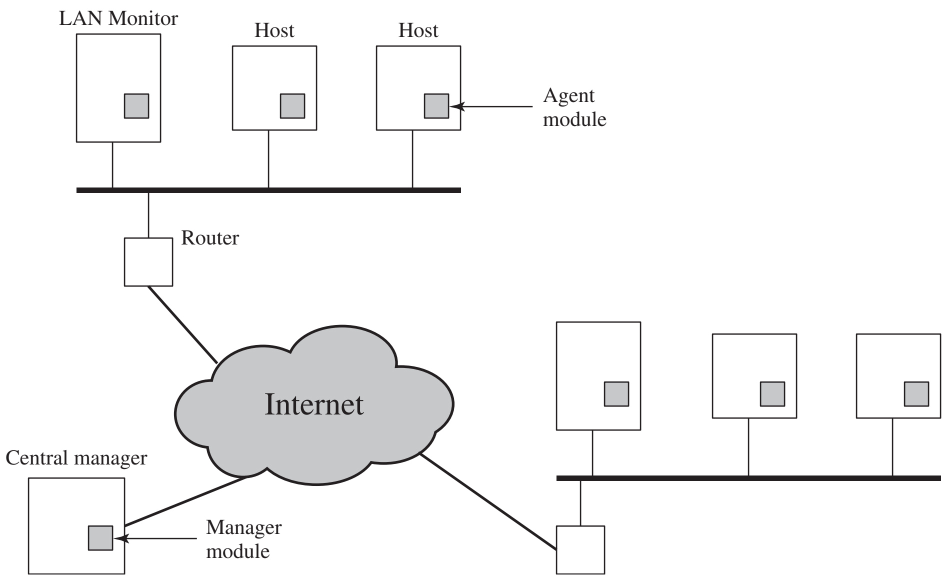


Figure 6.2 Architecture for Distributed Intrusion Detection

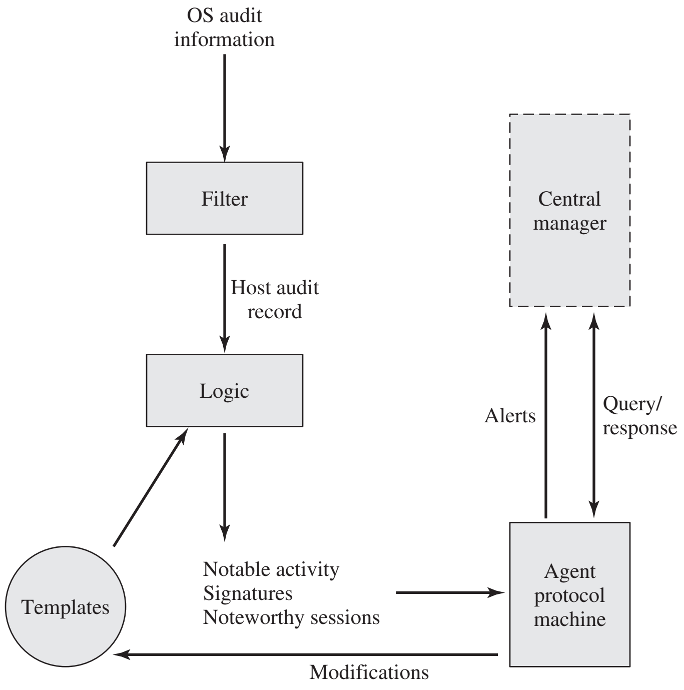


Figure 6.3 Agent Architecture

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 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 network activities such as *rlogin*.

The architecture depicted in Figures 6.2 and 6.3 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.

6.5 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.

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 that 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 6.4 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.

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 6.5 illustrates a number of possibilities. In general terms, this configuration is typical of larger organizations. All Internet traffic passes through an external

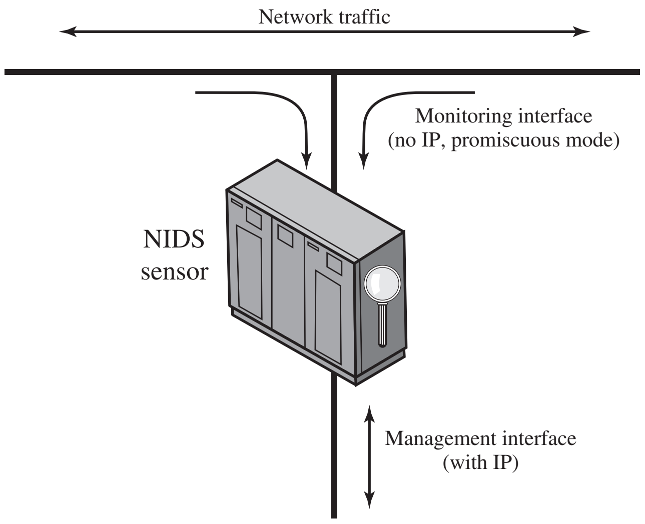


Figure 6.4 Passive NIDS Sensor
Source: Based on [CREM06].

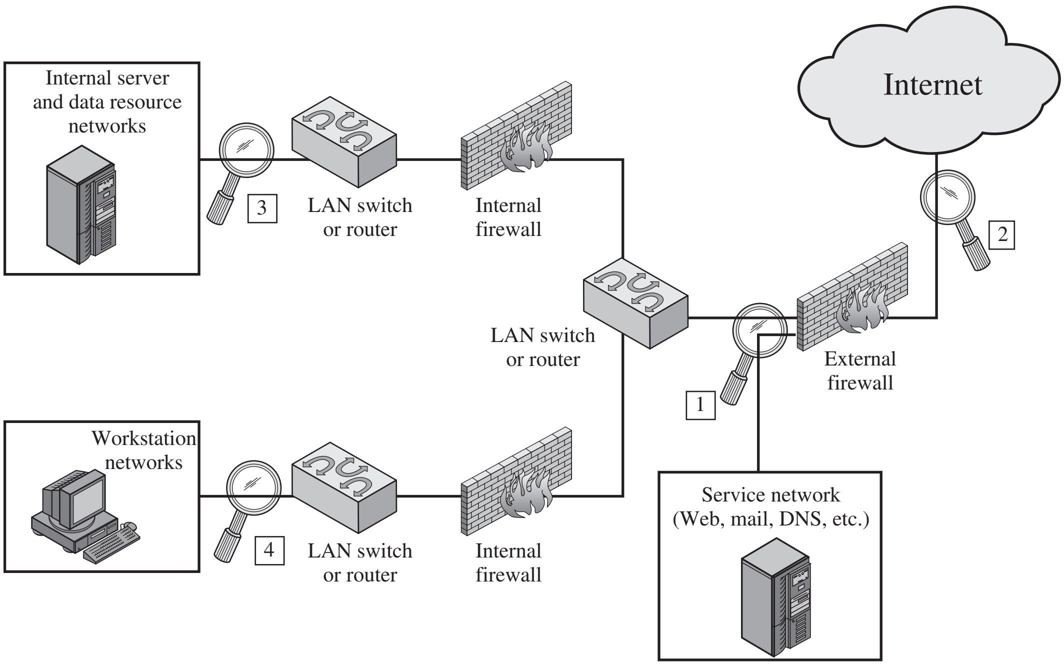


Figure 6.5 Example of NIDS Sensor Deployment

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.

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

[2]Firewalls are 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.

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.

Signature Detection [SCAR07] 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, 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 Web sites and use of forbidden application protocols.

Anomaly Detection Techniques [SCAR07] lists the following as examples of that 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 8. 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 7 discusses worms in detail.

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
- 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)

6.6 DISTRIBUTED ADAPTIVE INTRUSION DETECTION

So far, we have looked at three overlapping and complementary architectures for intrusion detection: host-based, distributed host-based, and network intrusion detection. A distributed host-based IDS makes use of host-based IDSs that can

[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.

communicate with one another. A NIDS focuses on network events and network devices. Both host-based distributed IDSs and NIDSs may involve the use of a central IDS to manage and coordinate intrusion detection and response.

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. 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 rapidly 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 denial-of-service 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 that 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 and 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 that 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 6.6 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]).

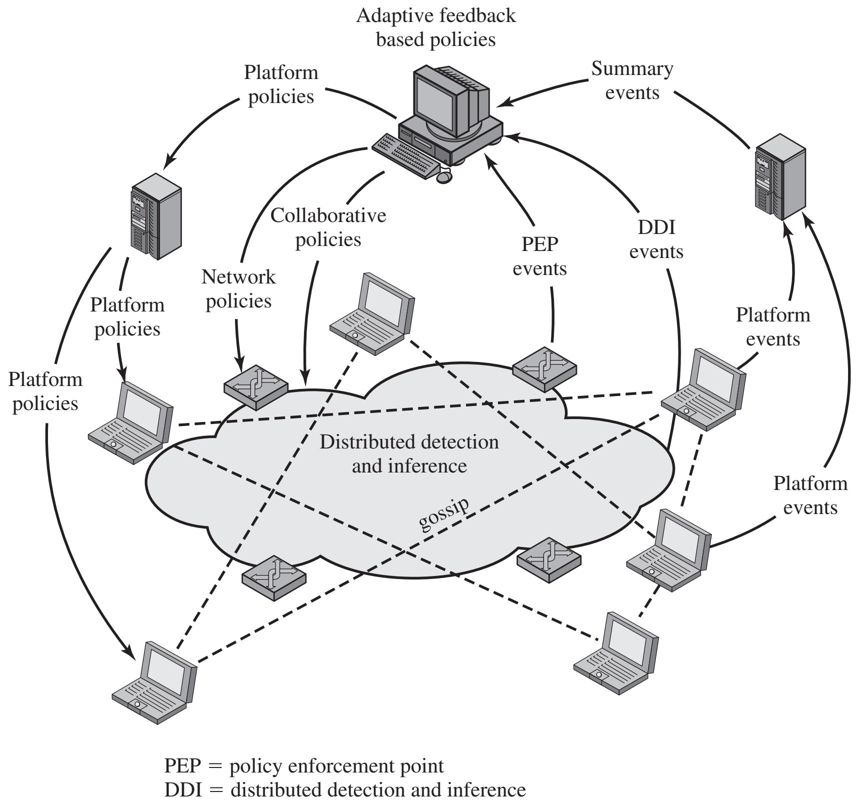


Figure 6.6 **Overall Architecture of an Autonomic Enterprise Security System**

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.

An analogy may help clarify the advantage of this distributed approach. Suppose that a single host is subject to a prolonged attack and that 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.

We now summarize the principal elements of this approach, illustrated in Figure 6.6. 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.

6.7 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 management systems that may need to interact with them. The outputs of this working group include the following:

1. A requirements document, which describes the high-level functional requirements for communication between IDSs and requirements for communication between IDSs and management systems, including the rationale for those requirements. Scenarios will be used to illustrate the requirements.
2. A common intrusion language specification, which describes data formats that satisfy the requirements.
3. A framework document, which identifies existing protocols best used for communication between IDSs and describes how the devised data formats relate to them.

As of this writing, all of these documents are in an Internet-draft document stage.

Figure 6.7 illustrates the key elements of the model on which the intrusion detection message exchange approach is based. This model does not correspond to any particular product or implementation, but its functional components are the key elements of any IDS. The functional components are as follows:

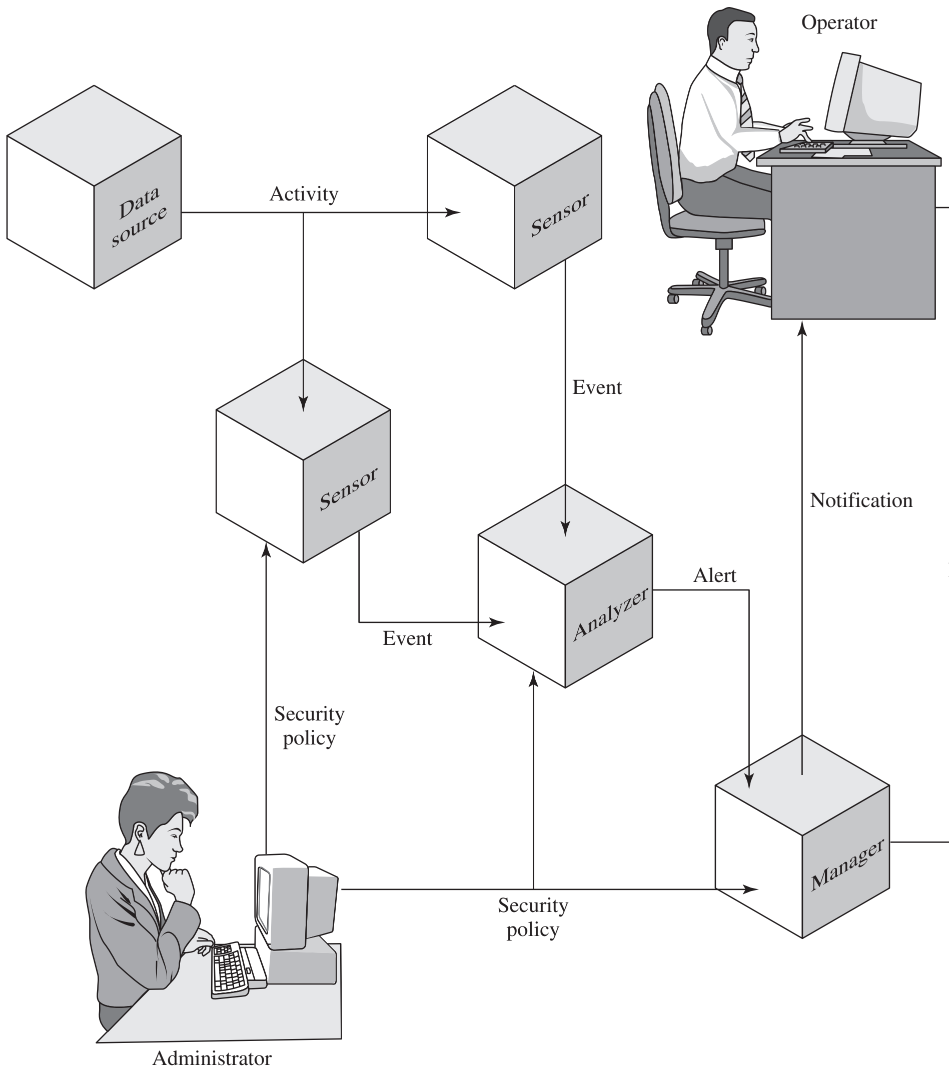


Figure 6.7 Model for Intrusion Detection Message Exchange

- **Data source:** The raw data that an IDS uses to detect unauthorized or undesired activity. Common data sources include network packets, operating system audit logs, application audit logs, and system-generated checksum data.
- **Sensor:** Collects data from the data source. The sensor forwards events to the analyzer.
- **Analyzer:** The ID component or process that analyzes the data collected by the sensor for signs of unauthorized or undesired activity or for events that might be of interest to the security administrator. In many existing IDSs, the sensor and the analyzer are part of the same component.

- **Administrator:** The human with overall responsibility for setting the security policy of the organization, and, thus, for decisions about deploying and configuring the IDS. This may or may not be the same person as the operator of the IDS. In some organizations, the administrator is associated with the network or systems administration groups. In other organizations, it's an independent position.
- **Manager:** The ID component or process from which the operator manages the various components of the ID system. Management functions typically include sensor configuration, analyzer configuration, event notification management, data consolidation, and reporting.
- **Operator:** The human that is the primary user of the IDS manager. The operator often monitors the output of the IDS and initiates or recommends further action.

In this model, intrusion detection proceeds in the following manner. The sensor monitors data sources looking for suspicious **activity**, such as network sessions showing unexpected telnet activity, operating system log file entries showing a user attempting to access files to which he or she is not authorized to have access, and application log files showing persistent login failures. The sensor communicates suspicious activity to the analyzer as an **event**, which characterizes an activity within a given period of time. If the analyzer determines that the event is of interest, it sends an **alert** to the manager component that contains information about the unusual activity that was detected, as well as the specifics of the occurrence. The manager component issues a **notification** to the human operator. A **response** can be initiated automatically by the manager component or by the human operator. Examples of responses include logging the activity; recording the raw data (from the data source) that characterized the event; terminating a network, user, or application session; or altering network or system access controls. The **security policy** is the predefined, formally documented statement that defines what activities are allowed to take place on an organization's network or on particular hosts to support the organization's requirements. This includes, but is not limited to, which hosts are to be denied external network access.

The specification defines formats for event and alter messages, message types, and exchange protocols for communication of intrusion detection information.

6.8 HONEYPOTS

A relatively recent innovation in intrusion detection technology is the honeypot. Honeypots are decoy systems that are designed to lure a potential attacker away from critical systems. Honeypots are designed to

- Divert an attacker from accessing critical systems.
- Collect information about the attacker's activity.
- Encourage the attacker to stay on the system long enough for administrators to respond.

These systems are filled with fabricated information designed to appear valuable but that a legitimate user of the system wouldn't access. Thus, any access to the honeypot is suspect. The system is instrumented with sensitive monitors and event loggers that detect these accesses and collect information about the attacker's activities. Because any attack against the honeypot is made to seem successful, administrators have time to mobilize and log and track the attacker without ever exposing productive systems.

The honeypot is a resource that has no production value. There is no legitimate reason for anyone outside the network to interact with a honeypot. Thus, any attempt to communicate with the system is most likely a probe, scan, or attack. Conversely, if a honeypot initiates outbound communication, the system has probably been compromised.

Initial efforts involved a single honeypot computer with IP addresses designed to attract hackers. More recent research has focused on building entire honeypot networks that emulate an enterprise, possibly with actual or simulated traffic and data. Once hackers are within the network, administrators can observe their behavior in detail and figure out defenses.

Honeypots can be deployed in a variety of locations. Figure 6.8 illustrates some possibilities. The location depends on a number of factors, such as the type of information the organization is interested in gathering and the level of risk that organizations can tolerate to obtain the maximum amount of data.

A honeypot outside the external firewall (**location 1**) is useful for tracking attempts to connect to unused IP addresses within the scope of the network. A honeypot at this location does not increase the risk for the internal network. The danger of having a compromised system behind the firewall is avoided. Further, because the honeypot attracts many potential attacks, it reduces the alerts issued by the firewall and by internal IDS sensors, easing the management burden. The disadvantage of an external honeypot is that it has little or no ability to trap internal attackers, especially if the external firewall filters traffic in both directions.

The network of externally available services, such as Web and mail, often called the DMZ (demilitarized zone), is another candidate for locating a honeypot (**location 2**). The security administrator must assure that the other systems in the DMZ are secure against any activity generated by the honeypot. A disadvantage of this location is that a typical DMZ is not fully accessible, and the firewall typically blocks traffic to the DMZ the attempts to access unneeded services. Thus, the firewall either has to open up the traffic beyond what is permissible, which is risky, or limit the effectiveness of the honeypot.

A fully internal honeypot (**location 3**) has several advantages. Its most important advantage is that it can catch internal attacks. A honeypot at this location can also detect a misconfigured firewall that forwards impermissible traffic from the Internet to the internal network. There are several disadvantages. The most serious of these is if the honeypot is compromised so that it can attack other internal systems. Any further traffic from the Internet to the attacker is not blocked by the firewall because it is regarded as traffic to the honeypot only. Another difficulty for this honeypot location is that, as with location 2, the firewall must adjust its filtering to allow traffic to the honeypot, thus complicating firewall configuration and potentially compromising the internal network.

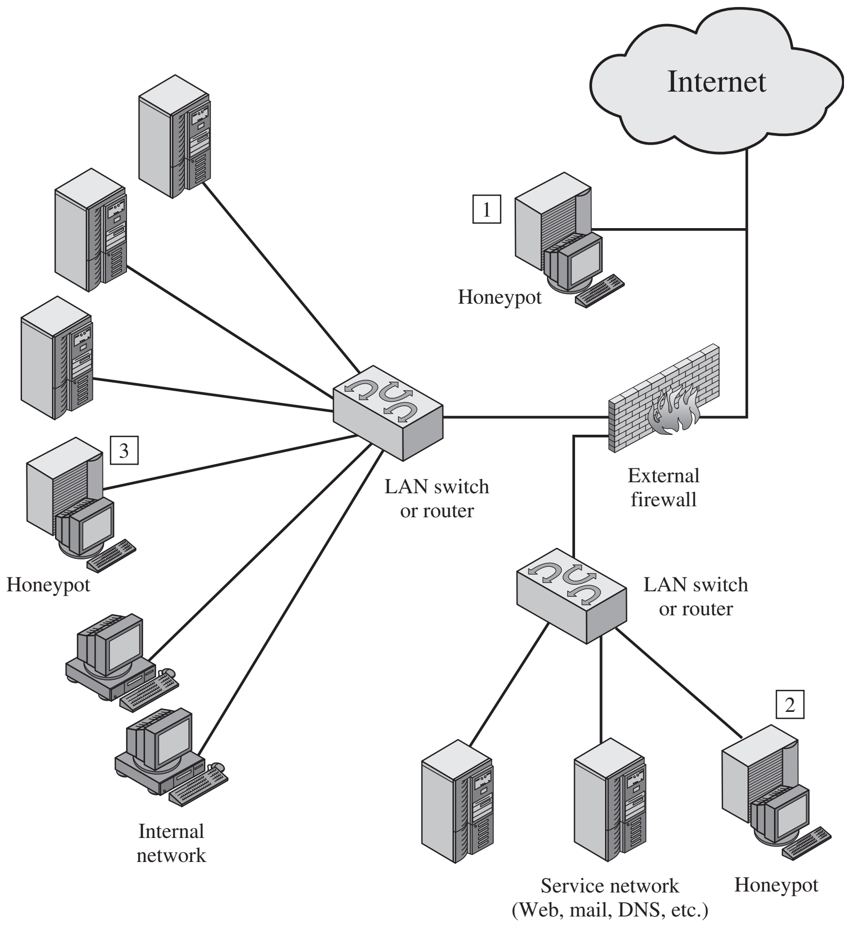


Figure 6.8 Example of Honeypot Deployment

6.9 EXAMPLE SYSTEM: SNORT

Snort is an open source, highly configurable and portable host-based or network-based IDS. Snort is referred to as a lightweight IDS, which has the following characteristics:

- Easily deployed on most nodes (host, server, router) of a network
- Efficient operation that uses small amount of memory and processor time
- Easily configured by system administrators who need to implement a specific security solution in a short amount of time

Snort can perform real-time packet capture, protocol analysis, and content searching and matching. Snort can detect a variety of attacks and probes, based on a set of rules configured by a system administrator.

Snort Architecture

A Snort installation consists of four logical components (Figure 6.9):

- **Packet decoder:** The packet decoder processes each captured packet to identify and isolate protocol headers at the data link, network, transport, and application layers. The decoder is designed to be as efficient as possible and its primary work consists of setting pointers so that the various protocol headers can be easily extracted.
- **Detection engine:** The detection engine does the actual work of intrusion detection. This module analyzes each packet based on a set of rules defined for this configuration of Snort by the security administrator. In essence, each packet is checked against all the rules to determine if the packet matches the characteristics defined by a rule. The first rule that matches the decoded packet triggers the action specified by the rule. If no rule matches the packet, the detection engine discards the packet.
- **Logger:** For each packet that matches a rule, the rule specifies what logging and alerting options are to be taken. When a logger option is selected, the logger stores the detected packet in human readable format or in a more compact binary format in a designated log file. The security administrator can then use the log file for later analysis.
- **Alerter:** For each detected packet, an alert can be sent. The alert option in the matching rule determines what information is included in the event notification. The event notification can be sent to a file, to a UNIX socket, or to a database. Alerting may also be turned off during testing or penetration studies. Using the UNIX socket, the alert can be sent to a management machine elsewhere on the network.

A Snort implementation can be configured as a passive sensor, which monitors traffic but is not in the main transmission path of the traffic, or an inline sensor, through which all packet traffic must pass. In the latter case, Snort can perform intrusion prevention as well as intrusion detection. We defer a discussion of intrusion prevention to Chapter 9.

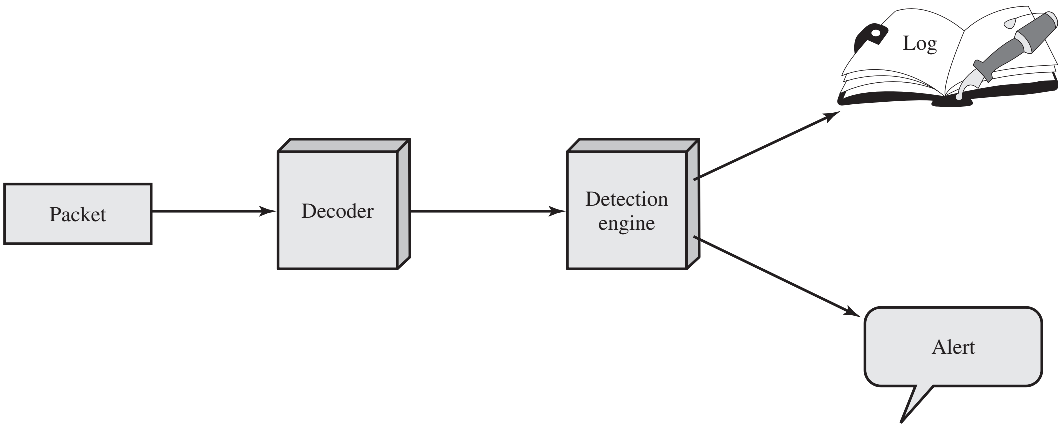


Figure 6.9 **Snort Architecture**

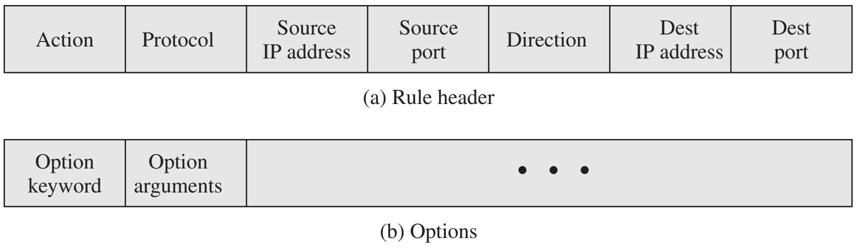


Figure 6.10 Snort Rule Formats

Snort Rules

Snort uses a simple, flexible rule definition language that generates the rules used by the detection engine. Although the rules are simple and straightforward to write, they are powerful enough to detect a wide variety of hostile or suspicious traffic.

Each rule consists of a fixed header and zero or more options (Figure 6.10). The header has the following elements:

- **Action:** The rule action tells Snort what to do when it finds a packet that matches the rule criteria. Table 6.4 lists the available actions. The last three actions in the list (drop, reject, sdrop) are only available in inline mode.
- **Protocol:** Snort proceeds in the analysis if the packet protocol matches this field. The current version of Snort (2.6) recognizes four protocols: TCP, UDP, ICMP, and IP. Future releases of Snort will support a greater range of protocols.
- **Source IP address:** Designates the source of the packet. The rule may specify a specific IP address, any IP address, a list of specific IP addresses, or the negation of a specific IP address or list. The negation indicates that any IP address other than those listed is a match.
- **Source port:** This field designates the source port for the specified protocol (e.g., a TCP port). Port numbers may be specified in a number of ways, including specific port number, any ports, static port definitions, ranges, and by negation.
- **Direction:** This field takes on one of two values: unidirectional (->) or bidirectional (<->). The bidirectional option tells Snort to consider the address/port pairs in the rule as either source followed by destination or destination

Table 6.4 Snort Rule Actions

Action	Description
alert	Generate an alert using the selected alert method, and then log the packet.
log	Log the packet.
pass	Ignore the packet.
activate	Alert and then turn on another dynamic rule.
dynamic	Remain idle until activated by an activate rule, then act as a log rule.
drop	Make iptables drop the packet and log the packet.
reject	Make iptables drop the packet, log it, and then send a TCP reset if the protocol is TCP or an ICMP port unreachable message if the protocol is UDP.
sdrop	Make iptables drop the packet but does not log it.

followed by source. The bidirectional option enables Snort to monitor both sides of a conversation.

- **Destination IP address:** Designates the destination of the packet.
- **Destination port:** Designates the destination port.

Following the rule header may be one or more rule options. Each option consists of an option keyword, which defines the option; followed by arguments, which specify the details of the option. In the written form, the set of rule options is separated from the header by being enclosed in parentheses. Snort rule options are separated from each other using the semicolon (;) character. Rule option keywords are separated from their arguments with a colon (:) character.

There are four major categories of rule options:

- **meta-data:** Provide information about the rule but do not have any affect during detection
- **payload:** Look for data inside the packet payload and can be interrelated
- **non-payload:** Look for non-payload data
- **post-detection:** Rule-specific triggers that happen after a rule has matched a packet

Table 6.5 provides examples of options in each category.

Here is an example of a Snort rule:

```
Alert tcp $EXTERNAL_NET any -> $HOME_NET any\
(msg: "SCAN SYN FIN" flags: SF, 12;\
reference: arachnids, 198; classtype: attempted-recon;)
```

In Snort, the reserved backslash character "\" is used to write instructions on multiple lines. This example is used to detect a type of attack at the TCP level known as a SYN-FIN attack. The names $EXTERNAL_NET and $HOME_NET are predefined

Table 6.5 Examples of Snort Rule Options

meta-data	
msg	defines the message to be sent when a packet generates an event.
reference	Defines a link to an external attack identification system, which provides additional information.
classtype	Indicates what type of attack the packet attempted.
payload	
content	Enables Snort to perform a case-sensitive search for specific content (text and/or binary) in the packet payload.
depth	Specifies how far into a packet Snort should search for the specified pattern. Depth modifies the previous content keyword in the rule.
offset	Specifies where to start searching for a pattern within a packet. Offset modifies the previous content keyword in the rule.
nocase	Snort should look for the specific pattern, ignoring case. Nocase modifies the previous content keyword in the rule.

(continued)

Table 6.5 (continued)

non-payload	
ttl	Check the IP time-to-live value. This option was intended for use in the detection of traceroute attempts.
id	Check the IP ID field for a specific value. Some tools (exploits, scanners and other odd programs) set this field specifically for various purposes, for example, the value 31337 is very popular with some hackers.
dsize	Test the packet payload size. This may be used to check for abnormally sized packets. In many cases, it is useful for detecting buffer overflows.
flags	Test the TCP flags for specified settings.
seq	Look for a specific TCP header sequence number.
icmp-id	Check for a specific ICMP ID value. This is useful because some covert channel programs use static ICMP fields when they communicate. This option was developed to detect the stacheldraht DDoS agent.
post-detection	
logto	Log packets matching the rule to the specified filename.
session	Extract user data from TCP Sessions. There are many cases where seeing what users are typing in telnet, rlogin, ftp, or even web sessions is very useful.

variable names to specify particular networks. In this example, any source port or destination port is specified. This example checks if just the SYN and the FIN bits are set, ignoring reserved bit 1 and reserved bit 2 in the flags octet. The reference option refers to an external definition of this attack, which is of type attempted-recon.

6.10 RECOMMENDED READING AND WEB SITES

Two thorough treatments of intrusion detection are [BACE00] and [PROC01]. Another detailed and worthwhile treatment is [SCAR07]. Two short but useful survey articles on the subject are [KENT00] and [MCHU00]. [NING04] surveys recent advances in intrusion detection techniques. [HONE01] is the definitive account on honeypots and provides a detailed analysis of the tools and methods of hackers.

BACE00 Bace, R. *Intrusion Detection.* Indianapolis, IN: Macmillan Technical Publishing, 2000.

HONE01 The Honeynet Project. *Know Your Enemy: Revealing the Security Tools, Tactics, and Motives of the Blackhat Community.* Reading, MA: Addison-Wesley, 2001.

KENT00 Kent, S. "On the Trail of Intrusions into Information Systems." *IEEE Spectrum*, December 2000.

MCHU00 McHugh, J.; Christie, A.; and Allen, J. "The Role of Intrusion Detection Systems." *IEEE Software*, September/October 2000.

NING04 Ning, P., et al. "Techniques and Tools for Analyzing Intrusion Alerts." *ACM Transactions on Information and System Security*, May 2004.

PROC01 Proctor, P., *The Practical Intrusion Detection Handbook.*` Upper Saddle River, NJ: Prentice Hall, 2001.

SCAR07 Scarfone, K., and Mell, P. *Guide to Intrusion Detection and Prevention Systems.* NIST Special Publication SP 800-94, February 2007.

Recommended Web sites:

- **STAT Project:** A research and open source project that focuses on signature-based intrusion detection tools for hosts, applications, and networks.
- **Honeynet Project:** A research project studying the techniques of predatory hackers and developing honeypot products.
- **Honeypots:** A good collection of research papers and technical articles.
- **Intrusion Detection Exchange Format Working Group:** IETF group developing standards for exchange formats and exchange procedures for intrusion detection systems. Includes RFCs and Internet drafts.
- **Snort:** Web site for Snort, an open source network intrusion prevention and detection system.

6.11 KEY TERMS, REVIEW QUESTIONS, AND PROBLEMS

Key Terms

anomaly detection	inline sensor	network sensor
banner grabbing	intruder	passive sensor
base-rate fallacy	intrusion detection	rule-based anomaly detection
false negative	intrusion detection exchange format	rule-based penetration identification
false positive	intrusion detection system (IDS)	scanning
hacker	network-based IDS (NIDS)	signature detection
honeypot		Snort
host-based IDS		

Review Questions

6.1 List and briefly define three classes of intruders.
6.2 Describe the three logical components of an IDS.
6.3 Describe the differences between a host-based IDS and a network-based IDS.
6.4 What are three benefits that can be provided by an IDS?
6.5 List some desirable characteristics of an IDS.
6.6 What is the difference between anomaly detection and signature intrusion detection?
6.7 What metrics are useful for profile-based intrusion detection?
6.8 What is the difference between rule-based anomaly detection and rule-based penetration identification?
6.9 Explain the base-rate fallacy.
6.10 What is the difference between a distributed host-based IDS and a NIDS?
6.11 Describe the types of sensors that can be used in a NIDS.
6.12 What are possible locations for NIDS sensors?
6.13 What is a honeypot?

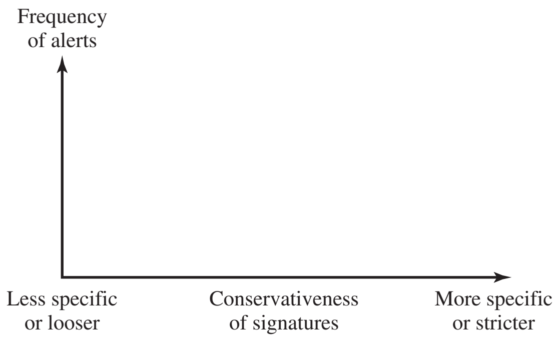
Problems

6.1 Design a file access system to allow certain users read and write access to a file, depending on authorization set up by the system. The instructions should be of the format

READ (F, User A): attempt by User A to read file F
WRITE (F, User A): attempt by User A to store a possibly modified copy of F

Each file has a *header record,* which contains authorization privileges; that is, a list of users who can read and write. The file is to be encrypted by a key that is not shared by the users but known only to the system.

6.2 In the context of an IDS, we define a false positive to be an alarm generated by an IDS in which the IDS alerts to a condition that is actually benign. A false negative occurs when an IDS fails to generate an alarm when an alert-worthy condition is in effect. Using the following diagram, depict two curves that roughly indicate false positives and false negatives, respectively.



6.3 Wireless networks present different problems from wired networks for NIDS deployment because of the broadcast nature of transmission. Discuss the considerations that should come into play when deciding on locations for wireless NIDS sensors.

6.4 One of the non-payload options in Snort is flow. This option distinguishes between clients and servers. This option can be used to specify a match only for packets flowing in one direction (client to server or vice versa) and can specify a match only on established TCP connections. Consider the following Snort rule:

```
alert tcp $EXTERNAL_NET any -> $SQL_SERVERS $ORACLE_PORTS\
(msg: "ORACLE create database attempt:;\
flow: to_server, established; content: "create database";
nocase;\
classtype: protocol-command-decode;)
```

a. What does this rule do?
b. Comment on the significance of this rule if the Snort devices is placed inside or outside of the external firewall.

6.5 The overlapping area of the two probability density functions of Figure 6.1 represents the region in which there is the potential for false positives and false negatives. Further, Figure 6.1 is an idealized and not necessarily representative depiction of the relative shapes of the two density functions. Suppose there is 1 actual intrusion for every 1000 authorized users, and the overlapping area covers 1% of the authorized users and 50% of the intruders.

a. Sketch such a set of density functions and argue that this is not an unreasonable depiction.
b. What is the probability that an event that occurs in this region is that of an authorized user? Keep in mind that 50% of all intrusions fall in this region.

6.6 An example of a host-based intrusion detection tool is the tripwire program. This is a file integrity checking tool that scans files and directories on the system on a regular basis and notifies the administrator of any changes. It uses a protected database of cryptographic checksums for each file checked and compares this value with that recomputed on each file as it is scanned. It must be configured with a list of files and directories to check and what changes, if any, are permissible to each. It can allow, for example, log files to have new entries appended, but not for existing entries to be changed. What are the advantages and disadvantages of using such a tool? Consider the problem of determining which files should only change rarely, which files may change more often and how, and which change frequently and hence cannot be checked. Hence consider the amount of work in both the configuration of the program and on the system administrator monitoring the responses generated.

6.7 A decentralized NIDS is operating with two nodes in the network monitoring anomalous inflows of traffic. In addition, a central node is present, to generate an alarm signal upon receiving input signals from the two distributed nodes. The signatures of traffic inflow into the two IDS nodes follow one of four patterns: P1, P2, P3, P4. The threat levels are classified by the central node based upon the observed traffic by the two NIDS at a given time and are given by the following table:

Threat Level	Signature
Low	1 P1 + 1 P2
Medium	1 P3 + 1 P4
High	2 P4

If, at a given time instance, at least one distributed node generates an alarm signal P3, what is the probability that the observed traffic in the network will be classified at threat level 'Medium'?

6.8 A taxicab was involved in a fatal hit-and-run accident at night. Two cab companies, the Green and the Blue, operate in the city. You are told that

- 85% of the cabs in the city are Green and 15% are Blue.
- A witness identified the cab as Blue.

The court tested the reliability of the witness under the same circumstances that existed on the night of the accident and concluded that the witness was correct in identifying the color of the cab 80% of the time. What is the probability that the cab involved in the incident was Blue rather than Green?

APPENDIX 6A THE BASE-RATE FALLACY

We begin with a review of important results from probability theory, then demonstrate the base-rate fallacy.

Conditional Probability and Independence

We often want to know a probability that is conditional on some event. The effect of the condition is to remove some of the outcomes from the sample space. For example, what is the probability of getting a sum of 8 on the roll of two dice if we know that the face of at least one die is an even number? We can reason as follows. Because one die is even and the sum is even, the second die must show an even number. Thus, there are three equally likely successful outcomes—(2, 6), (4, 4) and (6, 2)—out of a total set of possibilities of [36 − (number of events with both faces odd)] = 36 − 3 × 3 = 27. The resulting probability is 3/27 = 1/9.

Formally, the **conditional probability** of an event A assuming the event B has occurred, denoted by $\Pr[A|B]$, is defined as the ratio

$$\Pr[A \mid B] = \frac{\Pr[AB]}{\Pr[B]}$$

where we assume $\Pr[B]$ is not zero.

In our example, $A = \{\text{sum of 8}\}$ and $B = \{\text{at least one die even}\}$. The quantity $\Pr[AB]$ encompasses all of those outcomes in which the sum is 8 and at least one die is even. As we have seen, there are three such outcomes. Thus, $\Pr[AB] = 3/36 = 1/12$. A moment's thought should convince you that $\Pr[B] = 3/4$. We can now calculate

$$\Pr[A \mid B] = \frac{1/12}{3/4} = \frac{1}{9}$$

This agrees with our previous reasoning.

Two events A and B are called **independent** if $\Pr[AB] = \Pr[A]\Pr[B]$. It can easily be seen that if A and B are independent, $\Pr[A|B] = \Pr[A]$ and $\Pr[B|A] = \Pr[B]$.

Bayes' Theorem

One of the most important results from probability theory is known as Bayes' theorem. First we need to state the total probability formula. Given a set of mutually exclusive events $E_1, E_2, \ldots, En_n$, such that the union of these events covers all possible outcomes, and given an arbitrary event A, then it can be shown that

$$\Pr[A] = \sum_{i=1}^{n} \Pr[A|E_i] \Pr[E_i] \quad \textbf{(6.1)}$$

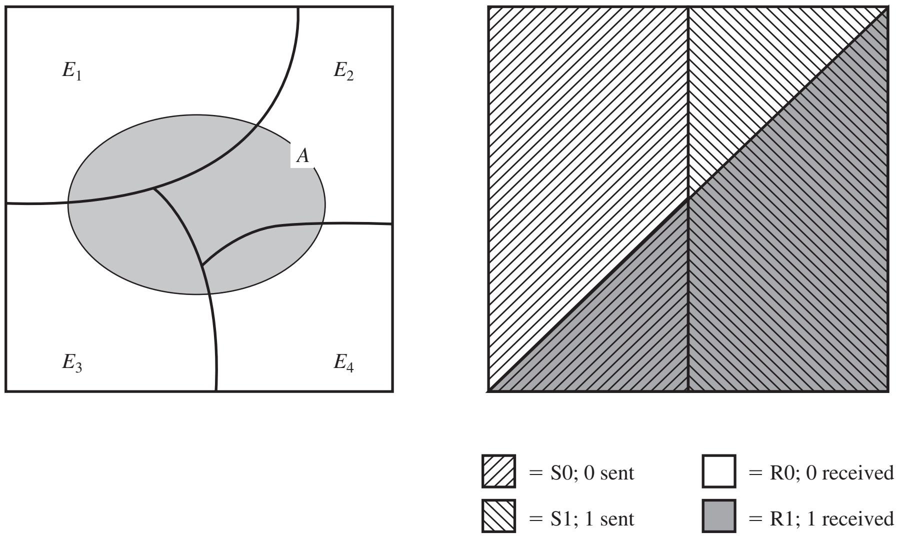
Bayes' theorem may be stated as follows:

$$\Pr[E_i \mid A] = \frac{\Pr[A|E_i]P[E_i]}{\Pr[A]} = \frac{\Pr[A|E_i]P[E_i]}{\sum_{j=1}^{n} \Pr[A|E_j]\Pr[E_j]} \quad \textbf{(6.2)}$$

Figure 6.11a illustrates the concepts of total probability and Bayes' theorem.

Bayes' theorem is used to calculate "posterior odds," that is, the probability that something really is the case, given evidence in favor of it. For example, suppose we are transmitting a sequence of zeroes and ones over a noisy transmission line. Let S0 and S1 be the events a zero is sent at a given time and a one is sent, respectively, and R0 and R1 be the events that a zero is received and a one is received. Suppose we know the probabilities of the source, namely $\Pr[S1] = p$ and $\Pr[S0] = 1 - p$. Now the line is observed to determine how frequently an error occurs when a one is sent and when a zero is sent, and the following probabilities are calculated: $\Pr[R0|S1] = p_a$ and $\Pr[R1|S0] = p_b$. If a zero is received, we can then calculate the conditional probability of an error, namely the conditional probability that a one was sent given that a zero was received, using Bayes' theorem:

$$\Pr[S1 \mid R0] = \frac{\Pr[R0|S1]\Pr[S1]}{\Pr[R0|S1]\Pr[S1] + \Pr[R0|S0]\Pr[S0]} = \frac{p_a p}{p_a p + (1 - p_b)(1 - p)}$$



(a) Diagram to illustrate concepts (b) Example

Figure 6.11 Illustration of Total Probability and Bayes' Theorem

Figure 6.11b illustrates the preceding equation. In the figure, the sample space is represented by a unit square. Half of the square corresponds to S0 and half to S1, so Pr[S0] = Pr[S1] = 0.5. Similarly, half of the square corresponds to R0 and half to R1, so Pr[R0] = Pr[R1] = 0.5. Within the area representing S0, one-quarter of that area corresponds to R1, so Pr[R1/S0] = 0.25. Other conditional probabilities are similarly evident.

The Base-Rate Fallacy Demonstrated

Consider the following situation. A patient has a test for some disease that comes back positive (indicating he has the disease). You are told that

- The accuracy of the test is 87% (i.e., if a patient has the disease, 87% of the time, the test yields the correct result, and if the patient does not have the disease, 87% of the time, the test yields the correct result).
- The incidence of the disease in the population is 1%.

Given that the test is positive, how probable is it that the patient does not have the disease? That is, what is the probability that this is a false alarm? We need Bayes' theorem to get the correct answer:

$$\text{Pr [well/positive]} = \frac{\text{Pr [positive/well]Pr[well]}}{\text{Pr[positive/disease]Pr[disease]} + \text{Pr [positive/well]Pr[well]}}$$

$$= \frac{(0.13)(0.99)}{(0.87)(0.01) + (0.13)(0.99)} = 0.937$$

Thus, in the vast majority of cases, when a disease condition is detected, it is a false alarm.

This problem, used in a study [PIAT91, PIAT94], was presented to a number of people. Most subjects gave the answer 13%. The vast majority, including many physicians, gave a number below 50%. Many physicians who guessed wrong lamented, "If you are right, there is no point in making clinical tests!" The reason most people get it wrong is that they do not take into account the basic rate of incidence (the baserate) when intuitively solving the problem. This error is known as the *base-rate fallacy* [BARH80].

How could this problem be fixed? Suppose we could drive both of the correct result rates to 99.9%. That is, suppose we have Pr[positive/disease] = 0.999 and Pr[negative/well] = 0.999. Plugging these numbers into the Equation. (62), we get Pr[well/positive] = 0.09. Thus, if we can accurately detect disease and accurately detect lack of disease at a level of 99.9%, then the rate of false alarms will be 9%. This is much better, but still not ideal. Moreover, again assume 99.9% accuracy, but now suppose that the incidence of the disease in the population is only 1/10000 = 0.0001. We then end up with a rate of false alarms of 91%. In actual situations, [AXEL00] found that the probabilities associated with IDSs were such that the false alarm rate was unsatisfactory.