PART IV

Security in Transmissions

- **Chapter 11** Intrusion Detection Systems
- **Chapter 12** Security Baselines
- **Chapter 13** Types of Attacks and Malicious Software
- **Chapter 14** E-Mail and Instant Messaging
- **Chapter 15** Web Components

CHAPTER 11

Intrusion Detection Systems

In this chapter, you will

- Understand host-based intrusion detection systems
- Understand PC-based malware protection
- Explore network-based intrusion detection systems
- Explore network traffic shaping and filtering tools
- Learn what honeypots are used for

Ensuring network security can be fairly easily compared to ensuring physical security—the more you want to protect and restrict access to an asset, the more security you need. In the world of physical security, you can use locks, walls, gates, guards, motion sensors, pressure plates, and so on, to protect physical assets. As you add more protective devices, you add "layers" of security that an intruder would have to overcome or breach to obtain access to whatever you are protecting. Correspondingly, in the network and data security arenas, you use protective layers in the form of passwords, firewalls, access lists, file permissions, and Intrusion Detection Systems (IDSs). Most organizations use their own approaches to network security, choosing the layers that make sense for them after they weigh risks, potentials for loss, costs, and manpower requirements.

The foundation for a layered network security approach usually starts with a well-secured system, regardless of the system's function (whether it's a user PC or a corporate e-mail server). A well-secured system uses up-to-date application and operating system patches, well-chosen passwords, the minimum number of services running, and restricted access to available services. On top of that foundation, you can add layers of protective measures such as antivirus products, firewalls, sniffers, and IDSs.

Some of the more complicated and interesting types of network/data security devices are IDSs, which are to the network world what burglar alarms are to the physical world. The main purpose of an IDS is to identify suspicious or malicious activity, note activity that deviates from normal behavior, catalog and classify the activity, and, if possible, respond to the activity. This chapter looks at the history of IDSs and various types of IDSs, considers how they work and the benefits and weaknesses of specific types, and what the future might hold for these systems. You'll also look at some topics complementary to IDSs: malware protection, traffic shaping/filtering, and honeypots.

History of Intrusion Detection Systems

Like much of the network technology we see today, IDSs grew from a need to solve specific problems. Like the Internet itself, the IDS concept came from U.S. Department of Defense–sponsored research. In the early 1970s, the U.S. government and military became increasingly aware of the need to protect the electronic networks that were becoming critical to daily operations. In 1972, James Anderson published a paper for the U.S. Air Force outlining the growing number of computer security problems and the immediate need to secure Air Force systems (James P. Anderson, "Computer Security Technology Planning Study Volume 2," October 1972, http://seclab.cs.ucdavis.edu/projects/history/papers/ande72.pdf). Anderson continued his research and in 1980 published a follow-up paper outlining methods to improve security auditing and surveillance methods ("Computer Security Threat Monitoring and Surveillance," April 15, 1980, http://csrc.nist.gov/publications/history/ande80.pdf). In this paper, Anderson pioneered the concept of using system audit files to detect unauthorized access and misuse. He also suggested the use of automated detection systems, which paved the way for misuse detection on mainframe systems in use at the time.

While Anderson's work got the efforts started, the concept of a real-time, rule-based IDS didn't really exist until Dorothy Denning and Peter Neumann developed the first real-time IDS model, called The Intrusion Detection Expert System (IDES), from their research between 1984 and 1986. In 1987, Denning published "An Intrusion-Detection Model," a paper that laid out the model on which most modern IDSs are based (and which appears in *IEEE Transactions on Software Engineering*, Vol. SE-13, No. 2 [February 1987]: 222–232).

With a model and definitions in place, the U.S. government continued to fund research that led to projects such as Discovery, Haystack, Multics Intrusion Detection and Alerting System (MIDAS), and Network Audit Director and Intrusion Reporter (NADIR). Finally, in 1989, Haystack Labs released Stalker, the first commercial IDS. Stalker was host-based and worked by comparing audit data to known patterns of suspicious activity. While the military and government embraced the concept, the commercial world was very slow to adopt IDS products, and it was several years before other commercial products began to emerge.

In the early to mid-1990s, computer systems continued to grow and companies were starting to realize the importance of IDSs; however, the solutions available were host-based and required a great deal of time and money to manage and operate effectively. Focus began to shift away from host-based systems, and network-based IDSs began to emerge. In 1995, WheelGroup was formed in San Antonio, Texas, to develop the first commercial network-based IDS product, called NetRanger. NetRanger was designed to monitor network links and the traffic moving across the links to identify misuse as well as suspicious and malicious activity. NetRanger's release was quickly followed by Internet Security Systems' RealSecure in 1996. Several other players followed suit and released their own IDS products, but it wasn't until the networking giant Cisco Systems acquired WheelGroup in February 1998 that IDSs were recognized as a vital part of any network security infrastructure. Figure 11-1 offers a timeline for these developments.

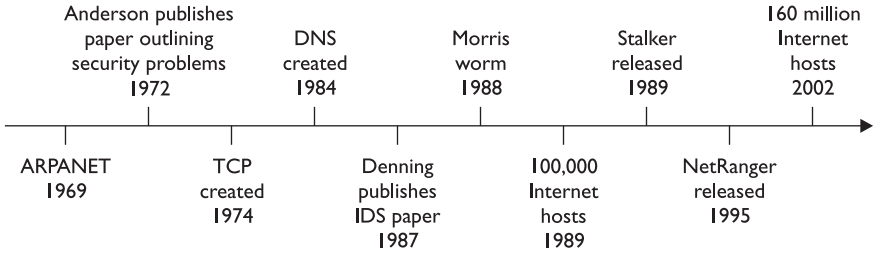


Figure 11-1 History of the Internet and IDS

IDS Overview

As mentioned, an IDS is somewhat like a burglar alarm. It watches the activity going on around it and tries to identify undesirable activity. IDSs are typically divided into two main categories, depending on how they monitor activity:

- **Host-based IDS** Examines activity on an individual system, such as a mail server, web server, or individual PC. It is concerned only with an individual system and usually has no visibility into the activity on the network or systems around it.
- **Network-based IDS** Examines activity on the network itself. It has visibility only into the traffic crossing the network link it is monitoring and typically has no idea of what is happening on individual systems.

EXAM TIP Know the differences between host-based and network-based IDSs. A host-based IDS runs on a specific system (server or workstation) and looks at all the activity on that host. A network-based IDS sniffs traffic from the network and sees only activity that occurs on the network.

Whether or not it is network- or host-based, an IDS will typically consist of several specialized components working together, as illustrated in Figure 11-2. These components are often logical and software-based rather than physical and will vary slightly from vendor to vendor and product to product. Typically, an IDS will have the following logical components:

- **Traffic collector (or sensor)** This component collects activity/events for the IDS to examine. On a host-based IDS, this could be log files, audit logs, or traffic coming to or leaving a specific system. On a network-based IDS, this is typically a mechanism for copying traffic off the network link—basically functioning as a sniffer. This component is often referred to as a sensor.
- **Analysis engine** This component examines the collected network traffic and compares it to known patterns of suspicious or malicious activity stored in the signature database. The analysis engine is the "brains" of the IDS.
- **Signature database** The signature database is a collection of patterns and definitions of known suspicious or malicious activity.

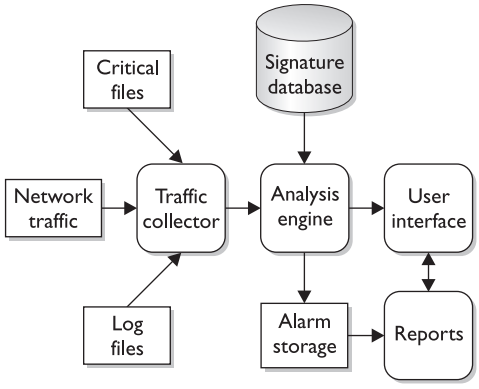


Figure 11-2 Logical depiction of IDS components

- **User interface and reporting** This component interfaces with the human element, providing alerts when appropriate and giving the user a means to interact with and operate the IDS.

Most IDSs can be tuned to fit a particular environment. Certain signatures can be turned off, telling the IDS not to look for certain types of traffic. For example, if you are operating in a pure UNIX environment, you may not wish to see Windows-based alarms, as they will not affect your systems. Additionally, the severity of the alarm levels can be adjusted depending on how concerned you are over certain types of traffic. Some IDSs will also allow the user to exclude certain patterns of activity from specific hosts. In other words, you can tell the IDS to ignore the fact that some systems generate traffic that looks like malicious activity, because it really isn't.

Host-based IDSs

The first IDSs were host-based and designed to examine activity only on a specific host. A host-based IDS (HIDS) examines log files, audit trails, and network traffic coming into or leaving a specific host. HIDSs can operate in *real time*, looking for activity as it occurs, or in *batch mode*, looking for activity on a periodic basis. Host-based systems are typically self-contained, but many of the newer commercial products have been designed to report to and be managed by a central system. Host-based systems also take local system resources to operate. In other words, a HIDS will use up some of the memory and CPU cycles of the system it is protecting. Early versions of HIDSs ran in batch mode, looking for suspicious activity on an hourly or daily basis, and typically looked only for specific events in a system's log files. As processor speeds increased, later versions of HIDSs looked through the log files in real time and even added the ability to examine the data traffic the host was generating and receiving.

Most HIDSs focus on the log files or audit trails generated by the local operating system. On UNIX systems, the examined logs usually include those created by syslog such as messages, kernel logs, and error logs. On Windows systems, the examined logs are typically the three event logs: Application, System, and Security. Some HIDSs can cover specific applications, such as FTP or web services, by examining the logs produced

by those specific applications or examining the traffic from the services themselves. Within the log files, the HIDS is looking for certain activities that typify hostile actions or misuse, such as the following:

- Logins at odd hours
- Login authentication failures
- Additions of new user accounts
- Modification or access of critical system files
- Modification or removal of binary files (executables)
- Starting or stopping processes
- Privilege escalation
- Use of certain programs

In general, most HIDSs will operate in a very similar fashion. (Figure 11-3 shows the logical layout of a HIDS.) By considering the function and activity of each component, you can gain some insight into how HIDSs operate.

As on any IDS, the *traffic collector* on a HIDS pulls in the information the other components, such as the analysis engine, need to examine. For most host-based systems, the traffic collector pulls data from information the local system has already generated, such as error messages, log files, and system files. The traffic collector is responsible for reading those files, selecting which items are of interest, and forwarding them to the analysis engine. On some host-based systems, the traffic collector will also examine specific attributes of critical files such as file size, date modified, or checksum.

NOTE Critical files are those that are vital to the system's operation or overall functionality. They may be program (or binary) files, files containing user accounts and passwords, or even scripts to start or stop system processes. Any unexpected modifications to these files could mean the system has been compromised or modified by an attacker. By monitoring these files, the IDS can warn users of potentially malicious activity.

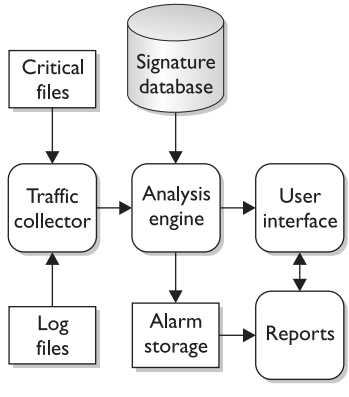
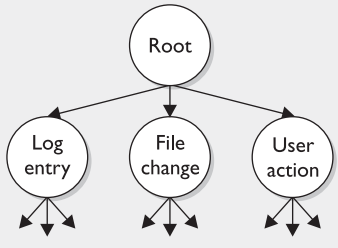


Figure 11-3 Host-based IDS components

Decision Tree

In computer systems, a *tree* is a data structure where each element in the structure is attached to one or more structures directly beneath it (the connections are called *branches*). Structures on the end of a branch without any elements below them are called *leaves*. Trees are most often drawn inverted, with the *root* at the top and all subsequent elements branching down from the root. Trees where each element has no more than two elements below it are called *binary* trees.

In intrusion detection systems, a decision tree is used to help the analysis engine quickly examine traffic patterns. The decision tree helps the analysis engine eliminate signatures that don't apply to the particular traffic being examined so that the fewest number of comparisons can be made. For example, in the following illustration, the sample IDS decision tree shown may contain a section dividing the traffic into three sections based upon origin of the traffic (a log entry for events taken from the system logs, file changes for modifications to critical files, or user actions for something a user has done). When the analysis engine looks at the traffic pattern and starts down the decision tree, it must decide which path to follow. If it is a log entry, the analysis engine can then concentrate on only the signatures that apply to log entries; it does not need to worry about signatures that apply to file changes or user actions. This type of decision tree allows the analysis engine to function much faster, as it does not have to compare traffic to every signature in the database, just the signatures that apply to that particular type of traffic.



The *analysis engine* is perhaps the most important component of the IDS, as it must decide what activity is "okay" and what activity is "bad." The analysis engine is a sophisticated decision and pattern-matching mechanism—it looks at the information provided by the traffic collector and tries to match it against known patterns of activity stored in the signature database. If the activity matches a known pattern, the analysis engine can react, usually by issuing an alert or alarm. An analysis engine may also be capable of remembering how the activity it is looking at right now compares to traffic it has already seen or may see in the near future so that it can match more complicated, multistep malicious activity patterns. An analysis engine must also be capable of examining traffic patterns as quickly as possible, as the longer it takes to match a malicious pattern, the less time the IDS or human operator has to react to malicious traffic. Most

IDS vendors build a "decision tree" into their analysis engines to expedite pattern matching.

The *signature database* is a collection of predefined activity patterns that have already been identified and categorized—patterns that typically indicate suspicious or malicious activity. When the analysis engine has a traffic pattern to examine, it will compare that pattern to the appropriate signatures in the database. The signature database can contain anywhere from a few to a few thousand signatures, depending on the vendor, type of IDS, space available on the system to store signatures, and other factors.

The *user interface* is the visible component of the IDS—the part that humans interact with. The user interface varies widely depending on the product and vendor and could be anything from a detailed GUI to a simple command line. Regardless of the type and complexity, the interface is provided to allow the user to interact with the system: changing parameters, receiving alarms, tuning signatures and response patterns, and so on.

To better understand how a HIDS operates, take a look at examples from a UNIX system and a Windows system.

On a UNIX system, the HIDS is likely going to examine any of a number of system logs—basically large text files containing entries about what is happening on the system. For this example, consider the following lines from the "messages" log on a Red Hat system:

```
Jan 5 18:20:39 jeep su(pam_unix)[32478]: session opened for user bob by (uid=0)
Jan 5 18:20:47 jeep su(pam_unix)[32516]: authentication failure;
   logname= uid=502 euid=0 tty= ruser=bob rhost= user=root
Jan 5 18:20:53 jeep su(pam_unix)[32517]: authentication failure; logname= id=5
02 euid=0 tty= ruser=bob rhost= user=root
Jan 5 18:21:06 jeep su(pam_unix)[32519]: authentication failure; logname= uid=5
02 euid=0 tty= ruser=bob rhost= user=root
```

In the first line, you see a session being opened by a user named *bob*. This usually indicates that whoever owns the account bob has logged into the system. On the next three lines, you see authentication failures as bob tries to become *root*—the superuser account that can do anything on the system. In this case, user bob tries three times to become root and fails on each try. This pattern of activity could mean a number of different things—bob could be an admin who has forgotten the password for the root account, bob could be an admin and someone changed the root password without telling him, bob could be a user attempting to guess the root password, or an attacker could have compromised user bob's account and is now trying to compromise the root account on the system. In any case, our HIDS will work through its decision tree to determine whether an authentication failure in the message log is something it needs to examine. In this instance, when the IDS examines these lines in the log, it will note the fact that three of the lines in the log match one of the patterns it has been told to look for (as determined by information from the decision tree and the signature database), and it will react accordingly, usually by generating an alarm or alert of some type that appears on the user interface or in an e-mail, page, or other form of message.

On a Windows system, the HIDS will likely examine the application logs generated by the operating system. The three logs (application, system, and security) are similar to the logs on a UNIX system, though the Windows logs are not stored as text files and typically require a utility or application to read them. This example uses the security log from a Windows 2000 Professional system:

```
Failure Audit  1/5/2003  6:47:29 PM  Security  Logon/Logoff        529  SYSTEM
Failure Audit  1/5/2003  6:47:27 PM  Security  Logon/Logoff        529  SYSTEM
Failure Audit  1/5/2003  6:47:26 PM  Security  Logon/Logoff        529  SYSTEM
Success Audit  1/5/2003  6:47:13 PM  Security  Privilege Use       578  Administrator
Success Audit  1/5/2003  6:47:12 PM  Security  Privilege Use       577  Administrator
Success Audit  1/5/2003  6:47:12 PM  Security  Privilege Use       577  Administrator
Success Audit  1/5/2003  6:47:06 PM  Security  Account Management  643  SYSTEM
Success Audit  1/5/2003  6:46:59 PM  Security  Account Management  643  SYSTEM
```

In the first three lines of the security log, you see a Failure Audit entry for the Logon/Logoff process. This indicates someone has tried to log in to the system three times and has failed each time (much like our UNIX example). You won't see the name of the account until you expand the log entry within the Windows event viewer tool, but for this example, assume it was the Administrator account—the Windows equivalent of the root account. Here again, you see three login failures—if the HIDS has been programmed to look for failed login attempts, it will generate alerts when it examines these log entries.

Advantages of HIDSs

HIDSs have certain advantages that make them a good choice for certain situations:

- *They can be very operating system–specific and have more detailed signatures.* A HIDS can be very specifically designed to run on a certain operating system or to protect certain applications. This narrow focus lets developers concentrate on the specific things that affect the specific environment they are trying to protect. With this type of focus, the developers can avoid generic alarms and develop much more specific, detailed signatures to identify malicious traffic more accurately.
- *They can reduce false positive rates.* When running on a specific system, the IDS process is much more likely to be able to determine whether the activity being examined is malicious. By more accurately identifying which activity is "bad," the IDS will generate fewer false positives (alarms generated when the traffic matches a pattern but is not actually malicious).
- *They can examine data after it has been decrypted.* With security concerns constantly on the rise, many developers are starting to encrypt their network communications. When designed and implemented in the right manner, a HIDS will be able to examine traffic that is unreadable to a network-based IDS. This particular ability is becoming more important each day as more and more websites start to encrypt all of their traffic.
- *They can be very application specific.* On a host level, the IDS can be designed, modified, or tuned to work very well on specific applications without having

to analyze or even hold signatures for other applications that are not running on that particular system. Signatures can be built for specific versions of web server software, FTP servers, mail servers, or any other application housed on that host.

- *They can determine whether or not an alarm may impact that specific system.* The ability to determine whether or not a particular activity or pattern will really affect the system being protected assists greatly in reducing the number of generated alarms. As the IDS resides on the system, it can verify things such as patch levels, presence of certain files, and system state when it analyzes traffic. By knowing what state the system is in, the IDS can more accurately determine whether an activity is potentially harmful to the system.

Disadvantages of HIDSs

HIDSs also have certain disadvantages that must be weighed into the decision to deploy this type of technology:

- *The IDS must have a process on every system you want to watch.* You must have an IDS process or application installed on every host you want to watch. To watch 100 systems, then, you would need to deploy 100 HIDSs.
- *The IDS can have a high cost of ownership and maintenance.* Depending on the specific vendor and application, a HIDS can be fairly costly in terms of time and manpower to maintain. Unless some type of central console that allows you to maintain remote processes, administrators must maintain each IDS process individually. Even with a central console, with a HIDS, there will be a high number of processes to maintain, software to update, and parameters to tune.
- *The IDS uses local system resources.* To function, the HIDS must use CPU cycles and memory from the system it is trying to protect. Whatever resources the IDS uses are no longer available for the system to perform its other functions. This becomes extremely important on applications such as high-volume web servers where fewer resources usually means fewer visitors served and the need for more systems to handle expected traffic.
- *The IDS has a very focused view and cannot relate to activity around it.* The HIDS has a limited view of the world, as it can see activity only on the host it is protecting. It has little to no visibility into traffic around it on the network or events taking place on other hosts. Consequently, a HIDS can tell you only if the system it is running on is under attack.
- *The IDS, if logged locally, could be compromised or disabled.* When an IDS generates alarms, it will typically store the alarm information in a file or database of some sort. If the HIDS stores its generated alarm traffic on the local system, an attacker that is successful in breaking into the system may be able to modify or delete those alarms. This makes it difficult for security personnel to discover the intruder and conduct any type of post-incident investigation. A capable intruder may even be able to turn off the IDS process completely.

Active vs. Passive HIDSs

Most IDSs can be distinguished by how they examine the activity around them and whether or not they interact with that activity. This is certainly true for HIDSs. On a *passive* system, the IDS is exactly that—it simply watches the activity, analyzes it, and generates alarms. It does not interact with the activity itself in any way, and it does not modify the defensive posture of the system to react to the traffic. A passive IDS is similar to a simple motion sensor—it generates an alarm when it matches a pattern much as the motion sensor generates an alarm when it sees movement.

An *active* IDS will contain all the same components and capabilities of the passive IDS with one critical exception—the active IDS can *react* to the activity it is analyzing. These reactions can range from something simple, such as running a script to turn a process on or off, to something as complex as modifying file permissions, terminating the offending processes, logging off specific users, and reconfiguring local capabilities to prevent specific users from logging in for the next 12 hours.

Resurgence and Advancement of HIDSs

The past few years have seen a strong resurgence in the use of HIDS. With the great advances in processer power, the introduction of multi-core processors, and the increased capacity of hard drives and memory systems, some of the traditional barriers to running a HIDS have been overcome. Combine that with the widespread adoption of always-on broadband connections and a rise in the use of telecommuting, and a greater overall awareness of the need for computer security and solutions such as HIDS start to become an attractive and sometimes effective solution for business and home users alike.

The latest generation of HIDS have introduced new capabilities designed to stop attacks by preventing them from ever executing or accessing protected files in the first place, rather than relying on a specific signature set that only matches known attacks. The more advanced host-based offerings, which most vendors refer to as host-based intrusion prevention systems (IPS), combine the following elements into a single package:

- **Integrated system firewall** The firewall component checks all network traffic passing into and out of the host. Users can set rules for what types of traffic they want to allow into or out of their system.
- **Behavioral- and signature-based IDS** This hybrid approach uses signatures to match well-known attacks and generic patterns for catching "zero-day" or unknown attacks for which no signatures exist.
- **Application control** This allows administrators to control how applications are used on the system and whether or not new applications can be installed. Controlling the addition, deletion, or modification of existing software can be a good way to control a system's baseline and prevent malware from being installed.
- **Enterprise management** Some host-based products are installed with an "agent" that allows them to be managed by and report back to a central server.

This type of integrated remote management capability is essential in any large-scale deployment of host-based IDS/IPS.

- **Malware detection and prevention** Some HIDSs/HIPSs include scanning and prevention capabilities that address spyware, malware, rootkits, and other malicious software.

PC-based Malware Protection

In the early days of PC use, threats were limited: most home users were not connected to the Internet 24/7 through broadband connections, and the most common threat was a virus passed from computer to computer via an infected floppy disk (much like the medical definition, a computer virus is something that can infect the host and replicate itself). But things have changed dramatically over the last decade and current threats pose a much greater risk than ever before. According to SANS Internet Storm Center, the average survival time of an unpatched Windows PC on the Internet is less than 60 minutes (http://isc.sans.org/survivaltime.html). This is the estimated time before an automated probe finds the system, penetrates it, and compromises it. Automated probes from botnets and worms are not the only threats roaming the Internet—viruses and malware spread by e-mail, phishing, infected websites that execute code on your system when you visit them, adware, spyware, and so on. Fortunately, as the threats increase in complexity and capability, so do the products designed to stop them.

Antivirus Products

Antivirus products attempt to identify, neutralize, or remove malicious programs, macros, and files. These products were initially designed to detect and remove computer viruses, though many of the antivirus products are now bundled with additional security products and features. At the present time, there is no real consensus regarding the first antivirus product. The first edition of Polish antivirus software *mks_vir* was released in 1987, and the first publicly-known neutralization of a PC virus was performed by European Bernt Fix (also known as Bernd) early in the same year. By 1990, software giants McAfee and Norton both had established commercial antivirus products.

Although antivirus products have had nearly two decades to refine their capabilities, the purpose of the antivirus products remains the same: to detect and eliminate computer viruses and malware. Most antivirus products combine the following approaches when scanning for viruses:

- **Signature-based scanning** Much like an IDS, the antivirus products scan programs, files, macros, e-mails, and other data for known worms, viruses, and malware. The antivirus product contains a virus dictionary with thousands of known virus signatures that must be frequently updated, as new viruses are discovered daily. This approach will catch known viruses but is limited by the virus dictionary—what it does not know about it cannot catch.
- **Heuristic scanning (or analysis)** Heuristic scanning does not rely on a virus dictionary. Instead, it looks for suspicious behavior—anything that does not fit into a "normal" pattern of behavior for the operating system and applications running on the system being protected.

As signature-based scanning is a familiar concept, let's examine heuristic scanning in more detail. Heuristic scanning typically looks for commands or instructions that are not normally found in application programs, such as attempts to access a reserved memory register. Most antivirus products use either a weight-based or rule-based system in their heuristic scanning (more effective products use a combination of both techniques). A weight-based system rates every suspicious behavior based on the degree of threat associated with that behavior. If the set threshold is passed based on a single behavior or combination of behaviors, the antivirus product will treat the process, application, macro, and so on, performing those behaviors as a threat to the system. A rules-based system compares activity to a set of rules meant to detect and identify malicious software. If part of the software matches a rule or a process, application, macro, and so on, and performs a behavior that matches a rule, the antivirus software will treat that as a threat to the local system.

Some heuristic products are very advanced and contain capabilities for examining memory usage and addressing, a parser for examining executable code, a logic flow analyzer, and a disassembler/emulator so they can "guess" what the code is designed to do and whether or not it is malicious.

As with IDS/IPS products, encryption poses a problem for antivirus products: anything that cannot be read cannot be matched against current virus dictionaries or activity patterns. To combat the use of encryption in malware and viruses, many heuristic scanners look for encryption and decryption loops. As malware is usually designed to run alone and unattended, if it uses encryption, it must contain all the instructions to encrypt and decrypt itself as needed. Heuristic scanners look for instructions such as the initialization of a pointer with a valid memory address, manipulation of a counter, or a branch condition based on a counter value. While these actions don't always indicate the presence of an encryption/decryption loop, if the heuristic engine can find a loop it might be able to decrypt the software in a protected memory space, such as an emulator, and evaluate the software in more detail. Many viruses share common encryption/decryption routines that help antivirus developers.

Current antivirus products are highly configurable and most offerings will have the following capabilities:

- **Automated updates** Perhaps the most important feature of a good antivirus solution is its ability to keep itself up to date by automatically downloading the latest virus signatures on a frequent basis. This usually requires that the system be connected to the Internet in some fashion and updates should be performed on a daily (or more frequent) basis.
- **Automated scanning** Most antivirus products allow for the scheduling of automated scans when the antivirus product will examine the local system for infected files. These automated scans can typically be scheduled for specific days and times, and the scanning parameters can be configured to specify what drives, directories, and types of files are scanned.
- **Media scanning** Removable media is still a common method for virus and malware propagation, and most antivirus products can be configured to automatically scan CDs, USB drives, memory sticks, or any other type of

removable media as soon as they are connected to or accessed by the local system.

- **Manual scanning** Many antivirus products allow the user to scan drives, files, or directories "on demand."
- **E-mail scanning** E-mail is still a major method of virus and malware propagation. Many antivirus products give users the ability to scan both incoming and outgoing messages as well as any attachments.
- **Resolution** When the antivirus product detects an infected file or application, it can typically perform one of several actions. The antivirus product may quarantine the file, making it inaccessible; it may try and repair the file by removing the infection or offending code; or it may delete the infected file. Most antivirus products allow the user to specify the desired action, and some allow for an escalation in actions such as cleaning the infected file if possible and quarantining the file if it cannot be cleaned.

Antivirus solutions are typically installed on individual systems (desktops and servers), but network-based antivirus capabilities are also available in many commercial gateway products. These gateway products often combine firewall, IDS/IPS, and antivirus capabilities into a single integrated platform. Most organizations will also employ antivirus solutions on e-mail servers, as that continues to be a very popular propagation method for viruses.

While the installation of a good antivirus product is still considered a necessary best practice, there is growing concern about the effectiveness of antivirus products against developing threats. Early viruses often exhibited destructive behaviors; were poorly written, modified files; and were less concerned with hiding their presence than they were with propagation. We are seeing an emergence of viruses and malware created by professionals, sometimes financed by criminal organizations that go to great lengths to hide their presence. These viruses and malware are often used to steal sensitive information or turn the infected PC into part of a larger botnet for use in spamming or attack operations.

NOTE Most antivirus products will include *antispyware* capabilities as well. Antispyware helps protect your systems from the ever-increasing flood of malware that seeks to watch your keystrokes, steal your passwords, and report sensitive information back to attackers.

Personal Software Firewalls

Personal firewalls are host-based protective mechanisms that monitor and control traffic passing into and out of a single system. Designed for the end user, software firewalls often have a configurable security policy that allows the user to determine what traffic is "good" and allowed to pass and what traffic is "bad" and is blocked. Software firewalls are extremely commonplace—so much so that most modern operating systems come with some type personal firewall included.

For example, with the introduction of the Windows XP Professional operating system, Microsoft included a utility called the Internet Connection Firewall. Though disabled by default and hidden in the network configuration screens where most users would never find it, the Internet Connection Firewall did give users some direct control over the network traffic passing through their systems. When Service Pack 2 was launched, Microsoft renamed the Internet Connection Firewall the Windows Firewall (see Figure 11-4) and enabled it by default (Vista also enables the Windows firewall by default). The Windows firewall is fairly configurable; it can be set up to block all traffic, make exceptions for traffic you want to allow, and log rejected traffic for later analysis.

With the introduction of the Vista operating system, Microsoft modified the Windows Firewall to make it more capable and configurable. More options were added to allow for more granular control of network traffic as well as the ability to detect when certain components are not behaving as expected. For example, if your MS Outlook client suddenly attempts to connect to a remote web server, the Windows Firewall can detect this as a deviation from normal behavior and block the unwanted traffic.

UNIX-based operating systems have had built-in software-based firewalls (see Figure 11-5) for a number of years including TCP wrappers, ipchains, and iptables.

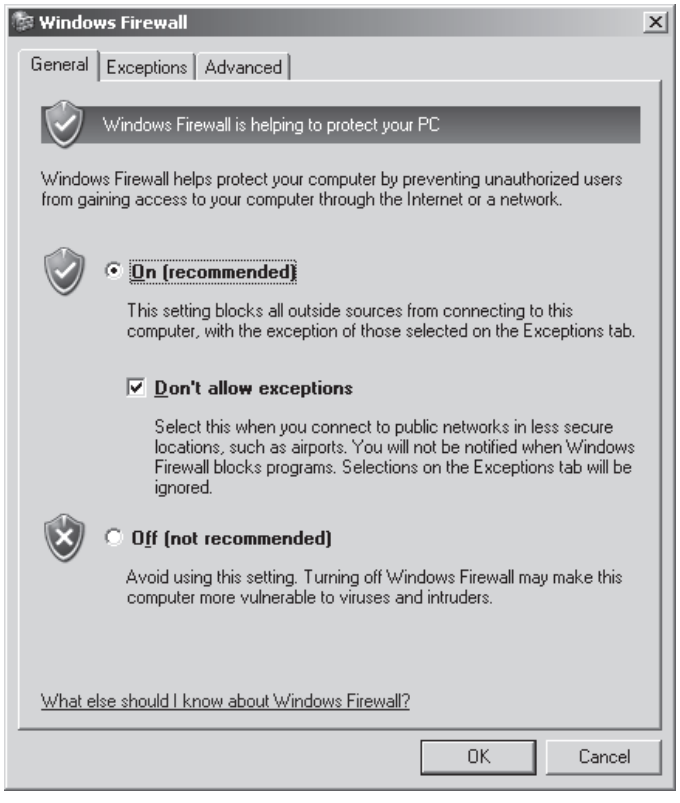


Figure 11-4 Windows Firewall is enabled by default in SP2 and Vista.

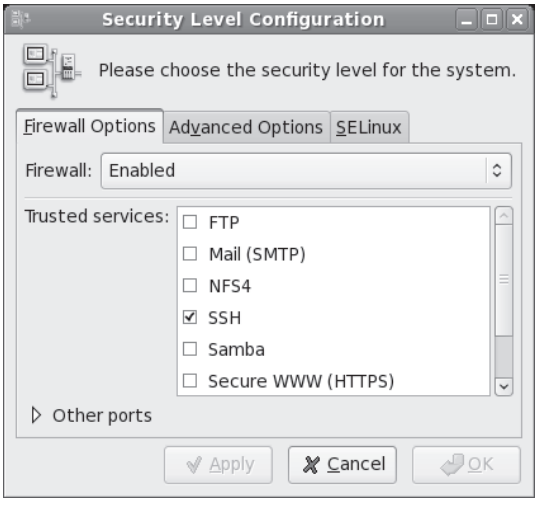


Figure 11-5 UNIX firewall

TCP Wrappers is a simple program that limits inbound network connections based on port number, domain, or IP address and is managed with two text files called hosts.allow and hosts.deny. If the inbound connection is coming from a trusted IP address and destined for a port to which it is allowed to connect, then the connection is allowed.

Ipchains is a more advanced, rule-based software firewall that allows for traffic filtering, Network Address Translation (NAT), and redirection. Three configurable "chains" are used for handling network traffic: input, output, and forward. The input chain contains rules for traffic that is coming into the local system. The output chain contains rules for traffic that is leaving the local system. The forward chain contains rules for traffic that was received by the local system but is not destined for the local system. Iptables is the latest evolution of ipchains and is designed to work with Linux kernels 2.4 and 2.6. Iptables uses the same three chains for policy rules and traffic handling as ipchains, but with iptables each packet is processed only by the appropriate chain. Under ipchains, each packet passes through all three chains for processing. With iptables, incoming packets are processed only by the input chain and packets leaving the system are processed only by the output chain. This allows for more granular control of network traffic and enhances performance.

In addition to the "free" firewalls that come bundled with operating systems, many commercial personal firewall packages are available. Programs such as ZoneAlarm from Check Point Software provide or bundle additional capabilities not found in some bundled software firewalls. Many commercial software firewalls limit inbound and outbound network traffic, block pop-ups, detect adware, block cookies, block malicious processes, and scan instant messenger traffic. While you can still purchase or even download a free software-based personal firewall, most commercial vendors are bundling the firewall functionality with additional capabilities such as antivirus and antispyware.

Pop-up Blocker

One of the most annoying nuisances associated with web browsing is the pop-up ad. Pop-up ads are online advertisements designed to attract web traffic to specific websites, capture e-mail addresses, advertise a product, and perform other tasks. If you've spent more than an hour surfing the web, you've undoubtedly seen them. They're created when the website you are visiting opens a new web browser window for the sole purpose of displaying an advertisement. Pop-up ads typically appear in front of your current browser window to catch your attention (and disrupt your browsing). Pop-up ads can range from mildly annoying, generating one or two pop-ups, to system crippling if a malicious website attempts to open thousands of pop-up windows on your system.

Similar to the pop-up ad is the pop-under ad that opens up behind your current browser window. You won't see these ads until your current window is closed, and they are considered by some to be less annoying than pop-ups. Another form of pop-up is the hover ad that uses Dynamic HTML to appear as a floating window superimposed over your browser window. Dynamic HTML can be very CPU-intensive and can have a significant impact on the performance of older systems.

To some users, pop-up ads are as undesirable as spam, and many web browsers now allow users to restrict or prevent pop-ups either built into the web browser or available as an add-on. Internet Explorer contains a built-in Pop-up Blocker (shown in Figure 11-6 and available from the Tools menu in Internet Explorer 7).

Firefox also contains a built-in pop-up blocker (available by choosing Tools | Options and then selecting the Content tab). Popular add-ons such as the Google and Yahoo! toolbars also contain pop-up blockers. If these freely available options are not enough for your needs, many commercial security suites from McAfee, Symantec, and Check Point contain pop-up blocking capabilities as well. Users must be careful when selecting a pop-up blocker, as some unscrupulous developers have created adware products disguised as free pop-up blockers or other security tools.

Pop-ups ads can be generated in a number of ways, including JavaScript and Adobe Flash, and an effective pop-up blocker must be able to deal with the many methods used to create pop-ups. When a pop-up is created, users typically can click a close or cancel button inside the pop-up or close the new window using a method available through the operating system, such as closing the window from the taskbar in Windows. With the advanced features available to them in a web development environment, some unscrupulous developers program the close or cancel buttons in their pop-ups to launch new pop-ups, redirect the user, run commands on the local system, or even load software.

NOTE Pop-ups should not be confused with adware. Pop-ups are ads that appear as you visit web pages. Adware is advertising-supported software. Adware automatically downloads and displays ads on your computer after the adware has been installed, and these ads are typically shown while the software is being used. Adware is often touted as "free" software as the user pays nothing for the software but must agree to having ads downloaded and displayed before using the software. This approach is very popular on smartphones and mobile devices.

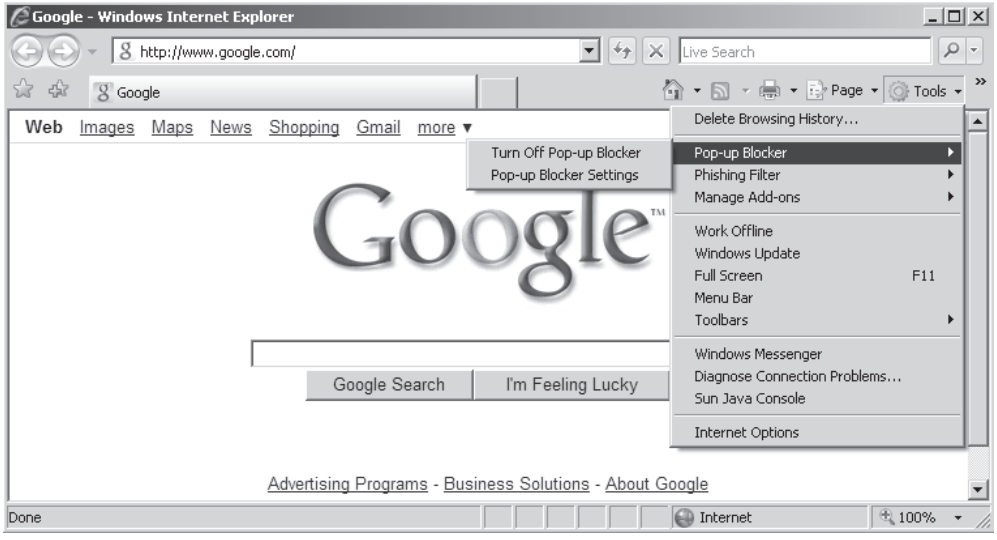


Figure 11-6 Pop-up Blocker in IE 7

PART IV

Windows Defender

As part of its ongoing efforts to help secure its PC operating systems, Microsoft created and released a free utility called Windows Defender in February 2006. The stated purpose of Windows Defender is to protect your computer from spyware and other unwanted software (http://www.microsoft.com/athome/security/spyware/software/default.mspx). Windows Defender is standard with all versions of the Vista operating system and is available via free download for Windows XP Service Pack 2 or later in both 32- and 64-bit versions. It has the following capabilities:

- **Spyware detection and removal** Windows Defender is designed to find and remove spyware and other unwanted programs that display pop-ups, modify browser or Internet settings, or steal personal information from your PC.
- **Scheduled scanning** You can schedule when you want your system to be scanned or you can run scans on demand.
- **Automatic updates** Updates to the product can be automatically downloaded and installed without user interaction.
- **Real-time protection** Processes are monitored in real time to stop spyware and malware when they first launch, attempt to install themselves, or attempt to access your PC.
- **Software Explorer** One of the more interesting capabilities within Windows Defender is the ability to examine the various programs running on your computer. Windows Defender allows you to look at programs that run automatically on startup, are currently running on your PC, or are accessing network connections on your PC. Windows Defender provides you with

details such as the publisher of the software, when it was installed on your PC, whether or not the software is "good" or considered to be known malware, the file size, publication date, and other information.

- **Configurable responses** Windows Defender (see Figure 11-7) lets you choose what actions you want to take in response to detected threats; you can automatically disable the software, quarantine it, attempt to uninstall it, and perform other tasks.

Network-based IDSs

Network-based IDSs (NIDS) came along a few years after host-based systems. After running host-based systems for a while, many organizations grew tired of the time, energy, and expense involved with managing the first generation of these systems. The desire for a "better way" grew along with the amount of interconnectivity between systems and consequently the amount of malicious activity coming across the networks themselves. This fueled development of a new breed of IDS designed to focus on the source for a great deal of the malicious traffic—the network itself.

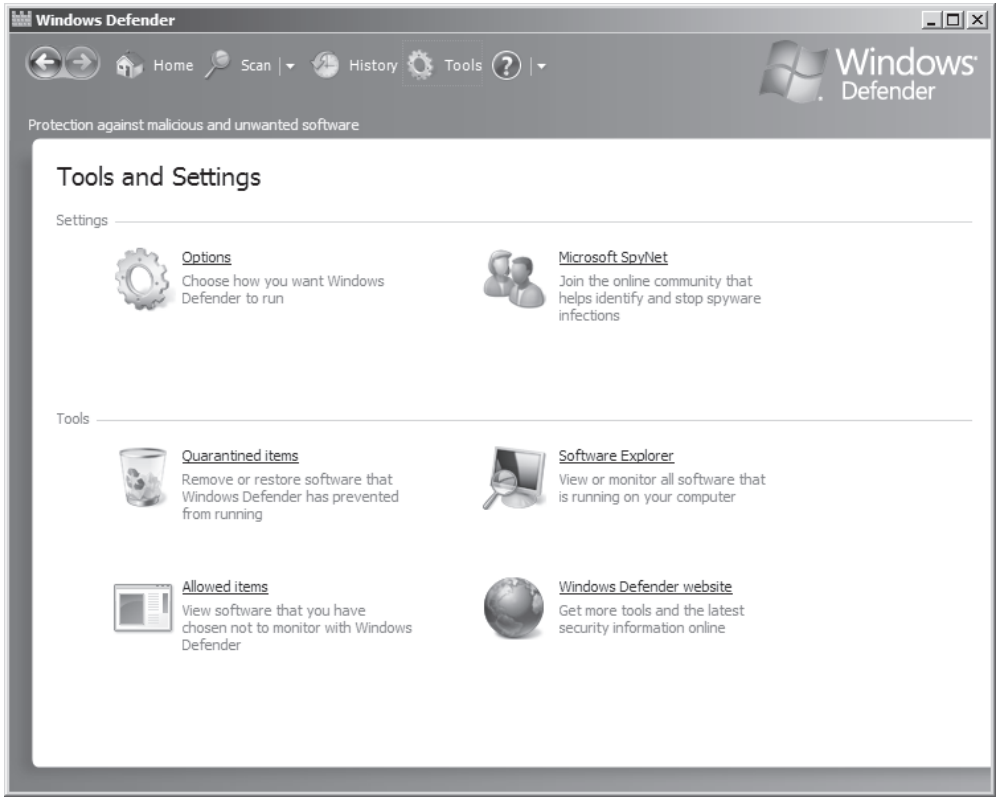


Figure 11-7 Windows Defender configuration options

The NIDS integrated very well into the concept of *perimeter security*. More and more companies began to operate their computer security like a castle or military base with attention and effort focused on securing and controlling the ways in and out—the idea being that if you could restrict and control access at the perimeter, you didn't have to worry as much about activity inside the organization. Even though the idea of a security perimeter is somewhat flawed (many security incidents originate inside the perimeter), it caught on very quickly, as it was easy to understand and devices such as firewalls, bastion hosts, and routers were available to define and secure that perimeter. The best way to secure the perimeter from outside attack is to reject all traffic from external entities, but as this is impossible and impractical to do, security personnel needed a way to let traffic in but still be able to determine whether or not the traffic was malicious. This is the problem that NIDS developers were trying to solve.

As its name suggests, a NIDS focuses on network traffic—the bits and bytes traveling along the cables and wires that interconnect the systems. A NIDS must examine the network traffic as it passes by and be able to analyze traffic according to protocol, type, amount, source, destination, content, traffic already seen, and other factors. This analysis must happen quickly, and the NIDS must be able to handle traffic at whatever speed the network operates to be effective.

NIDSs are typically deployed so that they can monitor traffic in and out of an organization's major links: connections to the Internet, remote offices, partners, and so on. Like host-based systems, NIDSs look for certain activities that typify hostile actions or misuse, such as the following:

- Denial-of-service attacks
- Port scans or sweeps
- Malicious content in the data payload of a packet or packets
- Vulnerability scanning
- Trojans, viruses, or worms
- Tunneling
- Brute-force attacks

In general, most NIDSs operate in a fairly similar fashion. Figure 11-8 shows the logical layout of a NIDS. By considering the function and activity of each component, you can gain some insight into how NIDS operate.

As you can see, the logical components of a NIDS are very similar to those of the host-based system. In the simplest form, a NIDS has the same major components: traffic collector, analysis engine, reports, and a user interface.

In a NIDS, the *traffic collector* is specifically designed to pull traffic from the network. This component usually behaves in much the same way as a network traffic sniffer—it simply pulls every packet it can see off the network to which it is connected. In a NIDS, the traffic collector will logically attach itself to a network interface card (NIC) and instruct the NIC to accept every packet it can. A NIC that accepts and processes every packet regardless of the packet's origin and destination is said to be in *promiscuous mode*.

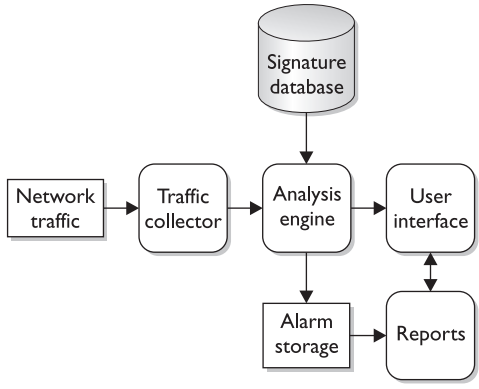


Figure 11-8 Network IDS components

The *analysis engine* in a NIDS serves the same function as its host-based counterpart, with some substantial differences. The network analysis engine must be able to collect packets and examine them individually or, if necessary, reassemble them into an entire traffic session. The patterns and signatures being matched are far more complicated than host-based signatures, so the analysis engine must be able to remember what traffic preceded the traffic currently being analyzed so that it can determine whether or not that traffic fits into a larger pattern of malicious activity. Additionally, the network-based analysis engine must be able to keep up with the flow of traffic on the network, rebuilding network sessions and matching patterns in real time.

The NIDS *signature database* is usually much larger than that of a host-based system. When examining network patterns, the IDS must be able to recognize traffic targeted at many different applications and operating systems as well as traffic from a wide variety of threats (worms, assessment tools, attack tools, and so on). Some of the signatures themselves can be quite large, as the NIDS must look at network traffic occurring in a specific order over a period of time to match a particular malicious pattern.

Using the lessons learned from the early host-based systems, NIDS developers modified the logical component design somewhat to distribute the user interface and reporting functions. As many companies had more than one network link, they would need an IDS capable of handling multiple links in many different locations. The early IDS vendors solved this dilemma by dividing the components and assigning them to separate entities. The traffic collection, analysis engine, and signature database were bundled into a single entity usually called a *sensor* or *appliance*. The sensors would report to and be controlled by a central system or master console. This central system, shown in Figure 11-9, consolidated alarms and provided the user interface and reporting functions that allowed users in one location to manage, maintain, and monitor sensors deployed in a variety of remote locations.

By creating separate entities designed to work together, the network IDS developers were able to build a more capable and flexible system. With encrypted communications, network sensors could be placed around both local and remote perimeters and still be monitored and managed securely from a central location. Placement of the sen-

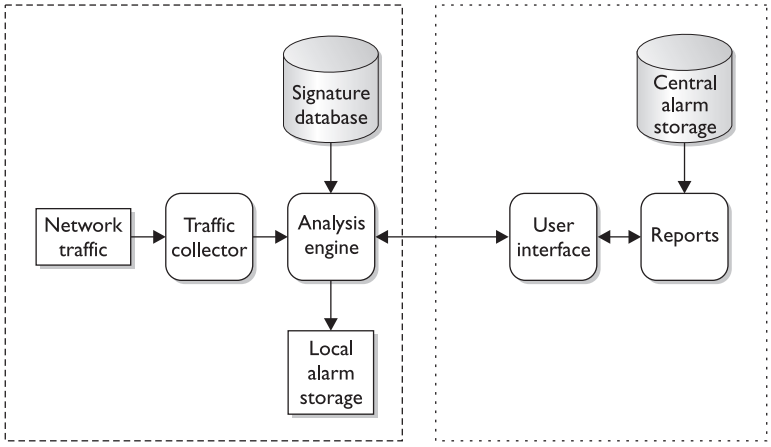


Figure 11-9 Distributed network IDS components

sors very quickly became an issue for most security personnel, as the sensors obviously had to have visibility of the network traffic in order to analyze it. Because most organizations with network-based IDSs also had firewalls, location of the IDS relative to the firewall had to be considered as well. Placed before the firewall, as shown in Figure 11-10, the IDS will see all traffic coming in from the Internet, including attacks against the firewall itself. This includes traffic that the firewall stops and does not permit into the corporate network. With this type of deployment, the network IDS sensor will generate a large number of alarms (including alarms for traffic that the firewall would stop) that tends to overwhelm the human operators managing the system.

Placed after the firewall, as shown in Figure 11-11, the NIDS sensor sees and analyzes the traffic that is being passed through the firewall and into the corporate network. While this does not allow the NIDS to see attacks against the firewall, it generally results in far fewer alarms and is the most popular placement for NIDS sensors.

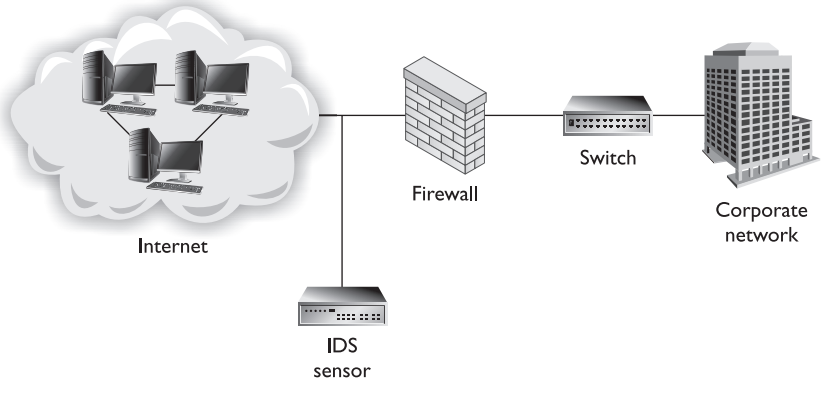


Figure 11-10 IDS sensor placed in front of firewall

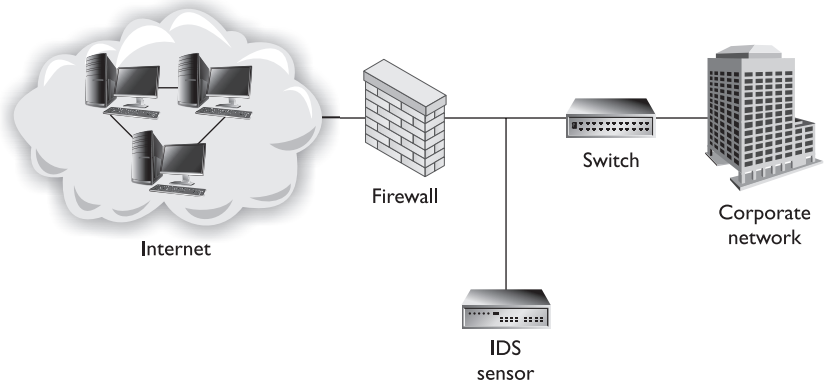


Figure 11-11 IDS sensor placed behind firewall

Another possible location for a NIDS is in front of or inside a *DMZ*. A DMZ (or demilitarized zone) is a physical or logical subnetwork where public services can be exposed to the Internet. Public services such as web servers and mail servers are placed inside a DMZ and external connections are allowed into the DMZ—but not allowed to continue through to the corporate network. Housing public services within a DMZ reduces the risk of compromise for other critical assets—if a public service in the DMZ is compromised, the damage is contained, as traffic is not allowed to pass from the DMZ back into the corporate network. Due to its very nature, a DMZ is a heavily targeted area, making it a natural location for a NIDS.

As you already know, NIDSs examine the network traffic for suspicious or malicious activity. Here are two examples to illustrate the operation of a NIDS:

- **Port scan** A port scan is a reconnaissance activity a potential attacker will use to find out information about the systems he wants to attack. Using any of a number of tools, the attacker will attempt to connect to various services (Web, FTP, SMTP, and so on) to see if they exist on the intended target. In normal network traffic, a single user might connect to the FTP service provided on a single system. During a port scan, an attacker may attempt to connect to the FTP service on every system. As the attacker's traffic passes by the IDS, this pattern of attempting to connect to different services on different systems will be noticed. When the IDS compares the activity to its signature database, it will very likely match this traffic against the port scanning signature and generate an alarm.
- **Ping of death** Toward the end of 1996, it was discovered that certain operating systems, such as Windows, could be crashed by sending a very large Internet Control Message Protocol (ICMP) echo request packet to that system. The vulnerable operating systems did not handle the packet correctly and would subsequently reboot or lock up after receiving the packets. This is a fairly simple traffic pattern for a NIDS to identify, as it simply has to look for ICMP packets over a certain size.

Advantages of a NIDS

A NIDS has certain advantages that make it a good choice for certain situations:

- *It takes fewer systems to provide IDS coverage.* With a few well-placed NIDS sensors, you can monitor all the network traffic going in and out of your organization. Fewer sensors usually equates to less overhead and maintenance, meaning you can protect the same number of systems at a lower cost.
- *Deployment, maintenance, and upgrade costs are usually lower.* The fewer systems that have to be managed and maintained to provide IDS coverage, the lower the cost to operate the IDS. Upgrading and maintaining a few sensors is usually much cheaper than upgrading and maintaining hundreds of host-based processes.
- *A NIDS has visibility into all network traffic and can correlate attacks among multiple systems.* Well-placed NIDS sensors can see the "big picture" when it comes to network-based attacks. The network sensors can tell you whether attacks are widespread and unorganized or focused and concentrated on specific systems.

Disadvantages of a NIDS

A NIDS has certain disadvantages:

- *It is ineffective when traffic is encrypted.* When network traffic is encrypted from application to application or system to system, a NIDS sensor will not be able to examine that traffic. With the increasing popularity of encrypted traffic, this is becoming a bigger problem for effective IDS operations.
- *It can't see traffic that does not cross it.* The IDS sensor can examine only traffic crossing the network link it is monitoring. With most IDS sensors being placed on perimeter links, traffic traversing the internal network is never seen.
- *It must be able to handle high volumes of traffic.* As network speeds continue to increase, the network sensors must be able to keep pace and examine the traffic as quickly as it can pass the network. When NIDSs were introduced, 10 Mbps networks were the norm. Now 100 Mbps and even 1 Gbps networks are commonplace. This increase in traffic speeds means IDS sensors must be faster and more powerful than ever before.
- *It doesn't know about activity on the hosts themselves.* NIDSs focus on network traffic. Activity that occurs on the hosts themselves will not be seen by a NIDS.

Active vs. Passive NIDSs

Most NIDSs can be distinguished by how they examine the traffic and whether or not they interact with that traffic. On a *passive* system, the IDS simply watches the traffic, analyzes it, and generates alarms. It does not interact with the traffic itself in any way, and it does not modify the defensive posture of the system to react to the traffic. A

passive IDS is very similar to a simple motion sensor—it generates an alarm when it matches a pattern much as the motion sensor generates an alarm when it sees movement. An *active* IDS will contain all the same components and capabilities of the passive IDS with one critical addition—the active IDS can *react* to the traffic it is analyzing. These reactions can range from something simple, such as sending a TCP reset message to interrupt a potential attack and disconnect a session, to something complex, such as dynamically modifying firewall rules to reject all traffic from specific source IP addresses for the next 24 hours.

The most common defensive ability for an active IDS is to send a TCP reset message. Within TCP, the reset message (RST) essentially tells both sides of the connection to drop the session and stop communicating immediately. While this mechanism was originally developed to cover situations such as systems accidentally receiving communications intended for other systems, the reset message works fairly well for IDSs—with one serious drawback: a reset message affects only the current session. Nothing prevents the attacker from coming back and trying again and again. Despite the "temporariness" of this solution, sending a reset message is usually the only defensive measure implemented on IDS deployments, as the fear of blocking legitimate traffic and disrupting business processes, even for a few moments, often outweighs the perceived benefit of discouraging potential intruders.

Signatures

As you have probably deduced from the discussion so far, one of the critical elements of any good IDS is the signature set—the set of patterns the IDS uses to determine whether or not activity is potentially hostile. Signatures can be very simple or remarkably complicated, depending on the activity they are trying to highlight. In general, signatures can be divided into two main groups, depending on what the signature is looking for: content-based and context-based.

Content-based signatures are generally the simplest. They are designed to examine the content of such things as network packets or log entries. Content-based signatures are typically easy to build and look for simple things, such as a certain string of characters or a certain flag set in a TCP packet. Here are some examples of content-based signatures:

- *Matching the characters* /etc/passwd *in a Telnet session*. On a UNIX system, the names of valid user accounts (and sometimes the passwords for those user accounts) are stored in a file called *passwd* located in the *etc* directory.
- *Matching a TCP packet with the synchronize, reset, and urgent flags all set within the same packet.* This combination of flags is impossible to generate under normal conditions, and the presence of all of these flags in the same packet would indicate this packet was likely created by a potential attacker for a specific purpose, such as to crash the targeted system.
- *Matching the characters* to: decode *in the header of an e-mail message.* On certain older versions of sendmail, sending an e-mail message to "decode" would cause the system to execute the contents of the e-mail.

Context-based signatures are generally more complicated, as they are designed to match large patterns of activity and examine how certain types of activity fit into the other activities going on around them. Context signatures generally address the question How does this event compare to other events that have already happened or might happen in the near future? Context-based signatures are more difficult to analyze and take more resources to match, as the IDS must be able to "remember" past events to match certain context signatures. Here are some examples of context-based signatures:

- *Match a potential intruder scanning for open web servers on a specific network.* A potential intruder may use a port scanner to look for any systems accepting connections on port 80. To match this signature, the IDS must analyze all attempted connections to port 80 and then be able to determine which connection attempts are coming from the same source but are going to multiple, different destinations.
- *Identify a Nessus scan.* Nessus is an open-source vulnerability scanner that allows security administrators (and potential attackers) to quickly examine systems for vulnerabilities. Depending on the tests chosen, Nessus will typically perform the tests in a certain order, one after the other. To be able to determine the presence of a Nessus scan, the IDS must know which tests Nessus runs as well as the typical order in which the tests are run.
- *Identify a ping flood attack.* A single ICMP packet on its own is generally regarded as harmless, certainly not worthy of an IDS signature. Yet thousands of ICMP packets coming to a single system in a short period of time can have a devastating effect on the receiving system. By flooding a system with thousands of valid ICMP packets, an attacker can keep a target system so busy it doesn't have time to do anything else—a very effective denial-of-service attack. To identify a ping flood, the IDS must recognize each ICMP packet and keep track of how many ICMP packets different systems have received in the recent past.

EXAM TIP Know the differences between content-based and context-based signatures. Content-based signatures match specific content such as a certain string or series of characters (matching the string */etc/passwd* in an FTP session). Context-based signatures match a pattern of activity based on the other activity around it (such as a port scan).

To function, the IDS must have a decent signature base with examples of known, undesirable activity that it can use when analyzing traffic or events. Any time an IDS matches current events against a signature, the IDS could be considered successful, as it has correctly matched the current event against a known signature and reacted accordingly (usually with an alarm or alert of some type).

False Positives and Negatives

Viewed in its simplest form, an IDS is really just looking at activity (be it host-based or network-based) and matching it against a predefined set of patterns. When it matches an activity to a specific pattern, the IDS cannot know the true intent behind that activity—whether or not it is benign or hostile—and therefore it can react only as it has been programmed to do. In most cases, this means generating an alert that must then be analyzed by a human who tries to determine the intent of the traffic from whatever information is available. When an IDS matches a pattern and generates an alarm for benign traffic, meaning the traffic was not hostile and not a threat, this is called a *false positive*. In other words, the IDS matched a pattern and raised an alarm when it didn't really need to do so. Keep in mind that the IDS can only match patterns and has no ability to determine intent behind the activity, so in some ways this is an unfair label. Technically, the IDS is functioning correctly by matching the pattern, but from a human standpoint this is not information the analyst needed to see, as it does not constitute a threat and does not require intervention.

An IDS is also limited by its signature set—it can match only activity for which it has stored patterns. Hostile activity that does not match an IDS signature and therefore goes undetected is called a *false negative*. In this case, the IDS is not generating any alarms, even though it should be, giving a false sense of security.

IDS Models

In addition to being divided along the host and network lines, IDSs are often classified according to the detection model they use: anomaly or misuse. For an IDS, a model is a method for examining behavior so that the IDS can determine whether that behavior is "not normal" or in violation of established policies.

An *anomaly* detection model is the more complicated of the two. In this model, the IDS must know what "normal" behavior on the host or network being protected really is. Once the "normal" behavior baseline is established, the IDS can then go to work identifying deviations from the norm, which are further scrutinized to determine whether that activity is malicious. Building the profile of normal activity is usually done by the IDS, with some input from security administrators, and can take days to months. The IDS must be flexible and capable enough to account for things such as new systems, new users, movement of information resources, and other factors, but be sensitive enough to detect a single user illegally switching from one account to another at 3 A.M. on a Saturday.

Anomaly detection was developed to make the system capable of dealing with variations in traffic and better able to determine which activity patterns were malicious. A perfectly functioning anomaly-based system would be able to ignore patterns from legitimate hosts and users but still identify those patterns as suspicious should they come from a potential attacker. Unfortunately, most anomaly-based systems suffer from extremely high numbers of false positives, especially during the "break-in" period while

the IDS is learning the network. On the other hand, an anomaly-based system is not restricted to a specific signature set and is far more likely to identify a new exploit or attack tool that would go unnoticed by a traditional IDS.

EXAM TIP Anomaly detection looks for things that are out of the ordinary, such as a user logging in when he's not supposed to or unusually high network traffic into and out of a workstation.

A misuse detection model is a little simpler to implement, and therefore it's the more popular of the two models. In a misuse model, the IDS looks for suspicious activity or activity that violates specific policies and then reacts as it has been programmed to do. This reaction can be an alarm, e-mail, router reconfiguration, or TCP reset message. Technically, misuse is the more efficient model, as it takes fewer resources to operate, does not need to learn what "normal" behavior is, and will generate an alarm whenever a pattern is successfully matched. However, the misuse model's greatest weakness is its reliance on a predefined signature base—any activity, malicious or otherwise, that the misuse-based IDS does not have a signature for will go undetected. Despite that drawback and because it is easier and cheaper to implement, most commercial IDS products are based on the misuse detection model.

Some analysts break IDS models down even further into four categories depending on how the IDS operates and detects malicious traffic (the same models can also be applied to Intrusion Prevention Systems as well):

- **Behavior-based** This model relies on a collected set of "normal behavior"—what should happen on the network and is considered "normal" or "acceptable" traffic. Behavior that does not fit into the normal activity categories or patterns is considered suspicious or malicious. This model can potentially detect zero-day or unpublished attacks but carries a high false positive rate because any new traffic pattern can be labeled as "suspect."
- **Signature-based** This model relies on a predefined set of patterns (called *signatures*). The IDS has to know what behavior is considered "bad" ahead of time before it can identify and act upon suspicious or malicious traffic.
- **Anomaly-based** This model is essentially the same as behavior-based. The IDS is first taught what normal traffic looks like and then looks for deviations from those normal patterns.
- **Heuristic** This model uses artificial intelligence to detect intrusions and malicious traffic. This is typically implemented through algorithms that help an IDS decide if a traffic pattern is malicious or not. For example, a URL containing a character repeated 10 times may be considered "bad" traffic as a single signature. With a heuristic model the IDS will understand that if 10 repeating characters is bad, 11 is still bad, and 20 is even worse. This implementation of fuzzy logic allows this model to fall somewhere between signature-based and behavior-based models.

Intrusion Prevention Systems

An intrusion prevention system (IPS) monitors network traffic for malicious or unwanted behavior and can block, reject, or redirect that traffic in real time. Sound familiar? It should: While many vendors will argue that an IPS is a different animal from an IDS, the truth is that most IPSs are merely expansions of existing IDS capabilities. As a core function, an IPS must be able to monitor for and detect potentially malicious network traffic, which is essentially the same function as an IDS. However, an IPS does not stop at merely monitoring traffic—it must be able to block, reject, or redirect that traffic in real time to be considered a true IPS. It must be able to stop or prevent malicious traffic from having an impact. To qualify as an IDS a system just needs to see and classify the traffic as malicious. To qualify as an IPS, the system must be able to do something about that traffic. In reality, most products that are called IDSs, including the first commercially available IDS, NetRanger, can interact with and stop malicious traffic, so the distinction between the two is often blurred. The term *intrusion prevention system* was originally coined by Andrew Plato in marketing literature developed for NetworkICE, a company that was purchased by ISS and which is now part of IBM.

EXAM TIP IPSs are often divided into Network-based Intrusion Prevention Systems (NIPSs) and Host-based Intrusion Prevention Systems (HIPSs) depending on whether they're protecting an entire network or a single host. A NIPS is essentially a network IDS with the ability to stop/block malicious traffic. A HIPS is essentially a host-based IDS with the ability to stop or block malicious traffic.

Like IDSs, most IPSs have an internal signature base to compare network traffic against known "bad" traffic patterns. IPSs can perform content-based inspections, looking inside network packets for unique packets, data values, or patterns that match known malicious patterns. Some IPSs can perform protocol inspection, in which the IPS decodes traffic and analyzes it as it would appear to the server receiving it. For example, many IPSs can do HTTP protocol inspection, so they can examine incoming and outgoing HTTP traffic and process it as an HTTP server would. The advantage here is that the IPS can detect and defeat popular evasion techniques such as encoding URLs as the IPS "sees" the traffic in the same way the web server would when it receives and decodes it. The IPS can also detect activity that is abnormal or potentially malicious for that protocol, such as passing an extremely large value (over 10,000 characters) to a login field on a web page.

Unlike a traditional IDS, an IPS must sit in line to be able to interact effectively with the network traffic. Most IPSs can operate in "stealth mode" and do not require an IP address for the connections they are monitoring. When an IPS detects malicious traffic, it can drop the offending packets, reset incoming or established connections, generate alerts, quarantine traffic to/from specific IP addresses, or even block traffic from offending IP addresses on a temporary or permanent basis. As they are sitting in line, most IPSs can also offer *rate-based monitoring* to detect and mitigate denial-of-service attacks. With rate-based monitoring, the IPS can watch the amount of traffic traversing the network. If the IPS sees too much traffic coming into or going out from a specific system

or set of systems, the IPS can intervene and throttle down the traffic to a lower and more acceptable level. Many IPSs perform this function by "learning" what are "normal" network traffic patterns with regard to number of connections per second, amount of packets per connection, packets coming from or going to specific ports, and so on, and comparing current traffic rates for network traffic (TCP, UDP, ARP, ICMP, and so on) to those established norms. When a traffic pattern reaches a threshold or varies dramatically from those norms, the IPS can react and intervene as needed.

Like a traditional IDS, the IPS has a potential weakness when dealing with encrypted traffic. Traffic that is encrypted will typically pass by the IPS untouched (provided it does not trigger any non-content–related alarms such as rate-based alarms). To counter this problem, some IPS vendors are including the ability to decrypt Secure Sockets Layer (SSL) sessions for further inspection. To do this, some IPS solutions will store copies of any protected web servers' private keys on the sensor itself. When the IPS sees a session initiation request, it monitors the initial transactions between the server and the client. By using the server's stored private keys, the IPS will be able to determine the session keys negotiated during the SSL session initiation. With the session keys, the IPS can decrypt all future packets passed between server and client during that web session. This gives the IPS the ability to perform content inspection on SSL-encrypted traffic.

You will often see IPS (and IDS) advertised and marketed by their wire speed or amount of traffic they can process without dropping packets or interrupting the flow of network traffic. The term *wire speed* refers to the theoretical maximum transmission rate of a cable or other medium and is based on a number of factors, including the properties of the cable itself and the connection protocol in use (in other words, how much data can be pushed through under ideal conditions). In reality, a network will never reach its hypothetical maximum transmission rate due to errors, collisions, retransmissions, and other factors; therefore, a 1 Gbps network is not actually capable of passing an actual 1 Gbps of network traffic, even if all the components are rated to handle 1 Gbps. When used in a marketing sense, wire speed is the maximum throughput rate networking or security device equipment can process without impacting that network traffic. For example, a 1 Gbps IPS should be able to process, analyze, and protect 1 Gbps of network traffic without impacting traffic flow. IPS vendors will often quote their products' capacity as the combined throughput possible through all available ports on the IPS sensor—a 10 Gbps sensor may have 12-gigabit Ethernet ports but is capable of handling only 10 Gbps of network traffic.

Detection Controls vs. Prevention Controls

When securing your organizations, especially your network perimeter and critical systems, you will likely have to make some choices as to what type of protective measures and controls you need to implement. For example, you may need to decide between *detection controls* (capabilities that detect and alert on suspicious or malicious activity) and *prevention controls* (capabilities that stop suspicious or malicious activity). Consider the differences between a traditional IDS and IPS. While many IDSs have some type of response capability, their real purpose is to watch for activity and then alert when "hostile" activity is noted. On the other hand, an IPS is designed to block, thwart, and prevent that same "hostile" activity.

A parallel example in the physical security space would be a camera and a security guard. A camera watches activity and can even generate alerts when motion is detected. But a camera cannot stop an intruder from breaking into a facility and stealing—it can only record and alert. A security guard, however, has the ability to physically stop intruders either before they break into the facility or before they can leave with the stolen goods.

Honeypots and Honeynets

As is often the case, one of the best tools for information security personnel has always been knowledge. To secure and defend a network and the information systems on that network properly, security personnel need to know what they are up against. What types of attacks are being used? What tools and techniques are popular at the moment? How effective is a certain technique? What sort of impact will this tool have on my network? Often this sort of information is passed through white papers, conferences, mailing lists, or even word of mouth. In some cases, the tool developers themselves provide much of the information in the interest of promoting better security for everyone.

Information is also gathered through examination and forensic analysis, often after a major incident has already occurred and information systems are already damaged. One of the most effective techniques for collecting this type of information is to observe activity first-hand—watching an attacker as she probes, navigates, and exploits his way through a network. To accomplish this without exposing critical information systems, security researchers often use something called a *honeypot*.

A honeypot, sometimes called a *digital sandbox*, is an artificial environment where attackers can be contained and observed without putting real systems at risk. A good honeypot appears to an attacker to be a real network consisting of application servers, user systems, network traffic, and so on, but in most cases it's actually made up of one or a few systems running specialized software to simulate the user and network traffic common to most targeted networks. Figure 11-12 illustrates a simple honeypot layout in which a single system is placed on the network to deliberately attract attention from potential attackers.

Figure 11-12 shows the security researcher's view of the honeypot, while Figure 11-13 shows the attacker's view. The security administrator knows that the honeypot, in this case, actually consists of a single system running software designed to react to probes, reconnaissance attempts, and exploits as if it were an entire network of systems. When the attacker connects to the honeypot, she is presented with an entire "virtual" network of servers and PCs running a variety of applications. In most cases, the honeypot will appear to be running versions of applications that are known to be vulnerable to specific exploits. All this is designed to provide the attacker with an enticing, hopefully irresistible, target.

Any time an attacker has been lured into probing or attacking the virtual network, the honeypot records the activity for later analysis: what the attacker does, which systems and applications she concentrates on, what tools are run, how long the attacker stays, and so on. All this information is collected and analyzed in the hopes that it will allow security personnel to better understand and protect against the threats to their systems.

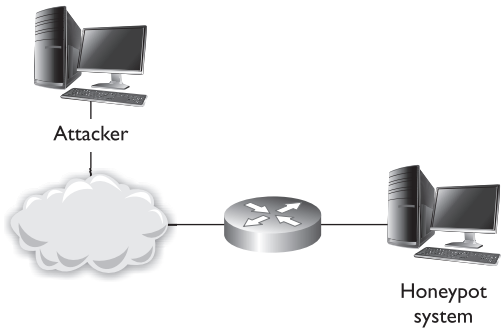


Figure 11-12 Logical depiction of a honeypot

There are many honeypots in use, specializing in everything from wireless to denial-of-service attacks; most are run by research, government, or law enforcement organizations. Why aren't more businesses running honeypots? Quite simply, the time and cost are prohibitive. Honeypots take a lot of time and effort to manage and maintain and even more effort to sort, analyze, and classify the traffic the honeypot collects. Unless they are developing security tools, most companies focus their limited security efforts on preventing attacks, and in many cases, companies aren't even that concerned with detecting attacks as long as the attacks are blocked, are unsuccessful, and don't affect business operations. Even though honeypots can serve as a valuable resource by luring attackers away from production systems and allowing defenders to identify and thwart potential attackers before they cause any serious damage, the costs and efforts involved deter many companies from using honeypots.

EXAM TIP A honeypot is a system designed to attract potential attackers by pretending to be one or more systems with open network services.

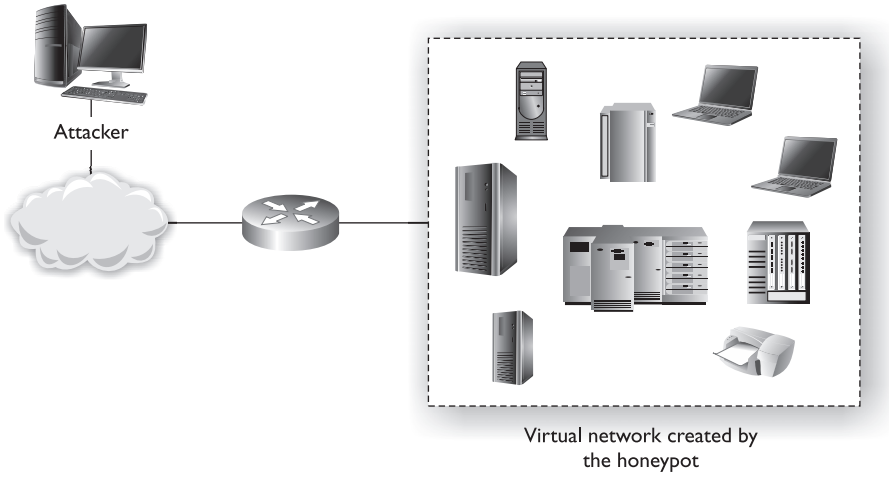


Figure 11-13 Virtual network created by the honeypot

A *honeynet* is a collection of two or more honeypots. Larger, very diverse network environments can deploy multiple honeypots (thus forming a honeynet) when a single honeypot device does not provide enough coverage. Honeynets are often integrated into an organization-wide IDS/IPS as the honeynet can provide relevant information about potential attackers.

Firewalls

Arguably one of the first and most important network security tools is the firewall. A *firewall* is a device that is configured to permit or deny network traffic based on an established policy or rule set. In their simplest form, firewalls are like network traffic cops; they determine which packets are allowed to pass into or out of the network perimeter. The term *firewall* was borrowed from the construction field, in which a fire wall is literally a wall meant to confine a fire or prevent a fire's spread within or between buildings. In the network security world, a firewall stops the malicious and untrusted traffic (the fire) of the Internet from spreading into your network. Firewalls control traffic flow between zones of network traffic; for example, between the Internet (a zone with no trust) and an internal network (a zone with high trust). (Personal software firewalls were already discussed in this chapter; for more discussion on network firewalls refer to Chapter 8.)

EXAM TIP Many firewalls contain by default an *implicit* deny at the end of every ACL or firewall rule set. This simply means that any traffic not specifically permitted by a previous rule in the rule set is denied.

Web Application Firewalls vs. Network Firewalls

Increasingly, the term "firewall" is being attached to any device or software package that is used to control the flow of packets or data into or out of an organization. For example, a *web application firewall* is the term given to any software package, appliance, or filter that applies a rule set to HTTP/HTTPS traffic. Web application firewalls shape web traffic and can be used to filter out SQL injection attacks, malware, Cross Site Scripting, and so on. By contrast, a network firewall is a hardware or software package that controls the flow of packets into and out of a network. Web application firewalls operate on traffic at a much higher level than network firewalls as web application firewalls must be able to decode the web traffic to determine whether or not it is malicious. Network firewalls operate on much simpler aspects of network traffic such as source/destination port and source/destination address.

Proxy Servers

Though not strictly a security tool, a *proxy server* can be used to filter out undesirable traffic and prevent employees from accessing potentially hostile web sites. A proxy server takes requests from a client system and forwards them to the destination server on behalf of the client. Proxy servers can be completely transparent (these are usually

called *gateways* or *tunneling proxies*), or a proxy server can modify the client request before sending it on or even serve the client's request without needing to contact the destination server. Several major categories of proxy servers are in use:

- **Anonymizing proxy** An anonymizing proxy is designed to hide information about the requesting system and make a user's web browsing experience "anonymous." This type of proxy service is often used by individuals concerned with the amount of personal information being transferred across the Internet and the use of tracking cookies and other mechanisms to track browsing activity.
- **Caching proxy** This type of proxy keeps local copies of popular client requests and is often used in large organizations to reduce bandwidth usage and increase performance. When a request is made, the proxy server first checks to see whether it has a current copy of the requested content in the cache; if it does, it services the client request immediately without having to contact the destination server. If the content is old or the caching proxy does not have a copy of the requested content, the request is forwarded to the destination server.
- **Content filtering proxy** Content filtering proxies examine each client request and compare it to an established acceptable use policy. Requests can usually be filtered in a variety of ways including the requested URL, destination system, or domain name or by keywords in the content itself. Content filtering proxies typically support user-level authentication so access can be controlled and monitored and activity through the proxy can be logged and analyzed. This type of proxy is very popular in schools, corporate environments, and government networks.
- **Open proxy** An open proxy is essentially a proxy that is available to any Internet user and often has some anonymizing capabilities as well. This type of proxy has been the subject of some controversy with advocates for Internet privacy and freedom on one side of the argument, and law enforcement, corporations, and government entities on the other side. As open proxies are often used to circumvent corporate proxies, many corporations attempt to block the use of open proxies by their employees.
- **Reverse proxy** A reverse proxy is typically installed on the server side of a network connection, often in front of a group of web servers. The reverse proxy intercepts all incoming web requests and can perform a number of functions including traffic filtering, SSL decryption, serving of common static content such as graphics, and performing load balancing.
- **Web proxy** A web proxy is solely designed to handle web traffic and is sometimes called a *web cache*. Most web proxies are essentially specialized caching proxies.

Deploying a proxy solution within a network environment is usually done by either setting up the proxy and requiring all client systems to configure their browsers to use the proxy or by deploying an intercepting proxy that actively intercepts all requests without requiring client-side configuration.

From a security perspective, proxies are most useful in their ability to control and filter outbound requests. By limiting the types of content and websites employees can access from corporate systems, many administrators hope to avoid loss of corporate data, hijacked systems, and infections from malicious websites. Administrators also use proxies to enforce corporate acceptable use policies and track use of corporate resources.

Internet Content Filters

With the dramatic proliferation of Internet traffic and the push to provide Internet access to every desktop, many corporations have implemented content-filtering systems to protect them from employees' viewing of inappropriate or illegal content at the workplace and the subsequent complications that occur when such viewing takes place. Internet content filtering (or content inspection) is also popular in schools, libraries, homes, government offices, and any other environment where there is a need to limit or restrict access to undesirable content. In addition to filtering undesirable content, such as pornography, some content filters can also filter out malicious activity such as browser hijacking attempts or cross-site–scripting attacks. In many cases, content filtering is performed with or as a part of a proxy solution as the content requests can be filtered and serviced by the same device. Content can be filtered in a variety of ways, including via the requested URL (called *URL filtering*), the destination system, the domain name, by keywords in the content itself, and by type of file requested.

In addition to the ability to filter out malicious or undesirable URLs, most content-filtering systems have the ability to scan for and filter out malware inside HTTP traffic—such as adware, scripting attacks, hostile applets, and so on. Unfortunately most malware filters still rely on some type of signature-based detection mechanism that must be managed and kept up to date to provide effective protection. Some of the more effective filters provide the ability to block entire classes of potentially malicious HTTP traffic (such as JavaScript), but these types of large-scale restrictions can also restrict the functionality of legitimate sites.

Content-filtering systems face many challenges, because the ever-changing Internet makes it difficult to maintain lists of undesirable sites (sometimes called black lists); terms used on a medical site can also be used on a pornographic site, making keyword filtering challenging; and determined users are always seeking ways to bypass proxy filters. To help administrators, most commercial content-filtering solutions provide an update service, much like IDS or antivirus products, that updates keywords and undesirable sites automatically.

Web Security Gateway

Some security vendors combine proxy functions with content-filtering functions to create a product called a *web security gateway*. Web security gateways are intended to address the security threats and pitfalls unique to web-based traffic. Web security gateways typically provide the following capabilities:

- Real-time Malware Protection (also known as malware inspection) Some web security gateways have the ability to scan all outgoing and incoming web traffic to detect and block undesirable traffic such as malware, spyware, adware, malicious scripts, file-based attacks, and so on.
- Content monitoring Some web security gateways provide the ability to monitor the content of web traffic being examined to ensure that it complies with organizational policies.
- Productivity monitoring Some web security gateways measure how much web traffic is being generated by specific users, groups of users, or the entire organization as well as the types of traffic being generated.
- Data protection and compliance Some web security gateways can scan web traffic for sensitive or proprietary information being sent outside of the organization as well as the use of social network or inappropriate sites.

Protocol Analyzers

A *protocol analyzer* (also known as a *packet sniffer*, *network analyzer*, or *network sniffer*) is a piece of software or an integrated software/hardware system that can capture and decode network traffic. Protocol analyzers have been popular with system administrators and security professionals for decades because they are such versatile and useful tools for a network environment. From a security perspective, protocol analyzers can be used for a number of activities, such as the following:

- Detecting intrusions or undesirable traffic (IDS/IPS must have some type of capture and decode ability to be able to look for suspicious/malicious traffic)
- Capturing traffic during incident response or incident handling
- Looking for evidence of botnets, Trojans, and infected systems
- Looking for unusual traffic or traffic exceeding certain thresholds
- Testing encryption between systems or applications

From a network administration perspective, protocol analyzers can be used for activities such as these:

- Analyzing network problems
- Detecting misconfigured applications or misbehaving applications
- Gathering and reporting network usage and traffic statistics
- Debugging client/server communications

Regardless of the intended use, a protocol analyzer must be able to see network traffic in order to capture and decode it. A software-based protocol analyzer must be able to place the NIC it is going to use to monitor network traffic in *promiscuous mode* (sometimes called *promisc mode*). Promiscuous mode tells the NIC to process every network

packet it sees regardless of the intended destination. Normally, a NIC will process only *broadcast* packets (that are going to everyone on that subnet) and packets with the NIC's Media Access Control (MAC) address as the destination address inside the packet. As a sniffer, the analyzer must process every packet crossing the wire, so the ability to place a NIC into promiscuous mode is critical.

EXAM TIP A sniffer must use a NIC placed in promiscuous (promisc) mode, or it will not see all the network traffic coming into the NIC.

With older networking technologies, such as hubs, it was easier to operate a protocol analyzer, as the hub broadcast every packet across every interface regardless of the destination. With switches becoming the standard for networking equipment, placing a protocol analyzer became more difficult as switches do not broadcast every packet across every port. While this may make it harder for administrators to sniff the traffic, it also makes it harder for eavesdroppers and potential attacks.

To accommodate protocol analyzers, IDS, and IPS devices, most switch manufacturers support *port mirroring* or a *Switched Port Analyzer (SPAN)* port. Depending on the manufacturer and the hardware, a mirrored port will see all the traffic passing through the switch or through a specific VLAN(s), or all the traffic passing through other specific switch ports. The network traffic is essentially copied (or mirrored) to a specific port, which can then support a protocol analyzer.

Another option for traffic capture is to use a *network tap*, a hardware device that can be placed in-line on a network connection and that will copy traffic passing through the tap to a second set of interfaces on the tap. Network taps are often used to sniff traffic passing between devices at the network perimeter, such as the traffic passing between a router and a firewall. Many common network taps work by bridging a network connection and passing incoming traffic out one tap port (A) and outgoing traffic out another tap port (B), as shown in Figure 11-14.

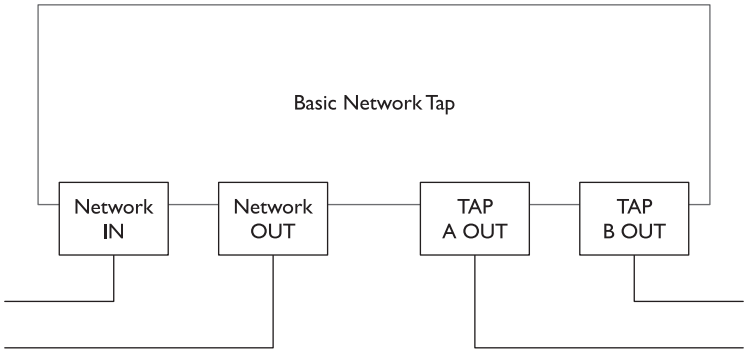


Figure 11-14 A basic network tap

A popular open-source protocol analyzer is Wireshark (www.wireshark.org/). Available for both UNIX and Windows operating systems, Wireshark is a GUI-based protocol analyzer that allows users to capture and decode network traffic on any available network interface in the system on which the software is running (including wireless interfaces). Wireshark has some interesting features, including the ability to "follow the TCP stream," which allows the user to select a single TCP packet and then see all the other packets involved in that TCP conversation.

Network Mappers

One of the biggest challenges in securing a network can be simply knowing what is connected to that network at any given point in time. For most organizations, the "network" is a constantly changing entity. While servers may remain fairly constant, user workstations, laptops, printers, and network-capable peripherals may connect to and then disconnect from the network on a daily basis, making the network at 3 AM look quite different than the network at 10 AM. To help identify devices connected to the network, many administrators use networking mapping tools.

Network mappers are tools designed to identify what devices are connected to a given network and, where possible, the operating system in use on that device. Most network mapping tools are "active" in that they generate traffic and then listen for responses to determine what devices are connected to the network. These tools typically use the ICMP or SNMP protocol for discovery and some of the more advanced tools will create a "map" of discovered devices showing their connectivity to the network in relation to other network devices. A few network mapping tools have the ability to perform device discovery passively by examining all the network traffic in an organization and noting each unique IP address and MAC address in the traffic stream.

Anti-spam

The bane of users and system administrators everywhere, *spam* is essentially unsolicited or undesired bulk electronic messages. While typically applied to e-mail, spam can be transmitted via text message to phones and mobile devices, as postings to Internet forums, and by other means. If you've ever used an e-mail account, chances are you've received spam.

From a productivity and security standpoint, spam costs businesses and users billions of dollars each year, and it is such a widespread problem that the U.S. Congress passed the CAN-SPAM Act of 2003 to empower the Federal Trade Commission to enforce the act and the Department of Justice to enforce criminal sanctions against spammers. The act establishes requirements for those who send commercial e-mail, spells out penalties for spammers and companies whose products are advertised in spam if they violate the law, and gives consumers the right to ask e-mailers to stop spamming them. Despite all our best efforts, however, spam just keeps coming; as the technologies and techniques developed to stop the spam get more advanced and complex, so do the tools and techniques used to send out the unsolicited messages.

Here are a few of the more popular methods used to fight the spam epidemic; most of these techniques are used to filter e-mail but could be applied to other mediums as well:

- **Blacklisting** Blacklisting is essentially noting which domains and source addresses have a reputation for sending spam and rejecting messages coming from those domains and source addresses. This is basically a permanent "ignore" or "call block" type capability. Several organizations and a few commercial companies provide lists of known spammers.
- **Content or keyword filtering** Similar to Internet content filtering, this method filters e-mail messages for undesirable content or indications of spam. Much like content filtering of web content, filtering e-mail based on something like keywords can cause unexpected results, as certain terms can be used in both legitimate and spam e-mail. Most content-filtering techniques use regular expression matching for keyword filtering.
- **Trusted servers** The opposite of blacklisting, a trusted server list includes SMTP servers that are being "trusted" not to forward spam.
- **Delay-based filtering** Some Simple Mail Transfer Protocol (SMTP) servers are configured to insert a deliberate pause between the opening of a connection and the sending of the SMTP server's welcome banner. Some spam-generating programs do not wait for that greeting banner, and any system that immediately starts sending data as soon as the connection is opened is treated as a spam generator and dropped by the SMTP server.
- **PTR and reverse DNS checks** Some e-mail filters check the origin domain of an e-mail sender. If the reverse checks show the mail is coming from a dial-up user, home-based broadband, a dynamically assigned address, or has a generic or missing domain, then the filter rejects it as these are common sources of spam messages.
- **Callback verification** As many spam messages use forged "from" addresses, some filters attempt to validate the "from" address of incoming e-mail. The receiving server can contact the sending server in an attempt to validate the sending address, but this is not always effective as spoofed addresses are sometimes valid e-mail addresses that can be verified.
- **Statistical content filtering** Statistical filtering is much like a document classification system. Users mark received messages as either spam or legitimate mail and the filtering system learns from the user's input. The more messages that are seen and classified as spam, the better the filtering software should get at intercepting incoming spam. Spammers counteract many filtering technologies by inserting random words and characters into the messages, making it difficult for content filters to identify patterns common to spam.
- **Rule-based filtering** Rule-based filtering is a simple technique that merely looks for matches in certain fields or keywords. For example, a rule-based filtering system may look for any message with the words "get rich" in the

subject line of the incoming message. Many popular e-mail clients have the ability to implement rule-based filtering.

- **Egress filtering** Some organizations perform spam filtering on e-mail leaving their organization as well, and this is called egress filtering. The same types of anti-spam techniques can be used to validate and filter outgoing e-mail in an effort to combat spam.
- **Hybrid filtering** Most commercial anti-spam methods use hybrid filtering, or a combination of several different techniques to fight spam. For example, a filtering solution may take each incoming message and match it against known spammers, then against a rule-based filter, then a content filter, and finally against a statistic-based filter. If the message passes all filtering stages, it will be treated as a legitimate message; otherwise, it is rejected as spam.

Much spam filtering is done at the network or SMTP server level. It's more efficient to scan all incoming and outgoing messages with a centralized solution than it is to deploy individual solutions on user desktops throughout the organization. E-mail is essentially a proxied service by default: messages generally come into and go out of an organization's mail server. (Users don't typically connect to remote SMTP servers to send and receive messages, but they can.) Anti-spam solutions are available in the form of software that is loaded on the SMTP server itself or on a secondary server that processes messages either before they reach the SMTP server or after the messages are processed by the SMTP server. Anti-spam solutions are also available in appliance form, where the software and hardware are a single integrated solution. Many centralized anti-spam methods allow individual users to customize spam filtering for their specific inbox, specifying their own filter rules and criteria for evaluating inbound e-mail.

The central issue with spam is that, despite all the effort placed into building effective spam filtering programs, spammers continue to create new methods for flooding inboxes. Spam filtering solutions are good but are far from perfect and continue to fight the constant challenge of allowing in legitimate messages while keeping the spam out. The lack of central control over Internet traffic also makes anti-spam efforts more difficult. Different countries have different laws and regulations governing e-mail, which range from draconian to nonexistent. For the foreseeable future, spam will continue to be a burden to administrators and users alike.

All-in-one Security Appliances

Many security vendors offer "all-in-one security appliances," which are devices that combine multiple functions into the same hardware appliance. Most commonly these functions are firewall, IDS/IPS, and antivirus, although all-in-one appliances can include VPN capabilities, anti-spam, malicious web traffic filtering, antispyware, content filtering, traffic shaping, and so on. All-in-one appliances are often sold as being cheaper, easier to manage, and more efficient than having separate solutions that accomplish each of the functions the all-in-one appliance is capable of performing.

Chapter Review

Intrusion detection is a mechanism for detecting unexpected or unauthorized activity on computer systems. IDSs can be host-based, examining only the activity applicable to a specific system, or network-based, examining network traffic for a large number of systems. IDSs match patterns known as *signatures* that can be content or context-based. Some IDSs are model-based and alert an administrator when activity does not match normal patterns (anomaly based) or when it matches known suspicious or malicious patterns (misuse detection). Newer versions of IDSs include prevention capabilities that will automatically block suspicious or malicious traffic before it reaches its intended destination, and many vendors call these Intrusion Prevention Systems (IPSs).

Firewalls are security devices that protect an organization's network perimeter by filtering traffic coming into the organization based on an established policy. They can be simple packet filtering devices or can have more advanced application layer filtering capabilities. Personal software firewalls are software packages that help protect individual systems by controlling network traffic coming into and out of that individual system.

Antivirus technologies scan network traffic, e-mail, files, and removable media for malicious code. Available in software and appliance form, they provide a necessary line of defense against the massive amount of malicious code roaming the Internet.

Proxies service client requests by forwarding requests from users to other servers. Proxies can be used to help filter and manage network traffic, particularly web browsing. Proxies are often combined with a content-filtering capability that administrators can use to block access to malicious or inappropriate content. Many organizations and users also employ pop-up blockers, mechanisms that prevent the annoying ads that appear in new browser windows as you visit certain web pages.

Protocol analyzers, often called sniffers, are tools that capture and decode network traffic. Analyzers must be able to see and capture network traffic to be effective, and many switch vendors support network analysis through the use of mirroring or span ports. Network traffic can also be viewed using network taps, a device for replicating network traffic passing across a physical link.

Honeypots are specialized forms of intrusion detection that involve setting up simulated hosts and services for attackers to target. Honeypots are based on the concept of luring attackers away from legitimate systems by presenting more tempting or interesting systems that, in most cases, appear to be easy targets. By monitoring activity within the honeypot, security personnel are better able to identify potential attackers along with their tools and capabilities.

Questions

1. What are the three types of event logs generated by Windows NT and 2000 systems?
   - **A.** Event, Process, and Security
   - **B.** Application, User, and Security
   - **C.** User, Event, and Security
   - **D.** Application, System, and Security

2. What are the two main types of intrusion detection systems?
   - **A.** Network-based and host-based
   - **B.** Signature-based and event-based
   - **C.** Active and reactive
   - **D.** Intelligent and passive
3. The first commercial, network-based IDS product was
   - **A.** Stalker
   - **B.** NetRanger
   - **C.** IDES
   - **D.** RealSecure
4. What are the two main types of IDS signatures?
   - **A.** Network-based and file-based
   - **B.** Context-based and content-based
   - **C.** Active and reactive
   - **D.** None of the above
5. A passive, host-based IDS
   - **A.** Runs on the local system
   - **B.** Does not interact with the traffic around it
   - **C.** Can look at system event and error logs
   - **D.** All of the above
6. Which of the following is *not* a capability of network-based IDS?
   - **A.** Can detect denial-of-service attacks
   - **B.** Can decrypt and read encrypted traffic
   - **C.** Can decode UDP and TCP packets
   - **D.** Can be tuned to a particular network environment
7. An active IDS can
   - **A.** Respond to attacks with TCP resets
   - **B.** Monitor for malicious activity
   - **C.** A and B
   - **D.** None of the above
8. Honeypots are used to
   - **A.** Attract attackers by simulating systems with open network services
   - **B.** Monitor network usage by employees
   - **C.** Process alarms from other IDSs
   - **D.** Attract customers to e-commerce sites

9. Egress filtering is used to detect SPAM that is
   - **A.** Coming into an organization
   - **B.** Sent from known spammers outside your organization
   - **C.** Leaving an organization
   - **D.** Sent to mailing lists in your organization
10. Preventative intrusion detection systems
   - **A.** Are cheaper
   - **B.** Are designed to stop malicious activity from occurring
   - **C.** Can only monitor activity
   - **D.** Were the first types of IDS
11. Which of the following is not a type of proxy?
   - **A.** Reverse
   - **B.** Web
   - **C.** Open
   - **D.** Simultaneous
12. IPS stands for
   - **A.** Intrusion processing system
   - **B.** Intrusion prevention sensor
   - **C.** Intrusion prevention system
   - **D.** Interactive protection system
13. A protocol analyzer can be used to
   - **A.** Troubleshoot network problems
   - **B.** Collect network traffic statistics
   - **C.** Monitor for suspicious traffic
   - **D.** All of the above
14. True or False: Windows Defender is available with every version of the Windows operating system.
   - **A.** True
   - **B.** False
15. Heuristic scanning looks for
   - **A.** Normal network traffic patterns
   - **B.** Viruses and spam only
   - **C.** Firewall policy violations
   - **D.** Commands or instructions that are not normally found in application programs

16. Implicit deny in a firewall rule set means:
    - **A.** All traffic is rejected
    - **B.** All incoming traffic is rejected
    - **C.** Any traffic not expressly permitted is denied
    - **D.** Any traffic not denied by a prior rule is permitted
17. An "all-in-one security appliance" typically performs which of the following functions?
    - **A.** Intrusion detection/prevention
    - **B.** Antivirus
    - **C.** Network firewall
    - **D.** All of the above
18. Which of the following security devices might have the ability to scan all outgoing and incoming web traffic to detect and block undesirable traffic such as malware, spyware, adware, malicious scripts, and file-based attacks?
    - **A.** Spam filter
    - **B.** Web security gateway
    - **C.** Honeypot
    - **D.** Packet-filtering firewall
19. A web application firewall is designed to detect and stop which of the following?
    - **A.** SQL injection attacks
    - **B.** Port scans
    - **C.** Infected e-mail traffic
    - **D.** Worms
20. What IDS model requires the system to learn what "normal" network activity looks like before it can effectively detect malicious activity?
    - **A.** Signature-based
    - **B.** Malware-based
    - **C.** Behavior-based
    - **D.** Activity-based
21. Which IDS model uses artificial intelligence and algorithms to detect malicious activity?
    - **A.** Signature-based model
    - **B.** Web-based model
    - **C.** Heuristic model
    - **D.** Denning model

22. Which of the following is a tool designed to identify what devices are connected to a given network and, where possible, the operating system in use on that device?
    - **A.** Firewall
    - **B.** Web security gateway
    - **C.** All-in-one security appliance
    - **D.** Network mapper
23. Egress filtering is:
    - **A.** Filtering e-mail traffic leaving your organization for spam
    - **B.** Filtering e-mail traffic entering your organization for spam
    - **C.** Filtering e-mail traffic from known spam senders
    - **D.** Filtering e-mail traffic between employees in your organization
24. A web security gateway performs all of the following functions except:
    - **A.** Content monitoring
    - **B.** Port mirroring
    - **C.** Data protection and compliance monitoring
    - **D.** Malware protection
25. When discussing Intrusion Prevention systems, HIPS refers to:
    - **A.** Host-based Intrusion Prevention Systems
    - **B.** Heuristic-based Intrusion Prevention Systems
    - **C.** Hardware-based Intrusion Prevention Systems
    - **D.** Holistic-based Intrusion Prevention Systems

Answers

1. **D.** The three main types of event logs generated by Windows NT and 2000 systems are Application, System, and Security.
2. **A.** The two main types of intrusion detection systems are network-based and host-based. Network-based systems monitor network connections for suspicious traffic. Host-based systems reside on an individual system and monitor that system for suspicious or malicious activity.
3. **B.** The first commercial network-based IDS product was NetRanger, released by Wheelgroup in 1995.
4. **B.** The two main types of IDS signatures are context-based and content-based. Context-based signatures examine traffic and how that traffic fits into the other traffic around it. A port scan is a good example of a context-based

signature. A content-based signature looks at what is inside the traffic, such as the contents of a specific packet.

5. **D.** A passive, host-based IDS runs on the local system, cannot interfere with traffic or activity on that system, and would have access to local system logs.
6. **B.** A network-based IDS typically cannot decrypt and read encrypted traffic. This is one of the principle weaknesses of network-based intrusion detection systems.
7. **C.** An active IDS can perform all the functions of a passive IDS (monitoring, alerting, reporting, and so on) with the added ability of responding to suspected attacks with capabilities such as sending TCP reset messages to the source and destination IP addresses.
8. **A.** Honeypots are designed to attract attackers by providing what appear to be easy, inviting targets. The honeypot collects and records the activity of attackers and their tools.
9. **C.** Egress filtering is performed to detect and stop SPAM from leaving your organization. Mail is checked as it leaves your organization.
10. **B.** Preventative intrusion detection systems are designed to "prevent" malicious actions from having any impact on the targeted system or network. For example, a host-based preventative IDS may intercept an attacker's buffer overflow attempt and prevent it from executing. By stopping the attack, the IDS prevents the attacker from affecting the system.
11. **D.** Reverse, Web, and Open are all types of proxies discussed in the chapter. Simultaneous is not a type of known proxy.
12. **C.** IPS stands for intrusion prevention system.
13. **D.** A protocol analyzer is a very flexible tool and can be used for network traffic analysis, statistics collection, and monitoring and identification of suspicious or malicious traffic.
14. **B.** False. Windows Defender is available for Windows XP, Vista, Windows Server 2003, and Windows Server 2008.
15. **D.** Heuristic scanning typically looks for commands or instructions that are not normally found in application programs.
16. **D.** Implicit deny means that any traffic not expressly permitted by a rule in the firewall's rule set or ACL is denied and rejected by the firewall.
17. **D.** All of the above. All-in-one security appliances perform multiple security roles including firewall, IDS/IPS, VPN capabilities, anti-spam, malicious web traffic filtering, antispyware, content filtering, and traffic shaping.
18. **B.** A web security gateway has the ability to scan all outgoing and incoming web traffic to detect and block undesirable traffic such as malware, spyware, adware, malicious scripts, and file-based attacks.

19. **A.** Web application firewalls are intended to address the security threats and pitfalls unique to web-based traffic such as SQL injection attacks.
20. **C.** A behavior-based IDS model relies on a collected set of "normal behavior"—what should happen on the network and is considered "normal" or "acceptable" traffic. Behavior that does not fit into the "normal" activity categories or patterns is considered suspicious or malicious.
21. **C.** The heuristic model uses artificial intelligence to detect intrusions and malicious traffic. This is typically implemented through algorithms that help an IDS decide if a traffic pattern is malicious or not.
22. **D.** A network mapper is a tool designed to identify what devices are connected to a given network and, where possible, the operating system in use on that device.
23. **A.** Egress filtering is filtering e-mail traffic leaving your organization.
24. **B.** Port mirroring is used on switches to copy packets seen on one or more ports to a different port, typically for monitoring purposes.
25. **A.** HIPS refers to Host-based Intrusion Prevention Systems.