UNIT-5

5.1 INTRUDERS

One of the most publicized attacks to security is the intruder, generally referred to as hacker or cracker. Three classes of intruders are as follows:

· **Masquerader** – an individual who is not authorized to use the computer and who penetrates a system's access controls to exploit a legitimate user's account.

Misfeasor – a legitimate user who accesses data, programs, or resources for which such access is not authorized, or who is authorized for such access but misuse his or her privileges.

Clandestine user – an individual who seizes supervisory control of the system and uses this control to evade auditing and access controls or to suppress audit collection.

The masquerader is likely to be an outsider; the misfeasor generally is an insider; and the clandestine user can be either an outsider or an insider.
Intruder attacks range from the benign to the serious. At the benign end of the scale, there are many people who simply wish to explore internets and see what is out there. At the serious end are individuals who are attempting to read privileged data, perform unauthorized modifications to data, or disrupt the system. Benign intruders might be tolerable, although they do consume resources and may slow performance for legitimate users. However there is no way in advance to know whether an intruder will be benign or malign.

An analysis of previous attack revealed that there were two levels of hackers:

The high levels were sophisticated users with a thorough knowledge of the technology. The low levels were the 'foot soldiers' who merely use the supplied cracking programs with little understanding of how they work. One of the results of the growing awareness of the intruder problem has been the establishment of a number of Computer Emergency Response Teams (CERT). these co- operative ventures collect information about system vulnerabilities and disseminate it to systems managers. Unfortunately, hackers can also gain access to CERT reports.

In addition to running password cracking programs, the intruders attempted to modify login software to enable them to capture passwords of users logging onto the systems.

Intrusion techniques

The objective of the intruders is to gain access to a system or to increase the range of privileges

accessible on a system. Generally, this requires the intruders to acquire information that should be protected. In most cases, the information is in the form of a user password.
Typically, a system must maintain a file that associates a password with each authorized user. If such a file is stored with no protection, then it is an easy matter to gain access to it. The password files can be protected in one of the two ways:

One way encryption – the system stores only an encrypted form of user's password. In practice, the system usually performs a one way transformation (not reversible) in which the password is used to generate a key for the encryption function and in which a fixed length output is produced.

Access control – access to the password file is limited to one or a very few accounts.

The following techniques are used for learning passwords.

Try default passwords used with standard accounts that are shipped with the system. Many administrators do not bother to change these defaults.

Exhaustively try all short passwords.

Try words in the system's online dictionary or a list of likely passwords.

Collect information about users such as their full names, the name of their spouse and children, pictures in their office and books in their office that are related to hobbies.

Try user's phone number, social security numbers and room numbers. Try all legitimate license plate numbers.

Use a torjan horse to bypass restriction on access.

Tap the line between a remote user and the host system. Two principle countermeasures: Detection – concerned with learning of an attack, either before or after its success. Prevention – challenging security goal and an uphill bottle at all times.

5.2 INTRUSION DETECTION:

Inevitably, the best intrusion prevention system will fail. A system's second line of defense is intrusion detection, and this has been the focus of much research in recent years. This interest is motivated by a number of considerations, including the following:

If an intrusion is detected quickly enough, the intruder can be identified and ejected from the system before any damage is done or any data are compromised.

An effective intrusion detection system can serve as a deterrent, so acting to prevent intrusions.

3. Intrusion detection enables the collection of information about intrusion techniques that can be used to strengthen the intrusion prevention facility.

Intrusion detection is based on the assumption that the behavior of the intruder differs from that of a legitimate user in ways that can be quantified.

Figure 18.1 suggests, in very abstract terms, the nature of the task confronting the designer of an intrusion detection system. Although the typical behavior of an intruder differs from the typical behavior of an authorized user, there is an overlap in these behaviors. Thus, a loose interpretation of intruder behavior, which will catch more intruders, will also lead to a number of "false positives," or authorized users identified as intruders. On the other hand, an attempt to limit false positives by a tight interpretation of intruder behavior will lead to an increase in false negatives, or intruders not identified as intruders. Thus, there is an element of compromise and art in the practice of intrusion detection.

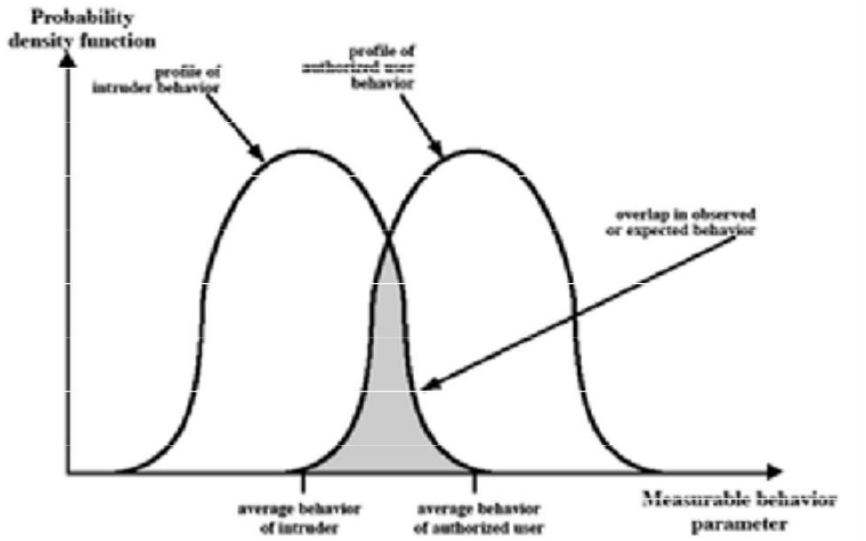


Figure 18.1 Profiles of Behavior of Intruders and Authorized Users

[PORR92] identifies the following approaches to intrusion detection:

1. **Statistical anomaly detection**: Involves the collection of data relating to the behavior of legitimate users over a period of time. Then statistical tests are applied to observed behavior to

determine with a high level of confidence whether that behavior is not legitimate user behavior.

a. **Threshold detection**: This approach involves defining thresholds, independent of user, for the frequency of occurrence of various events.
b. **Profile based:** A profile of the activity of each user is developed and used to detect changes in the behavior of individual accounts.
2. **Rule-based detection**: Involves an attempt to define a set of rules that can be used to decide that a given behavior is that of an intruder.
a. **Anomaly detection**: Rules are developed to detect deviation from previous usage patterns.

b. **Penetration identification**: An expert system approach that searches for suspicious behavior. In terms of the types of attackers listed earlier, statistical anomaly detection is effective against masqueraders. On the other hand, such techniques may be unable to deal with misfeasors. For such attacks, rule-based approaches may be able to recognize events and sequences that, in context, reveal penetration. In practice, a system may exhibit a combination of both approaches to be effective against a broad range of attacks.

Audit Records

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

Native audit records: Virtually all multiuser operating systems include accounting software that collects information on user activity. The advantage of using this information is that no additional collection software is needed. The disadvantage is that the native audit records may not contain the needed information or may not contain it in a convenient form.

Detection-specific audit records: A collection facility can be implemented that generates audit records containing only that information required by the intrusion detection system. One advantage of such an approach is that it could be made vendor independent and ported to a variety of systems. The disadvantage is the extra overhead involved in having, in effect, two accounting packages running on a machine.

Each audit record contains the following fields:

Subject: Initiators of actions. A subject is typically a terminal user but might also be a process acting on behalf of users or groups of users.

Object: Receptors of actions. Examples include files, programs, messages, records, terminals, printers, and user- or program-created structures

Resource-Usage: A list of quantitative elements in which each element gives the amount used of some resource (e.g., number of lines printed or displayed, number of records read or written, processor time, I/O units used, session elapsed time).

Time-Stamp: Unique time-and-date stamp identifying when the action took place.

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

COPY GAME.EXE TO <Library>GAME.EXE

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

Smith	execute	<Library>COPY.EXE	0	CPU = 00002	11058721678

Smith	read	<Smith>GAME.EXE	0	RECORDS = 0	11058721679

Smith	execute	<Library>COPY.EXE	write-viol	RECORDS = 0	11058721680

In this case, the copy is aborted because Smith does not have write permission to <Library>. The decomposition of a user operation into elementary actions has three advantages:

Because objects are the protectable entities in a system, the use of elementary actions enables an audit of all behavior affecting an object. Thus, the system can detect attempted subversions of access

Single-object, single-action audit records simplify the model and the implementation.

Because of the simple, uniform structure of the detection-specific audit records, it may be relatively easy to obtain this information or at least part of it by a straightforward mapping from existing native audit records to the detection-specific audit records.

Statistical Anomaly Detection:

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

Threshold analysis, by itself, is a crude and ineffective detector of even moderately sophisticated attacks. Both the threshold and the time interval must be determined.

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

The foundation of this approach is an analysis of audit records. The audit records provide input to the intrusion detection function in two ways. First, the designer must decide on a number of quantitative metrics that can be used to measure user behavior. Examples of metrics that are useful for profile-based intrusion detection are the following:

Counter: A nonnegative integer that may be incremented but not decremented until it is reset by management action. Typically, a count of certain event types is kept over a particular period of time. Examples include the number of logins by a single user during an hour, the number of times a given command is executed during a single user session, and the number of password failures during a minute.

Gauge: A nonnegative integer that may be incremented or decremented. Typically, a gauge is used to measure the current value of some entity. Examples include the number of logical connections assigned to a user application and the number of outgoing messages queued for a user process.

Interval timer: The length of time between two related events. An example is the length of time between successive logins to an account.

Resource utilization: Quantity of resources consumed during a specified period.

Examples include the number of pages printed during a user session and total time consumed by a program execution.Given these general metrics, various tests can be performed to determine whether current activity fits within acceptable limits. [DENN87] lists the following approaches that may be taken:

Mean and standard
deviation Multivariate
Markov
process
Time
series
Operational

The simplest statistical test is to measure the mean and standard deviation of a parameter over some historical period. This gives a reflection of the average behavior and its variability.
A multivariate model is based on correlations between two or more variables. Intruder behavior may be characterized with greater confidence by considering such correlations (for example, processor time and resource usage, or login frequency and session elapsed time).

A Markov process model is used to establish transition probabilities among various states. As an example, this model might be used to look at transitions between certain commands. A time series model focuses on time intervals, looking for sequences of events that happen too rapidly or too slowly. A variety of statistical tests can be applied to characterize abnormal timing. Finally, an operational model is based on a judgment of what is considered abnormal, rather than an automated analysis of past audit records. Typically, fixed limits are defined and intrusion is suspected for an observation that is outside the limits.

Rule-Based Intrusion Detection

Rule-based techniques detect intrusion by observing events in the system and applying a set of rules that lead to a decision regarding whether a given pattern of activity is or is not suspicious.

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

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

Rule-based penetration identification takes a very different approach to intrusion detection, one based on expert system technology. The key feature of such systems is the use of rules for identifying known penetrations or penetrations that would exploit known weaknesses.

Example heuristics are the following:

Users should not read files in other users' personal directories.

Users must not write other users' files.

Users who log in after hours often access the same files they used earlier.

Users do not generally open disk devices directly but rely on higher-level operating system utilities.

Users should not be logged in more than once to the same system.

Users do not make copies of system programs.

5.2.3 The Base-Rate Fallacy

To be of practical use, an intrusion detection system should detect a substantial percentage of intrusions while keeping the false alarm rate at an acceptable level. If only a modest percentage of actual intrusions are detected, the system provides a false sense of security. On the other hand, if

the system frequently triggers an alert when there is no intrusion (a false alarm), then either system managers will begin to ignore the alarms, or much time will be wasted analyzing the false alarms.

Unfortunately, because of the nature of the probabilities involved, it is very difficult to meet the standard of high rate of detections with a low rate of false alarms. In general, if the actual numbers of intrusions is low compared to the number of legitimate uses of a system, then the false alarm rate will be high unless the test is extremely discriminating.

5.2.4 Distributed Intrusion Detection

Until recently, work on intrusion detection systems focused on single-system stand-alone facilities. The typical organization, however, needs to defend a distributed collection of hosts supported by a LAN Porras points out the following major issues in the design of a distributed intrusion detection system

A distributed intrusion detection system may need to deal with different audit record formats. In a heterogeneous environment, different systems will employ different native audit collection systems and, if using intrusion detection, may employ different formats for security-related audit records.

One or more nodes in the network will serve as collection and analysis points for the data from the systems on the network. Thus, either raw audit data or summary data must be transmitted across the network. Therefore, there is a requirement to assure the integrity and confidentiality of these data.

Either a centralized or decentralized architecture can be used.

Host agent module: An audit collection module operating as a background process on a monitored system. Its purpose is to collect data on security-related events on the host and transmit these to the central manager.

LAN monitor agent module: Operates in the same fashion as a host agent module except that it analyzes LAN traffic and reports the results to the central manager.

Central manager module: Receives reports from LAN monitor and host agents and processes and correlates these reports to detect intrusion.

A filter is applied that retains only those records that are of security interest.

These records are then reformatted into a standardized format referred to as the host audit record (HAR).

Next, a template-driven logic module analyzes the records for suspicious activity.

At the lowest level, the agent scans for notable events that are of interest independent of any past events.

Examples include failed file accesses, accessing system files, and changing a file's access control. At the next higher level, the agent looks for sequences of events, such as known attack
patterns (signatures).

Finally, the agent looks for anomalous behavior of an individual user based on a historical profile of that user, such as number of programs executed, number of files accessed, and the like.

When suspicious activity is detected, an alert is sent to the central manager.

The central manager includes an expert system that can draw inferences from received data. The manager may also query individual systems for copies of HARs to correlate with those from other agents.

The LAN monitor agent also supplies information to the central manager.

The LAN monitor agent audits host-host connections, services used, and volume of traffic.

I t searches for significant events, such as sudden changes in network load, the use of security-related services, and network activities such as rlogin.

The architecture depicted in Figures 18.2 and 18.3 is quite general and flexible. It offers a foundation for a machine-independent approach that can expand from stand-alone intrusion detection to a system that is able to correlate activity from a number of sites and networks to detect suspicious activity that would otherwise remain undetected.

5.2.5 Honeypots

A relatively recent innovation in intrusion detection technology is the honeypot. Honeypots are decoy systems that are designed to lure a potential attacker away from critical systems. Honeypots are designed to divert an attacker from accessing critical systems collect information about the attacker's activity encourage the attacker to stay on the system long enough for administrators to respond These systems are filled with fabricated information designed to appear valuable but that a legitimate user of the system wouldn't access. Thus, any access to the honeypot is suspect.

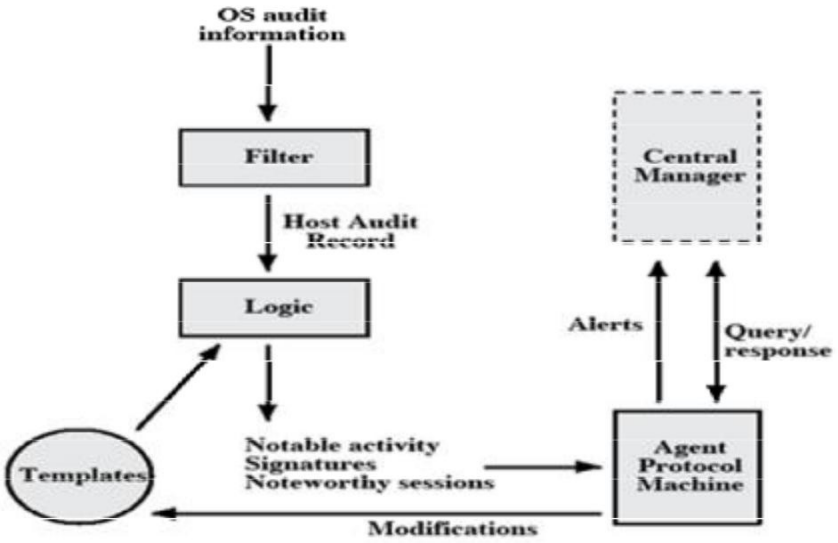


Figure 18.3 Agent Architecture

5.2.6 Intrusion Detection Exchange Format

To facilitate the development of distributed intrusion detection systems that can function across a wide range of platforms and environments, standards are needed to support interoperability. Such standards are the focus of the IETF Intrusion Detection Working Group. The outputs of this working group include the following:

A requirements document, which describes the high-level functional requirements for communication between intrusion detection systems and with management systems, including the rationale for those requirements.

A common intrusion language specification, which describes data formats that satisfy the requirements.
A framework document, which identifies existing protocols best used for communication between intrusion detection systems, and describes how the devised data formats relate to them.

5.3 FIREWALLS

Firewall design principles

Internet connectivity is no longer an option for most organizations. However, while internet access provides benefits to the organization, it enables the outside world to reach and interact with local network assets. This creates the threat to the organization. While it is possible to equip each workstation and server on the premises network with strong security features, such as intrusion protection, this is not a practical approach. The alternative, increasingly accepted, is the firewall.
The firewall is inserted between the premise network and internet to establish a controlled link and to erect an outer security wall or perimeter. The aim of this perimeter is to protect the premises network from internet based attacks and to provide a single choke point where security and audit can be imposed. The firewall can be a single computer system or a set of two or more systems that cooperate to perform the firewall function.

Firewall characteristics:
All traffic from inside to outside, and vice versa, must pass through the firewall.
This is achieved by physically blocking all access to the local network except via the firewall. Various configurations are possible. Only authorized traffic, as defined by the local security policy, will be allowed to pass. Various types of firewalls are used, which implement various types of security policies.

the firewall itself is immune to penetration. This implies that use of a trusted system with a secure operating system. This implies that use of a trusted system with a secure operating system.

Four techniques that firewall use to control access and enforce the site's security policy is as follows:

Service control – determines the type of internet services that can be accessed, inbound or outbound. The firewall may filter traffic on this basis of IP address and TCP port number; may provide proxy software that receives and interprets each service request before passing it on; or may host the server software itself, such as web or mail service.

Direction control – determines the direction in which particular service request may be initiated and allowed to flow through the firewall.

User control – controls access to a service according to which user is attempting to access it.

Behavior control – controls how particular services are used.

Capabilities of firewall

A firewall defines a single choke point that keeps unauthorized users out of the protected network, prohibits potentially vulnerable services from entering or leaving the network, and provides protection from various kinds of IP spoofing and routing attacks.

A firewall provides a location for monitoring security related events. Audits and alarms can be implemented on the firewall system.

A firewall is a convenient platform for several internet functions that are not security related.

A firewall can serve as the platform for IPsec.

Limitations of firewall

The firewall cannot protect against attacks that bypass the firewall. Internal systems may have dial-out capability to connect to an ISP. An internal LAN may support a modem pool that provides dial-in capability for traveling employees and telecommuters.

the firewall does not protect against internal threats. The firewall does not protect against internal threats, such as a disgruntled employee or an employee who unwittingly cooperates with an external attacker.

The firewall cannot protect against the transfer of virus-infected programs or files.

Because of the variety of operating systems and applications supported inside the perimeter, it would be impractical and perhaps impossible for the firewall to scan all incoming files, e-mail, and messages for viruses.

Types of firewalls There are 3 common types of firewalls. Packet filters

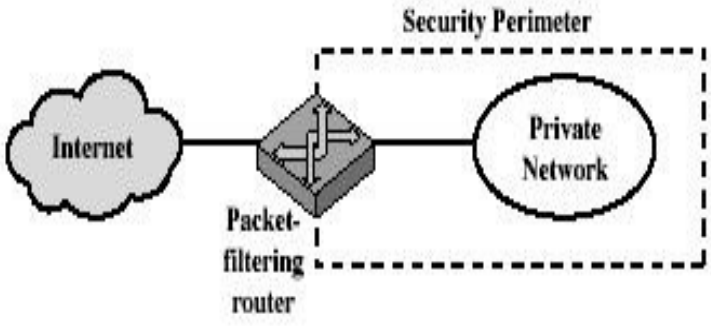
1. Application-level
2. gateways Circuit-
   level gateways
3. Packet filtering router

A packet filtering router applies a set of rules to each incoming IP packet and then forwards or discards the packet. The router is typically configured to filter packets going in both directions. Filtering rules are based on the information contained in a network packet:

Source IP address – IP address of the system that originated the IP packet.

Destination IP address – IP address of the system, the IP is trying to reach.
Source and destination transport level address – transport level port number.
IP protocol field – defines the transport protocol.

Interface – for a router with three or more ports, which interface of the router the packet come from or which interface of the router the packet is destined for.



(a) Packet-filtering router

The packet filter is typically set up as a list of rules based on matches to fields in the IP or TCP header. If there is a match to one of the rules, that rule is invoked to determine whether to forward or discard the packet. If there is no match to any rule, then a default action is taken.
Two default policies are possible:

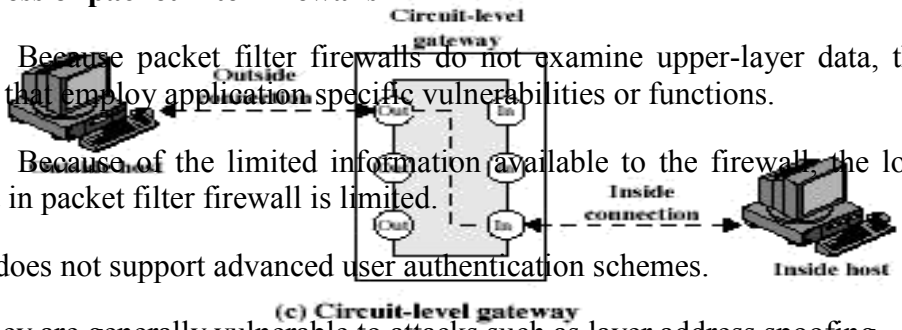
Default = discard: That which is not expressly permitted is prohibited.
Default = forward: That which is not expressly prohibited is permitted.

The default discard policy is the more conservative. Initially everything is blocked, and services must be added on a case-by-case basis. This policy is more visible to users, who are most likely to see the firewall as a hindrance. The default forward policy increases ease of use for end users but provides reduced security.

Advantages of packet filter router

Simple
Transparent to
users Very fast

Weakness of packet filter firewalls



(c) Circuit-level gateway

Because packet filter firewalls do not examine upper-layer data, they cannot prevent attacks that employ application specific vulnerabilities or functions.

Because of the limited information available to the firewall, the logging functionality present in packet filter firewall is limited.

It does not support advanced user authentication schemes.

They are generally vulnerable to attacks such as layer address spoofing.

Some of the attacks that can be made on packet filtering routers and the appropriate counter measures are the following:

5 . 3 . 1 IP address spoofing –

the intruders transmit packets from the outside with a source IP address field containing an address of an internal host.

Countermeasure: to discard packet with an inside source address if the packet arrives on an external interface.

Source routing attacks – the source station specifies the route that a packet should take as it crosses the internet; i.e., it will bypass the firewall.

Countermeasure: to discard all packets that uses this option.

Tiny fragment attacks – the intruder create extremely small fragments and force the TCP header information into a separate packet fragment. The attacker hopes that only the first fragment is examined and the remaining fragments are passed through.

Countermeasure: to discard all packets where the protocol type is TCP and the IP Fragment offset is equal to 1.

Application level gateway

An Application level gateway, also called a proxy server, acts as a relay of application level traffic. The user contacts the gateway using a TCP/IP application, such as Telnet or FTP, and the gateway asks the user for the name of the remote host to be accessed. When the user responds and provides a valid user ID and authentication information, the gateway contacts the application on the remote host and relays TCP segments containing the application data between the two endpoints.

Application level gateways tend to be more secure than packet filters. It is easy to log and audit all incoming traffic at the application level. A prime disadvantage is the additional processing overhead on each connection.

(b) Application-level gateway

Circuit level gateway

Circuit level gateway can be a stand-alone system or it can be a specified function performed by an application level gateway for certain applications. A Circuit level gateway does not permit an end-to-end TCP connection; rather, the gateway sets up two TCP connections, one between itself and a TCP user on an inner host and one between itself and a TCP user on an outer host. Once the two connections are established, the gateway typically relays TCP segments from one connection to the other without examining the contents. The security function consists of determining which connections will be allowed.

A typical use of Circuit level gateways is a situation in which the system administrator trusts the internal users. The gateway can be configured to support application level or proxy service on inbound connections and circuit level functions for outbound connections.

Basiton host

It is a system identified by the firewall administrator as a critical strong point in the network's security. The Bastion host serves as a platform for an application level and circuit level gateway. Common characteristics of a Basiton host are as follows:

The Bastion host hardware platform executes a secure version of its operating system, making it a trusted system.

Only the services that the network administrator considers essential are installed on the Bastion host.
It may require additional authentication before a user is allowed access to the proxy services.

Each proxy is configured to support only a subset of standard application's command set.

Each proxy is configured to allow access only to specific host systems.

Each proxy maintains detailed audit information by logging all traffic, each connection and the duration of each connection.

Each proxy is independent of other proxies on the Bastion host.

A proxy generally performs no disk access other than to read its initial configuration file.

Each proxy runs on a non-privileged user in a private and secured directory on the Bastion host.

5.3.2 Firewall configurations

There are 3 common firewall configurations.

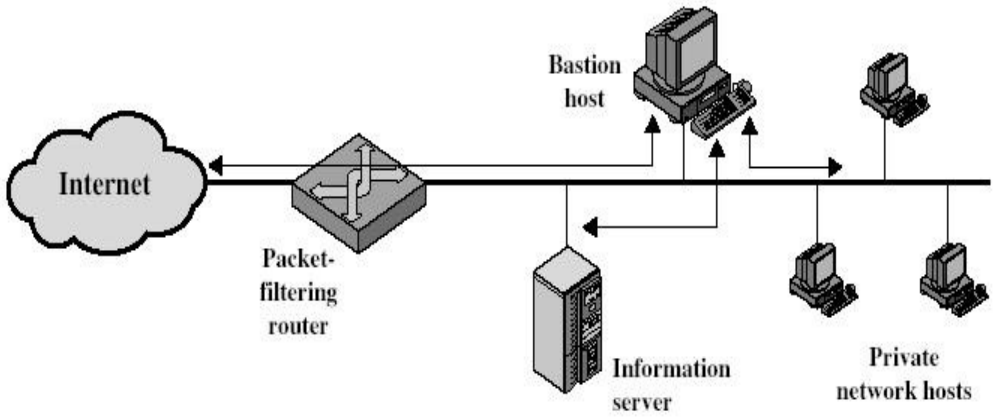
1. Screened host firewall, single-homed basiton configuration

In this configuration, the firewall consists of two systems: a packet filtering router and a bastion host. Typically, the router is configured so that For traffic from the internet, only IP packets destined for the bastion host are allowed in. For traffic from the internal network, only IP packets from the basiton host are allowed out.

The basiton host performs authentication and proxy functions. This configuration has greater security than simply a packet filtering router or an application level gateway alone, for two reasons:

This configuration implements both packet level and application level filtering, allowing for considerable flexibility in defining security policy.

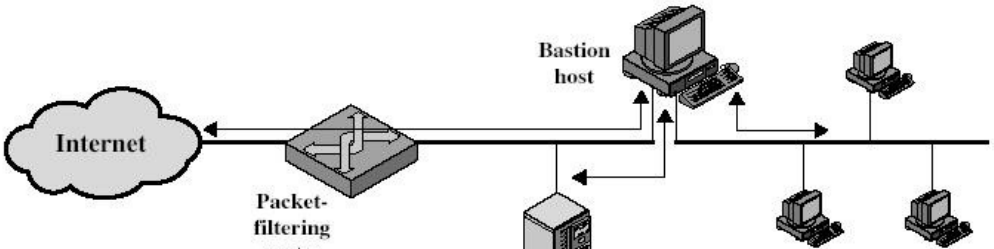
An intruder must generally penetrate two separate systems before the security of the internal network is compromised.

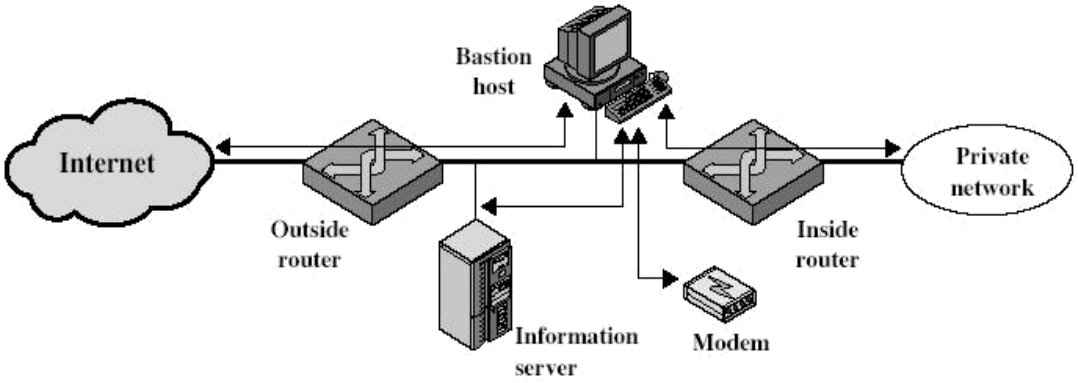


(a) Screened host firewall system (single-homed bastion host)

2. Screened host firewall, dual homed basiton configuration

In the previous configuration, if the packet filtering router is compromised, traffic could flow directly through the router between the internet and the other hosts on the private network. This configuration physically prevents such a security break.





(c) Screened-subnet firewall system

3. Screened subnet firewall configuration

In this configuration, two packet filtering routers are used, one between the basiton host and internet and one between the basiton host and the internal network. This configuration creates an isolated subnetwork, which may consist of simply the basiton host but may also include one or more information servers and modems for dial-in capability. Typically both the internet and the internal network have access to hosts on the screened subnet, but traffic across the screened subnet is blocked. This configuration offers several advantages:

There are now three levels of defense to thwart intruders.

The outside router advertises only the existence of the screened subnet to the internet; therefore the internal network is invisible to the internet.

Similarly, the inside router advertises only the existence of the screened subnet to the internal network; therefore the systems on the internal network cannot construct direct routes to the internet.

Trusted systems

One way to enhance the ability of a system to defend against intruders and malicious programs is to implement trusted system technology.

Data access control

Following successful logon, the user has been granted access to one or set of hosts and applications. This is generally not sufficient for a system that includes sensitive data in its database. Through the user access control procedure, a user can be identified to the system. Associated with each user, there can be a profile that specifies permissible operations and file accesses. The operating system can then enforce rules based on the user profile. The database

management system, however, must control access to specific records or even portions of records. The operating system may grant a user permission to access a file or use an application, following which there are no further security checks, the database management system must make a decision on each individual access attempt. That decision will depend not only on the user's identity but also on the specific parts of the data being accessed and even on the information already divulged to the user.

A general model of access control as exercised by an file or database management system is that of an access matrix. The basic elements of the model are as follows:

Subject: An entity capable of accessing objects. Generally, the concept of subject equates with that of process.

Object: Anything to which access is controlled. Examples include files, portion of files, programs, and segments of memory.

Access right: The way in which the object is accessed by a subject. Examples are read, write and execute.
One axis of the matrix consists of identified subjects that may attempt data access. Typically, this list will consist of individual users or user groups. The other axis lists the objects that may be accessed. Objects may be individual data fields. Each entry in the matrix indicates the access rights of that subject for that object. The matrix may be decomposed by columns, yielding **access control lists.** Thus, for each object, an access control list lists users and their permitted access rights. The access control list may contain a default, or public, entry.

a. Access matrix

Access control list for Program1:

Process1 (Read, Execute)

Access control list for Segment A:

Process1 (Read, Write)

Access control list for Segment B:

Process2 (Read)

b. Access control list

Capability list for Process1: Program1 (Read, Execute) Segment A (Read)

Capability list for Process2:

Segment B
(Read) **c.**
Capability list

Decomposition by rows yields **capability tickets**. A capability ticket specifies authorized objects and operations for a user. Each user has a number of tickets and may be authorized to loan or give them to others. Because tickets may be dispersed around the system, they present a greater security problem than access control lists. In particular, the ticket must be unforgeable. One way to accomplish this is to have the operating system hold all tickets on behalf of users. These tickets would have to be held in a region of memory inaccessible to users.

The concept of Trusted Systems

When multiple categories or levels of data are defined, the requirement is referred to as multilevel security. The general statement of the requirement for multilevel security is that a subject at a high level may not convey information to a subject at a lower or noncomparable level unless that flow accurately reflects the will of an authorized user. For implementation purposes, this requirement is in two parts and is simply stated. A multilevel secure system must enforce:

No read up: A subject can only read an object of less or equal security level. This is referred to as **simple security property.**

No write down: A subject can only write into an object of greater or equal security level. This is referred to as ***-property (star property).**

These two rules, if properly enforced, provide multilevel security.

Reference Monitor concept

The reference monitor is a controlling element in the hardware and operating system of a computer that regulates the access of subjects to objects on the basis of

security parameters of the subject and object. The reference monitor has access to a file, known as the security kernel database that lists the access privileges (security clearance) of each subject and the protection attributes (classification level) of each object. The reference monitor enforces the security rules and has the following properties:

Complete mediation: The security rules are enforced on every access, not just, fr example, when a file is opened.

Isolation: The reference monitor and database are protected from unauthorised modification.

Verifiability: The reference monitor's correctness must be provable. That is, it must be possible to demonstrate mathematically that the reference monitor enforces the security rules and provides complete mediation and isolation. Important security events, such as detected security violations and

authorized changes to the security kernel database, are stored in the audit file.

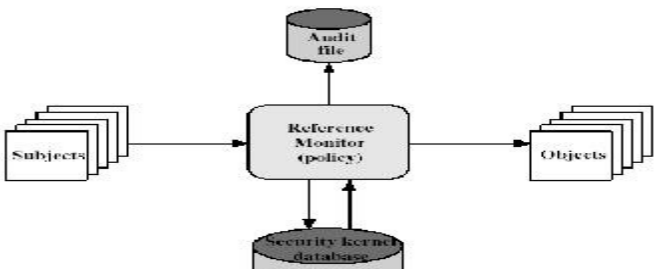


Fig: Reference Monitor Concept

5.4 VIRUSES AND RELATED THREATS

Perhaps the most sophisticated types of threats to computer systems are presented by programs that exploit vulnerabilities in computing systems.

Malicious Programs

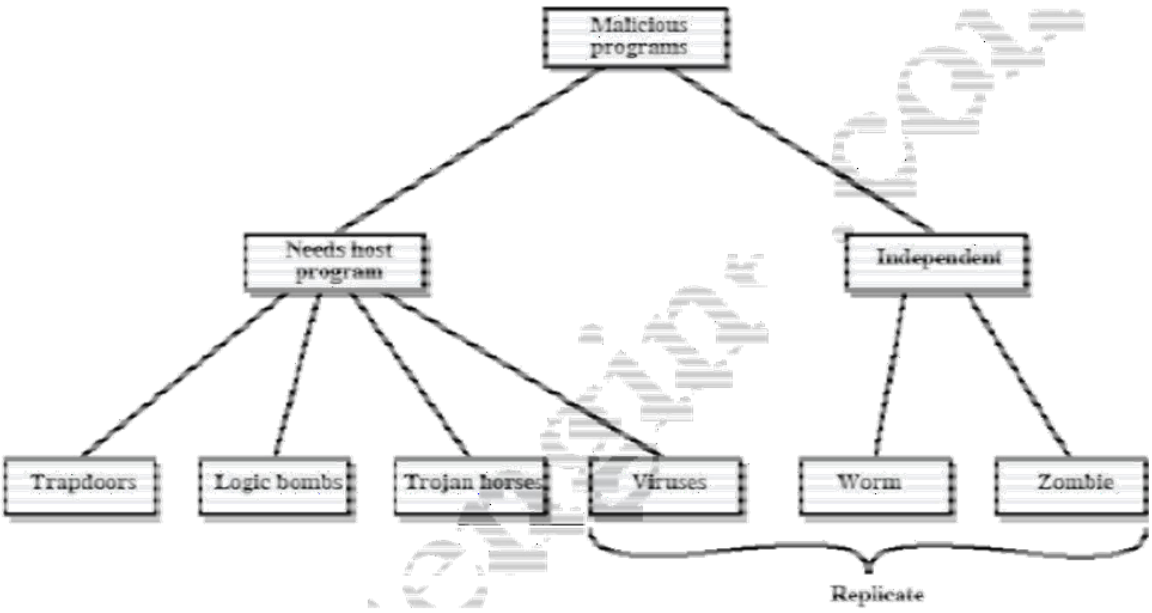


Figure 19.1 Taxonomy of Malicious Programs

Name	Description

Virus	Attaches itself to a program and propagates copies of itself to othe programs
Worm	Program that propagates copies of itself to other computers
Logic bomb	Triggers action when condition occurs
Trojan horse	Program that contains unexpected additional functionality
Backdoor	Programmodificationthatallowsunauthorizedaccesst functionality
Exploits	Code specific to a single vulnerability or set of vulnerabilities
Downloaders	Program that installs other items on a machine that is under attack Usually, a downloader is sent in an e-mail.
Auto-rooter	Malicious hacker tools used to break into new machines remotely
Kit (viru generator)	Set of tools for generating new viruses automatically
Spammer programs	Used to send large volumes of unwanted e-mail
Flooders	Used to attack networked computer systems with a large volume o traffic to carry out a denial of service (DoS) attack
Keyloggers	Captures keystrokes on a compromised system
Rootkit	Setofhackertoolsusedafterattackerhasbrokenintoacompute system and gained root-level access
Zombie	Program activated on an infected machine that is activated to launc attacks on other machines

Malicious software can be divided into two categories: those that need a host program, and those that are independent.

The former are essentially fragments of programs that cannot exist independently of some actual application program, utility, or system program. Viruses, logic bombs, and backdoors are examples. The latter are self-contained programs that can be scheduled and run by the operating system. Worms and zombie programs are examples.

The Nature of Viruses

A virus is a piece of software that can "infect" other programs by modifying them; the modification includes a copy of the virus program, which can then go on to infect other programs. A virus can do anything that other programs do. The only difference is that it attaches itself to another program and executes secretly when the host program is run. Once a virus is executing, it can perform any function, such as erasing files and programs.

During its lifetime, a typical virus goes through the following four phases:

Dormant phase: The virus is idle. The virus will eventually be activated by some event, such as a date, the presence of another program or file, or the capacity of the disk exceeding some limit. Not all viruses have this stage.

Propagation phase: The virus places an identical copy of itself into other programs or into certain system areas on the disk. Each infected program will now contain a clone of the virus, which will itself enter a propagation phase.

Triggering phase: The virus is activated to perform the function for which it was intended. As with the dormant phase, the triggering phase can be caused by a variety of system events, including a count of the number of times that this copy of the virus has made copies of itself.

Execution phase: The function is performed. The function may be harmless, such as a message on the screen, or damaging, such as the destruction of programs and data files.

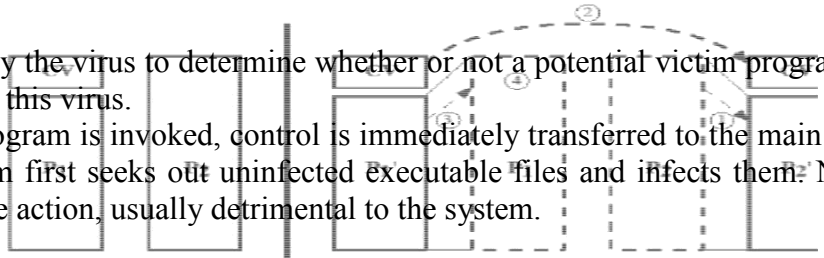
Virus Structure

A virus can be prepended or postpended to an executable program, or it can be embedded in some other fashion. The key to its operation is that the infected program, when invoked, will first execute the virus code and then execute the original code of the program.

An infected program begins with the virus code and works as follows.

The first line of code is a jump to the main virus program. The second line is a special marker

that is used by the virus to determine whether or not a potential victim program has already been infected with this virus.
When the program is invoked, control is immediately transferred to the main virus program. The virus program first seeks out uninfected executable files and infects them. Next, the virus may perform some action, usually detrimental to the system.



This action could be performed every time the program is invoked, or it could be a logic bomb that triggers only under certain conditions.
Finally, the virus transfers control to the original program. If the infection phase of the program is reasonably rapid, a user is unlikely to notice any difference between the execution of an infected and uninfected program.

Figure 19.4 A Compression Virus

A virus such as the one just described is easily detected because an infected version of a program is longer than the corresponding uninfected one. A way to thwart such a simple means of detecting a virus is to compress the executable file so that both the infected and

uninfected versions are of identical length.. The key lines in this virus are numbered, and Figure 19.3 [COHE94] illustrates the operation. We assume that program P1 is infected with the virus CV. When this program is invoked, control passes to its virus, which performs the following steps:

For each uninfected file P2 that is found, the virus first compresses that file to produce P'2,

which is shorter than the original program by the size of the virus.

A copy of the virus is prepended to the compressed program.

The compressed version of the original infected program, P'1, is uncompressed.

The uncompressed original program is executed.

In this example, the virus does nothing other than propagate. As in the previous example, the virus may include a logic bomb.

Initial Infection

Once a virus has gained entry to a system by infecting a single program, it is in a position to infect some or all other executable files on that system when the infected program executes. Thus, viral infection can be completely prevented by preventing the virus from gaining entry in the first place. Unfortunately, prevention is extraordinarily difficult because a virus can be part of any program outside a system. Thus, unless one is content to take an absolutely bare piece of

Iron and write all one's own system and application programs, one is vulnerable.

Types of Viruses

Following categories as being among the most significant types of viruses:

Parasitic virus: The traditional and still most common form of virus. A parasitic virus attaches itself to executable files and replicates, when the infected program is executed, by finding other executable files to infect.

Memory-resident virus: Lodges in main memory as part of a resident system program.

From that point on, the virus infects every program that executes.

Boot sector virus: Infects a master boot record or boot record and spreads when a system is booted from the disk containing the virus.

Stealth virus: A form of virus explicitly designed to hide itself from detection by antivirus software.

Polymorphic virus: A virus that mutates with every infection, making detection by the

"signature" of the virus impossible.

Metamorphic virus: As with a polymorphic virus, a metamorphic virus mutates with every infection. The difference is that a metamorphic virus rewrites itself completely at each iteration, increasing the difficulty of detection. Metamorphic viruses my change their behavior as well as their appearance.

One example of a **stealth virus** was discussed earlier: a virus that uses compression so that the infected program is exactly the same length as an uninfected version. Far more sophisticated techniques are possible. For example, a virus can place intercept logic in disk I/O routines, so that when there is an attempt to read suspected portions of the disk using these routines, the virus will present back the original, uninfected program.

A **polymorphic virus** creates copies during replication that are functionally equivalent but have distinctly different bit patterns.

Macro Viruses

In the mid-1990s, macro viruses became by far the most prevalent type of virus. Macro viruses are particularly threatening for a number of reasons:

1. A macro virus is platform independent. Virtually all of the macro viruses infect Microsoft Word documents. Any hardware platform and operating system that supports Word can be infected.
2. Macro v i r u s e si nf e c td o c u m e n t s , n o te x e c u t a b l ep ort i o nso fc o d e .Most o f t h e information introduced onto a computer system is in the form of a document rather than a program.
3. Macro viruses are easily spread. A very common method is by electronic mail.

Macro viruses take advantage of a feature found in Word and other office applications such as Microsoft Excel, namely the macro. In essence, a macro is an executable program embedded in a word processing document or other type of file. Typically, users employ macros to automate repetitive tasks and thereby save keystrokes. The macro language is usually some form of the Basic programming language. A user might define a sequence of keystrokes in a macro and set it up so that the macro is invoked when a function key or special short combination of keys is input.
Successive releases of Word provide increased protection against macro viruses. For example, Microsoft offers an optional Macro Virus Protection tool that detects suspicious Word files and alerts the customer to the potential risk of opening a file with macros. Various antivirus product vendors have also developed tools to detect and correct macro viruses.

E-mail Viruses
A more recent development in malicious software is the e-mail virus. The first rapidly spreading e-mail viruses, such as Melissa, made use of a Microsoft Word macro embedded in an attachment. If the recipient opens the e-mail attachment, the Word macro is activated. Then

1. The e-mail virus sends itself to everyone on the mailing list in the user's e-mail package.
2. The virus does local damage.

Worms

A worm is a program that can replicate itself and send copies from computer to computer across network connections. Upon arrival, the worm may be activated to replicate and propagate again.

Network worm programs use network connections to spread from system to system. Once active within a system, a network worm can behave as a computer virus or bacteria, or it could implant Trojan horse programs or perform any number of disruptive or destructive actions.

To replicate itself, a network worm uses some sort of network vehicle. Examples include the following:

Electronic mail facility: A worm mails a copy of itself to other systems.
Remote execution capability: A worm executes a copy of itself on another system.

Remote login capability: A worm logs onto a remote system as a user and then uses commands to copy itself from one system to the other.

The new copy of the worm program is then run on the remote system where, in addition to any functions that it performs at that system, it continues to spread in the same fashion.

A network worm exhibits the same characteristics as a computer virus: a dormant phase, a propagation phase, a triggering phase, and an execution phase. The propagation phase generally performs the following functions:

Search for other systems to infect by examining host tables or similar repositories of remote system addresses.

Establish a connection with a remote system.

Copy itself to the remote system and cause the copy to be run.

As with viruses, network worms are difficult to counter.

The Morris Worm

The Morris worm was designed to spread on UNIX systems and used a number of different techniques for propagation.
1. It attempted to log on to a remote host as a legitimate user. In this method, the worm first attempted to crack the local password file, and then used the discovered passwords and corresponding user IDs. The assumption was that many users would use the same password on different systems. To obtain the passwords, the worm ran a password- cracking program that tried
Each user's account name and simple permutations of it

A list of 432 built-in passwords that Morris thought to be likely candidates c. All the words in the local system directory
2. It exploited a bug in the finger protocol, which reports the whereabouts of a remote user. 3. It exploited a trapdoor in the debug option of the remote process that receives and sends mail.
If any of these attacks succeeded, the worm achieved communication with the operating system command interpreter.

Recent Worm Attacks

In late 2001, a more versatile worm appeared, known as Nimda. Nimda spreads by multiple mechanisms:
from client to client via e-mail

from client to client via open network shares

from Web server to client via browsing of compromised Web sites

from client to Web server via active scanning for and exploitation of various Microsoft IIS 4.0 / 5.0 directory traversal vulnerabilities

from client to Web server via scanning for the back doors left behind by the "Code Red II" worms
The worm modifies Web documents (e.g., .htm, .html, and .asp files) and certain executable files found on the systems it infects and creates numerous copies of itself under various filenames.In early 2003, the SQL Slammer worm appeared. This worm exploited a buffer overflow vulnerability in Microsoft SQL server.
Mydoom is a mass-mailing e-mail worm that appeared in 2004

The ideal solution to the threat of viruses is prevention: The next best approach is to be able to do the following:

Detection: Once the infection has occurred, determine that it has occurred and locate the virus.

Identification: Once detection has been achieved, identify the specific virus that has infected a program.

Removal: Once the specific virus has been identified, remove all traces of the virus from the infected program and restore it to its original state. Remove the virus from all infected systems so that the disease cannot spread further.
If detection succeeds but either identification or removal is not possible, then the alternative is to discard the infected program and reload a clean backup version.
There are four generations of antivirus

software: First generation: simple
scanners
Second generation: heuristic
scanners Third generation: activity
traps

Fourth generation: full-featured protection

A first-generation scanner requires a virus signature to identify a virus.. Such signature-specific scanners are limited to the detection of known viruses. Another type of first-generation scanner maintains a record of the length of programs and looks for changes in length.

A second-generation scanner does not rely on a specific signature. Rather, the scanner uses heuristic rules to search for probable virus infection. One class of such scanners looks for fragments of code that are often associated with viruses.

Another second-generation approach is integrity checking. A checksum can be appended to each program. If a virus infects the program without changing the checksum, then an integrity check will catch the change. To counter a virus that is sophisticated enough to change the checksum when it infects a program, an encrypted hash function can be used. The encryption key is stored separately from the program so that the virus cannot generate a new hash code and encrypt that. By using a hash function rather than a simpler checksum, the virus is prevented from adjusting the program to produce the same hash code as before.

Third-generation programs are memory-resident programs that identify a virus by its actions rather than its structure in an infected program. Such programs have the advantage that it is not necessary to develop signatures and heuristics for a wide array of viruses. Rather, it is necessary only to identify the small set of actions that indicate an infection is being attempted and then to intervene.

Fourth-generation products are packages consisting of a variety of antivirus techniques used in conjunction. These include scanning and activity trap components. In addition, such a package includes access control capability, which limits the ability of viruses to penetrate a system and then limits the ability of a virus to update files in order to pass on the infection.

The arms race continues. With fourth-generation packages, a more comprehensive defense strategy is employed, broadening the scope of defense to more general-purpose computer security measures.

Advanced Antivirus Techniques

More sophisticated antivirus approaches and products continue to appear. In this subsection, we highlight two of the most important.

Generic Decryption

Generic decryption (GD) technology enables the antivirus program to easily detect even the most complex polymorphic viruses, while maintaining fast scanning speeds . In order to detect such a structure, executable files are run through a GD scanner, which contains the following elements:

CPU emulator: A software-based virtual computer. Instructions in an executable file are interpreted by the emulator rather than executed on the underlying processor. The emulator includes software versions of all registers and other processor hardware, so that the underlying processor is unaffected by programs interpreted on the emulator.

Virus signature scanner: A module that scans the target code looking for known virus signatures.

Emulation control module: Controls the execution of the target code.

Digital Immune System

The digital immune system is a comprehensive approach to virus protection developed by IBM]. The motivation for this development has been the rising threat of Internet-based virus propagation.Two major trends in Internet technology have had an increasing impact on the rate of virus propagation in recent years:

Integrated mail systems: Systems such as Lotus Notes and Microsoft Outlook make it very simple to send anything to anyone and to work with objects that are received.

Mobile-program systems: Capabilities such as Java and ActiveX allow programs to move on their own from one system to another.

A monitoring program on each PC uses a variety of heuristics based on system behavior, suspicious changes to programs, or family signature to infer that a virus may be present. The monitoring program forwards a copy of any program thought to be infected to an administrative machine within the organization.

The administrative machine encrypts the sample and sends it to a central virus analysis machine.

This machine creates an environment in which the infected program can be safely run for analysis. Techniques used for this purpose include emulation, or the creation of a protected environment within which the suspect program can be executed and monitored. The virus analysis machine then produces a prescription for identifying and removing the virus.

The resulting prescription is sent back to the administrative machine.

The administrative machine forwards the prescription to the infected client.

The prescription is also forwarded to other clients in the organization.

Subscribers around the world receive regular antivirus updates that protect them from the new virus.
The success of the digital immune system depends on the ability of the virus analysis machine to detect new and innovative virus strains. By constantly analyzing and monitoring the viruses found in the wild, it should be possible to continually update the digital immune software to keep up with the threat.

Behavior-Software

Unlike heuristics or fingerprint-based scanners, behavior-blocking software integrates with the operating system of a host computer and monitors program behavior in real-time for malicious actions. Monitored behaviors can include the following:

Attempts to open, view, delete, and/or modify files;

Attempts to format disk drives and other unrecoverable disk
operations; Modifications to the logic of executable files or macros;
Modification of critical system settings, such as start-up settings;

Scripting of e-mail and instant messaging clients to send executable content;
and Initiation of network communications.

If the behavior blocker detects that a program is initiating would-be malicious behaviors as it runs, it can block these behaviors in real-time and/or terminate the offending software. This gives it a fundamental advantage over such established antivirus detection techniques as fingerprinting or heuristics.