Module 4

Intruders , Intrusion Detection.
MALICIOUS Software: Viruses and related threats, Virus Counter measures

A significant security problem is unwanted, trespass by users or software. User trespass can take the form of unauthorized logon to a machine or, in the case of an authorized user, acquisition of privileges or performance of actions beyond those that have been authorized. Software trespass can take the form of a virus, worm, or Trojan horse.

4.1 INTRUDERS

- One of the two most publicized threats to security is the intruder (the other is viruses), often referred to as a hacker or cracker. The three classes of intruders are:
  - ✓ **Masquerader:** An individual who is not authorized to use the computer and who penetrates a system's access controls to exploit a legitimate user's account
  - ✓ **Misfeasor**: A legitimate user who accesses data, programs, or resources for which such access is not authorized, or who is authorized for such access but misuses his or her privileges
  - ✓ **Clandestine user**: An individual who seizes supervisory control of the system and uses this control to evade auditing and access controls or to suppress audit collection
- The masquerader is likely to be an outsider; the misfeasor generally is an insider; and the clandestine user can be either an outsider or an insider.
- Intruder attacks range from the benign to the serious. At the benign end of the scale, there are many people who simply wish to explore internets and see what is out there. At the serious end are individuals who are attempting to read privileged data, perform unauthorized modifications to data, or disrupt the system.
- The following are examples of intrusion:
  - ✓ Performing a remote root compromise of an e-mail server
  - ✓ Defacing a Web server
  - ✓ Guessing and cracking passwords
  - ✓ Copying a database containing credit card numbers
  - ✓ Viewing sensitive data, including payroll records and medical information, without authorization
  - ✓ Running a packet sniffer on a workstation to capture usernames and passwords
  - ✓ Using a permission error on an anonymous FTP server to distribute pirated software and music files
  - ✓ Dialing into an unsecured modem and gaining internal network access

- ✓ Posing as an executive, calling the help desk, resetting the executive's e-mail password, and learning the new password
- ✓ Using an unattended, logged-in workstation without permission.

❖ Intruder Behavior Patterns

- ➢ The techniques and behavior patterns of intruders are constantly shifting, to exploit newly discovered weaknesses and to evade detection and countermeasures. Intruders typically follow one of a number of recognizable behavior patterns, and these patterns typically differ from those of ordinary users.
- ➢ The three broad classes of Intruder Behavior Patterns are
  - (a) Hacker
  - (b) Criminal Enterprise
  - (c) Internal Threat

(a) HACKERS

- ➢ Traditionally, those who hack into computers do so for the thrill of it or for status. The hacking community is a strong meritocracy in which status is determined by level of competence. Thus, attackers often look for targets of opportunity and then share the information with others.
- ➢ A typical example is a break-in at a large financial institution. The intruder took advantage of the fact that the corporate network was running unprotected services, some of which were not even needed.
- ➢ In this case, the key to the break-in was the pcAnywhere application. The manufacturer, Symantec, advertises this program as a remote control solution that enables secure connection to remote devices.
- ➢ But the attacker had an easy time gaining access to pcAnywhere; the administrator used the same three-letter username and password for the program. In this case, there was no intrusion detection system on the 700-node corporate network.
- ➢ The intruder was only discovered when a vice president walked into her office and saw the cursor moving files around on her Windows workstation.
- ➢ Benign intruders might be tolerable, although they do consume resources and may slow performance for legitimate users. However, there is no way in advance to know whether an intruder will be benign or malign.
- ➢ Intrusion detection systems (IDSs) and intrusion prevention systems (IPSs) are designed to counter this type of hacker threat. In addition to using such systems, organizations can consider restricting remote logons to specific IP addresses and/or use virtual private network technology.

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

(a) Examples of intruder patterns of behavior-HACKERS

1. Select the target using IP lookup tools such as NSLookup, Dig, and others.

2. Map network for accessible services using tools such as NMAP.

3. Identify potentially vulnerable services (in this case, pcAnywhere).

4. Brute force (guess) pcAnywhere password.

5. Install remote administration tool called DameWare.

6. Wait for administrator to log on and capture his password.

7. Use that password to access remainder of network

(b) CRIMINAL ENTERPRISE

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

- Whereas traditional hackers look for targets of opportunity, criminal hackers usually have specific targets, or at least classes of targets in mind. Once a site is penetrated, the attacker acts quickly, scooping up as much valuable information as possible and exiting.
- IDSs and IPSs can also be used for these types of attackers, but may be less effective because of the quick in-and-out nature of the attack. For e-commerce sites, database encryption should be used for sensitive customer information, especially credit cards. For hosted e-commerce sites (provided by an outsider service), the e-commerce organization should make use of a dedicated server (not used to support multiple customers) and closely monitor the provider's security services.

(b) Examples of intruder patterns of behavior -CRIMINAL ENTERPRISE

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

(c) INSIDER ATTACKS

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

countering insider attacks, other more direct approaches are of higher priority. Examples include the following:

• Enforce least privilege, only allowing access to the resources employees need to do their job.

• Set logs to see what users access and what commands they are entering.

• Protect sensitive resources with strong authentication.

• Upon termination, delete employee's computer and network access.

• Upon termination, make a mirror image of employee's hard drive before reissuing it. That evidence might be needed if your company information turns up at a competitor.

(c) Examples of intruder patterns of behavior -INTERNAL THREATS/INSIDER ATTACKS

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

❖ INTRUSION TECHNIQUES

➢ The objective of the intruder is to gain access to a system or to increase the range of privileges accessible on a system. Most initial attacks use system or software vulnerabilities that allow a user to execute code that opens a back door into the system.

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

➢ 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 and learn passwords. The password file can be protected in one of two ways:

✓ **One-way function:** The system stores only the value of a function based on the user's password. When the user presents a password, the system transforms that password and compares it with the stored value. In practice, the system usually performs a one-way transformation (not reversible) in which the password is used to generate a key for the one-way 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. If one or both of these countermeasures are in place, some effort is needed for a potential intruder to learn passwords. On the basis of a survey of the literature and interviews with a number of password crackers/hackers, reports the following techniques for learning passwords:

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

  2. Exhaustively try all short passwords (those of one to three characters).

  3. Try words in the system's online dictionary or a list of likely passwords. Examples of the latter are readily available on hacker bulletin boards.

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

  5. Try users' phone numbers, Social Security numbers, and room numbers.

  6. Try all legitimate license plate numbers for this state.

  7. Use a Trojan horse to bypass restrictions on access.

  8. Tap the line between a remote user and the host system.

- ➢ The first six methods are various ways of guessing a password. If an intruder has to verify the guess by attempting to log in, it is a tedious and easily countered means of attack. For example, a system can simply reject any login after three password attempts, thus requiring the intruder to reconnect to the host to try again. Under these circumstances, it is not practical to try more than a handful of passwords. However, the intruder is unlikely to try such crude methods.
- ➢ For example, if an intruder can gain access with a low level of privileges to an encrypted password file, then the strategy would be to capture that file and then use the encryption mechanism of that particular system at leisure until a valid password that provided greater privileges was discovered. Guessing attacks are feasible, and indeed highly effective, when a large number of guesses can be attempted automatically and each guess verified, without the guessing process being detectable
- ➢ The seventh method of attack listed above, the Trojan horse, can be particularly difficult to counter. A low-privilege user produced a game program and invited the system operator to use it in his or her spare time. The program did indeed play a game, but in the background, it also contained code to copy the password file, which was unencrypted but access protected, into the user's file. Because the game was running under the operator's high-privilege mode, it was able to gain access to the password file.

- The eighth attack listed, line tapping, is a matter of physical security. Other intrusion techniques do not require learning a password. Intruders can get access to a system by exploiting attacks such as buffer overflows on a program that runs with certain privileges. Privilege escalation can be done this way as well.

4.2 INTRUSION DETECTION

The two principal countermeasures for intruders are: detection and prevention.

- Detection is concerned with learning of an attack, either before or after its success.
- Prevention is a challenging security goal and an uphill battle at all times.

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, such as:

- ✓ If an intrusion is detected quickly enough, the intruder can be identified and ejected from the system before any damage is done or any data are compromised. Even if the detection is not sufficiently timely to preempt the intruder, the sooner that the intrusion is detected, the less the amount of damage and the more quickly that recovery can be achieved.
- ✓ An effective intrusion detection system can serve as a deterrent, acting to prevent intrusions.
- ✓ 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. Of course, there will not be a crisp, exact distinction between an attack by an intruder and the normal use of resources by an authorized user. Rather, there will be some overlap.
- ✓ Figure 4.1 suggests, in very abstract terms, the nature of the task confronting the designer of an intrusion detection system.

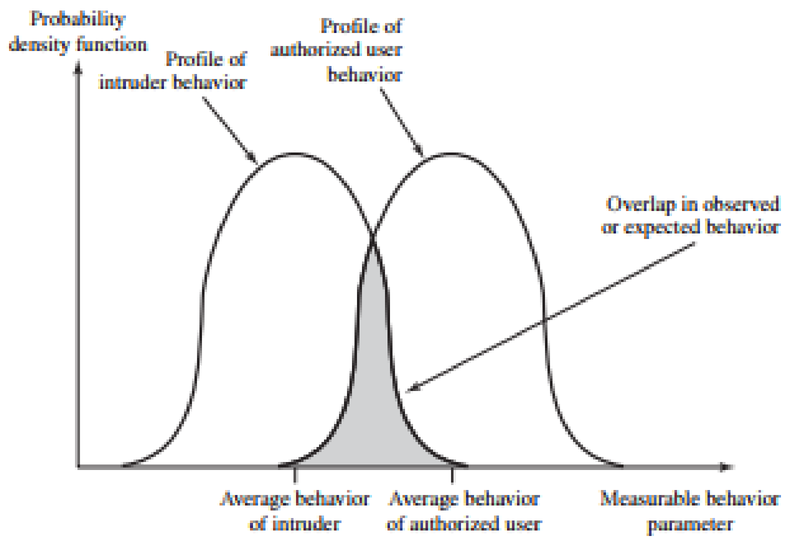

Probability density function
Profile of intruder behavior
Profile of authorized user behavior
Overlap in observed or expected behavior
Average behavior of intruder
Average behavior of authorized user
Measurable behavior parameter

Fig 4.1 Profiles of Behavior of Intruders and Authorized Users

- 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.
- It was postulated that one could, with reasonable confidence, distinguish between a masquerader and a legitimate user. Patterns of legitimate user behavior can be established by observing past history, and significant deviation from such patterns can be detected.
- The task of detecting a misfeasor (legitimate user performing in an unauthorized fashion) is more difficult, in that the distinction between abnormal and normal behavior may be small.
- However, misfeasor behavior might nevertheless be detectable by intelligent definition of the class of conditions that suggest unauthorized use. Finally, the detection of the clandestine user was felt to be beyond the scope of purely automated techniques. These observations, which were made in 1980, remain true today.

➢ The following are the 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.

- ✓ **Threshold detection**: This approach involves defining thresholds, independent of user, for the frequency of occurrence of various events.
- ✓ **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.

- ✓ **Anomaly detection**: Rules are developed to detect deviation from previous usage patterns.
- ✓ **Penetration identification:** An expert system approach that searches for suspicious behavior.

- In a nutshell, statistical approaches attempt to define normal, or expected, behavior, whereas rule-based approaches attempt to define proper behavior.
- In terms of the types of attackers listed earlier, statistical anomaly detection is effective against masqueraders, who are unlikely to mimic the behavior patterns of the accounts they appropriate.
- On the other hand, such techniques may be unable to deal with misfeasors. For such attacks, 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. All activity arises through commands issued by subjects. Subjects may be grouped into different access classes, and these classes may overlap.

• **Action:** Operation performed by the subject on or with an object; for example, login, read, perform I/O, execute.

• **Object:** Receptors of actions. Examples include files, programs, messages, records, terminals, printers, and user- or program-created structures. When a subject is the recipient of an action, such as electronic mail, then that subject is considered an object. Objects may be grouped by type.

• **Exception-Condition**: Denotes which, if any, exception condition is raised on return.

• **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.

❖ Statistical Anomaly Detection

Statistical anomaly detection techniques fall into two broad categories: threshold detection and profile-based systems.

a) Threshold detection

- Involves counting the number of occurrences of a specific event type over an interval of time. If the count surpasses what is considered a reasonable number that one might expect to occur, then intrusion is assumed.
- Threshold analysis, by itself, is a crude and ineffective detector of even moderately sophisticated attacks. Both the threshold and the time interval must be determined. Because of the variability across users, such thresholds are likely to generate either a lot of false positives or a lot of false negatives.
- However, simple threshold detectors may be useful in conjunction with more sophisticated techniques.

b) Profile-based anomaly detection

- Focuses on characterizing the past behavior of individual users or related groups of users and then detecting significant deviations. A profile may consist of a set of parameters, so that deviation on just a single parameter may not be sufficient in itself to signal an alert.
- The foundation of this approach is an analysis of audit records. The audit records provide input to the intrusion detection function in two ways.
  - ✓ First, the designer must decide on a number of quantitative metrics that can be used to measure user behavior. An analysis of audit records over a period of time can be

used to determine the activity profile of the average user. Thus, the audit records serve to define typical behavior.

- ✓ Second, current audit records are the input used to detect intrusion. That is, the intrusion detection model analyzes incoming audit records to determine deviation from average behavior.
- ✓ Examples of metrics that are useful for profile-based intrusion detection are the following:

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

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

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

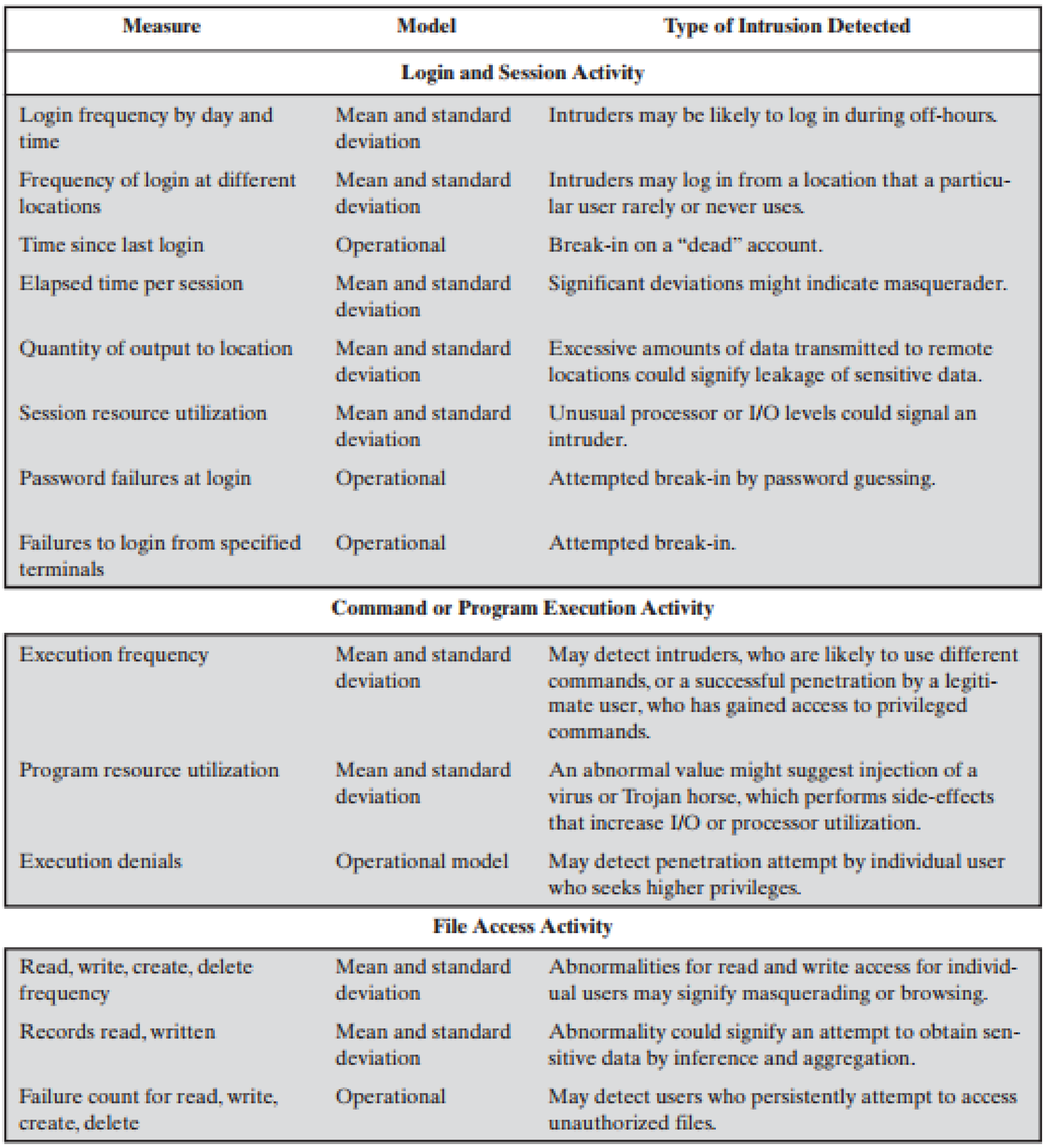
  • **Resource utilization**: Quantity of resources consumed during a specified period. Examples include the number of pages printed during a user session and total time consumed by a program execution.

- ➢ Given these general metrics, various tests can be performed to determine whether current activity fits within acceptable limits. The following approaches may be taken:
  - Mean and standard deviation
  - Multivariate
  - Markov process
  - Time series
  - Operational
- ➢ The simplest statistical test is to measure the **mean and standard deviation** of a parameter over some historical period. This gives a reflection of the average behavior and its variability. The use of mean and standard deviation is applicable to a wide variety of counters, timers, and resource measures. But these measures, by themselves, are typically too crude for intrusion detection purposes.
- ➢ A **multivariate model** is based on correlations between two or more variables. Intruder behavior may be characterized with greater confidence by considering such correlations

(for example, processor time and resource usage, or login frequency and session elapsed time).

- A **Markov process model** is used to establish transition probabilities among various states. As an example, this model might be used to look at transitions between certain commands.
- A **time series model** focuses on time intervals, looking for sequences of events that happen too rapidly or too slowly. A variety of statistical tests can be applied to characterize abnormal timing.
- An **operational model** is based on a judgment of what is considered abnormal, rather than an automated analysis of past audit records. Typically, fixed limits are defined and intrusion is suspected for an observation that is outside the limits. This type of approach works best where intruder behavior can be deduced from certain types of activities. For example, a large number of login attempts over a short period suggests an attempted intrusion.
- As an example of the use of these various metrics and models, the following table shows various measures considered or tested for the Stanford Research Institute (SRI) intrusion detection system (IDES)

Measures That May Be Used for Intrusion Detection



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

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

❖ 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.
- ➢ We can characterize all approaches as focusing on either anomaly detection or penetration identification, although there is some overlap in these approaches.
- ➢ **Rule-based anomaly detection** is similar in terms of its approach and strengths to statistical anomaly detection. With the rule-based approach, historical audit records are analyzed to identify usage patterns and to generate automatically rules that describe those patterns.
- ➢ Rules may represent past behavior patterns of users, programs, privileges, time slots, terminals,etc. 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. In order for this approach to be effective, a rather large database of rules will be needed.
- ➢ **Rule-based penetration identification** takes a very different approach to intrusion detection. The key feature of such systems is the use of rules for identifying known penetrations or penetrations that would exploit known weaknesses.
- ➢ Rules can also be defined that identify suspicious behavior, even when the behavior is within the bounds of established patterns of usage. Typically, the rules used in these systems are specific to the machine and operating system. The best approach to developing such rules is to analyze attack tools and scripts collected on the Internet.
- ➢ These rules can be supplemented with rules generated by knowledgeable security personnel. In this latter case, the normal procedure is to interview system administrators and security analysts to collect a suite of known penetration scenarios and key events that threaten the security of the target system.
- ➢ A simple example of the type of rules that can be used is found in NIDX, an early system that used heuristic rules that can be used to assign degrees of suspicion to activities. Example heuristics are the following:

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

  2. Users must not write other users' files.

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

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

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

  6. Users do not make copies of system programs.

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

❖ **The 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.

❖ Distributed Intrusion Detection

- Until recently, work on intrusion detection systems focused on single-system standalone facilities. The typical organization, however, needs to defend a distributed collection of hosts supported by a LAN or internetwork.
- Although it is possible to mount a defense by using stand-alone intrusion detection systems on each host, a more effective defense can be achieved by coordination and cooperation among intrusion detection systems across the network.
- The following are the 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. Integrity is required to prevent an intruder from masking his or her activities by altering the transmitted audit information. Confidentiality is required because the transmitted audit information could be valuable.

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

➢ A good example of a distributed intrusion detection system is one developed at the University of California at Davis. Figure 4.2 shows the overall architecture, which consists of three main components:
  - ✓ **Host agent module**: An audit collection module operating as a background process on a monitored system. Its purpose is to collect data on security related events on the host and transmit these to the central manager.
  - ✓ **LAN monitor agent module**: Operates in the same fashion as a host agent module except that it analyzes LAN traffic and reports the results to the central manager.
  - ✓ **Central manager module**: Receives reports from LAN monitor and host agents and processes and correlates these reports to detect intrusion.

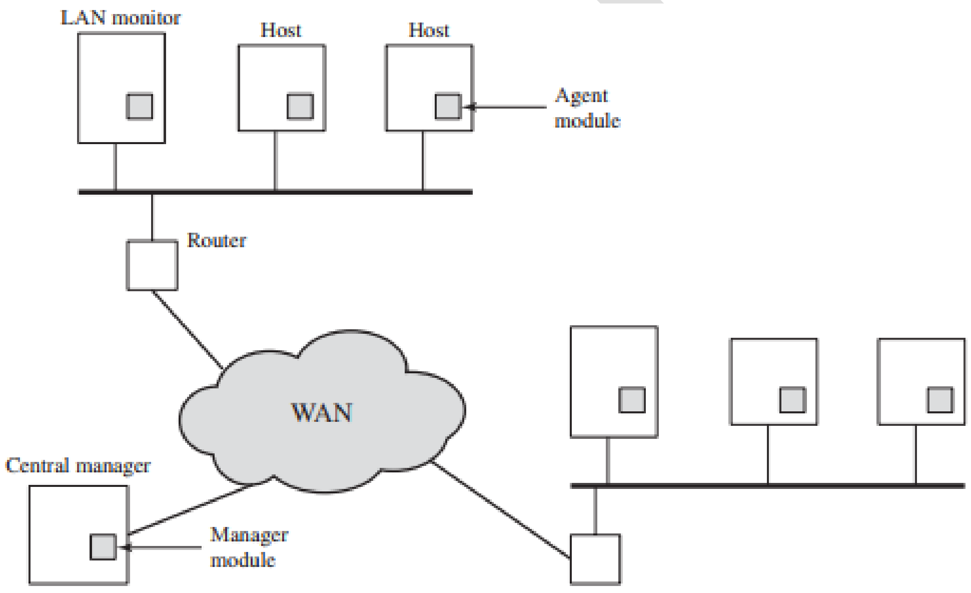


Fig 4.2 Architecture for Distributed Intrusion Detection

➢ The scheme is designed to be independent of any operating system or system auditing implementation. Figure 4.3 shows the general approach that is taken.

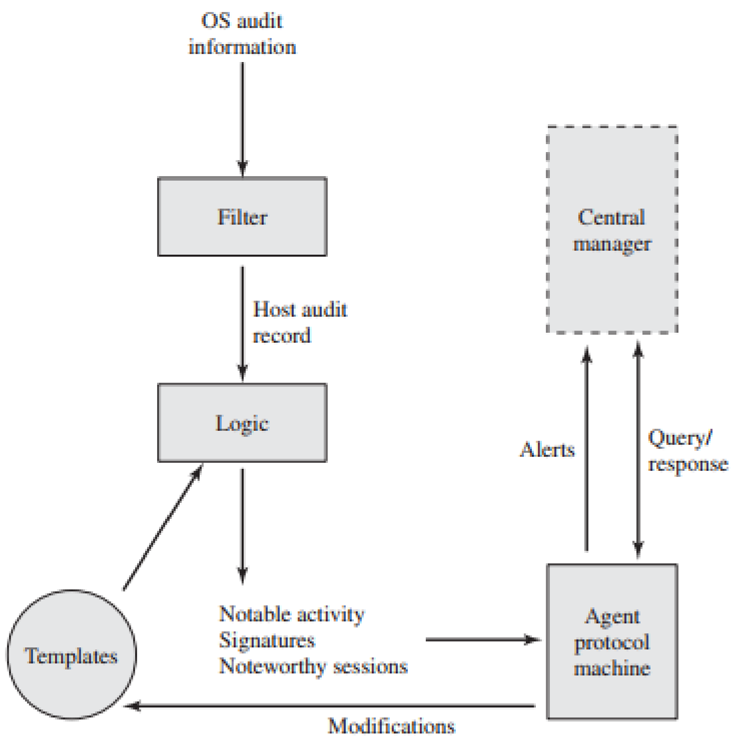


Fig 4.3 Agent Architecture

- The agent captures each audit record produced by the native audit collection system. A filter is applied that retains only those records that are of security interest.
- These records are then reformatted into a standardized format referred to as the host 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. It searches for significant events, such as sudden changes in network load, the use of security-related services, and network activities such as rlogin.

❖ Honeypots

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

  • divert an attacker from accessing critical systems

  • collect information about the attacker's activity

  • encourage the attacker to stay on the system long enough for administrators to respond

- These systems are filled with fabricated information designed to appear valuable but that a legitimate user of the system wouldn't access. Thus, any access to the honeypot is suspect.
- The system is instrumented with sensitive monitors and event loggers that detect these accesses and collect information about the attacker's activities. Because any attack against the honeypot is made to seem successful, administrators have time to mobilize and log and track the attacker without ever exposing productive systems.
- Initial efforts involved a single honeypot computer with IP addresses designed to attract hackers. More recent research has focused on building entire honeypot networks that emulate an enterprise, possibly with actual or simulated traffic and data.
- Once hackers are within the network, administrators can observe their behavior in detail and figure out defenses.

❖ 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 purpose of the working group is to define data formats and exchange procedures for sharing information of interest to intrusion detection and response systems and to management systems that may need to interact with them.
- The outputs of this working group include:
  1. A requirements document, which describes the high-level functional requirements for communication between intrusion detection systems and requirements for communication between intrusion detection systems and with management systems, including the rationale for those requirements. Scenarios will be used to illustrate the requirements.
  2. A common intrusion language specification, which describes data formats that satisfy the requirements.

3. A framework document, which identifies existing protocols best used for communication between intrusion detection systems, and describes how the devised data formats relate to them.

MALICIOUS Software

- Malicious software is software that is intentionally included or inserted in a system for a harmful purpose. The most sophisticated types of threats to computer systems are presented by programs that exploit vulnerabilities in computing systems. Such threats are referred to as **malicious software**, or **malware.**
- Malicious software can be divided into two categories:
- Those that need a host program, and those that are independent. The former, referred to as parasitic, 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.
- Independent malware is a self-contained program that can be scheduled and run by the operating system. Worms and bot programs are examples

4.3 VIRUSES

❖ The Nature of Viruses

- A computer virus is a piece of software that can “infect” other programs by modifying them; the modification includes injecting the original program with a routine to make copies of the virus program, which can then go on to infect other programs.
- Biological viruses are tiny scraps of genetic code—DNA or RNA—that can take over the machinery of a living cell and trick it into making thousands of flawless replicas of the original virus.
- Like its biological counterpart, a computer virus carries in its instructional code the recipe for making perfect copies of itself. The typical virus becomes embedded in a program on a computer. Then, whenever the infected computer comes into contact with an uninfected piece of software, a fresh copy of the virus passes into the new program. Thus, the infection can be spread from computer to computer by unsuspecting users who either swap disks or send programs to one another over a network.
- In a network environment, the ability to access applications and system services on other computers provides a perfect culture for the spread of a virus.
- A virus can do anything that other programs do. The difference is that a virus 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 that is allowed by the privileges of the current user.

A computer virus has three parts :

- ✓ **Infection mechanism**: The means by which a virus spreads, enabling it to replicate. The mechanism is also referred to as the infection vector.
- ✓ **Trigger:** The event or condition that determines when the payload is activated or delivered.
- ✓ **Payload:** What the virus does, besides spreading. The payload may involve damage or may involve benign but noticeable activity.

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 a copy of itself into other programs or into certain system areas on the disk. The copy may not be identical to the propagating version; viruses often morph to evade detection. 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. 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.
- A very general depiction of virus structure is shown in Figure 4.4. In this case, the virus code, V, is prepended to infected programs, and it is assumed that the entry point to the program, when invoked, is the first line of the program.
- The 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 may first seek out uninfected executable files and infect them.

```
    program V :=

{goto main;
    1234567;

    subroutine infect-executable :=
        {loop:
        file := get-random-executable-file;
        if (first-line-of-file = 1234567)
            then goto loop
            else prepend V to file; }

    subroutine do-damage :=
        {whatever damage is to be done}

    subroutine trigger-pulled :=
        {return true if some condition holds}

main:   main-program :=
        {infect-executable;
        if trigger-pulled then do-damage;
        goto next;}
next:

}
```

Fig 4.4 A simple Virus

- ✓ 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 an uninfected program.
- ✓ A virus like thus 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. Figure 4.5 illustrates the operation. We assume that program P1 is infected with the virus CV.

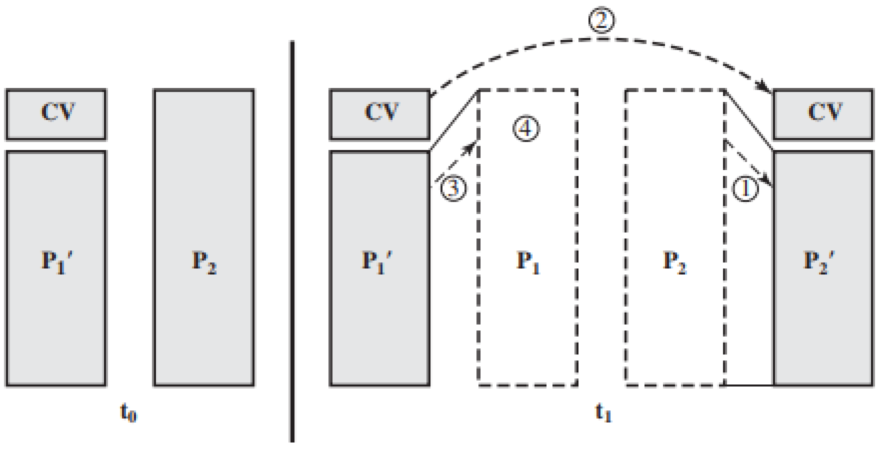


Fig 4.5 A compression Virus

When this program is invoked, control passes to its virus, which performs the following steps:

1. For each uninfected file P2 that is found, the virus first compresses that file to produce P2**’** which is shorter than the original program by the size of the virus.
2. A copy of the virus is prepended to the compressed program.
3. The compressed version of the original infected program, is uncompressed.
4. The uncompressed original program is executed.

➢ INITIAL INFECTION

- Once a virus has gained entry to a system by infecting a single program, it is in a position to potentially 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. Many forms of infection can also be blocked by denying normal users the right to modify programs on the system.
- The lack of access controls on early PCs is a key reason why traditional machine code based viruses spread rapidly on these systems.
- In contrast, while it is easy enough to write a machine code virus for UNIX systems, they were almost never seen in practice because the existence of access controls on these systems prevented effective propagation of the virus.
- Traditional machine code based viruses are now less prevalent, because modern PC OSs do have more effective access controls. However, virus creators have found other avenues, such as macro and e-mail viruses.

➢ Viruses Classification

Viruses can be classified along two orthogonal axes: the type of target the virus tries to infect and the method the virus uses to conceal itself from detection by users and antivirus software.

A virus classification by target includes the following categories:

- **Boot sector infector**: Infects a master boot record or boot record and spreads when a system is booted from the disk containing the virus.
- **File infector**: Infects files that the operating system or shell consider to be executable.
- **Macro virus**: Infects files with macro code that is interpreted by an application.

A virus classification by concealment strategy includes the following categories:

- **Encrypted virus**: A portion of the virus creates a random encryption key and encrypts the remainder of the virus. The key is stored with the virus. When an infected program is invoked, the virus uses the stored random key to decrypt the virus.
- **Stealth virus**: A form of virus explicitly designed to hide itself from detection by antivirus software. Thus, the entire virus, not just a payload is hidden.
- **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 may change their behavior as well as their appearance.

➢ Virus Kit

A virus toolkit enables to quickly create a number of different viruses. Although viruses created with toolkits tend to be less sophisticated than viruses designed from scratch, the sheer number of new viruses that can be generated using a toolkit creates a problem for antivirus schemes.

➢ 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. Many macro viruses infect Microsoft Word documents or other Microsoft Office documents. Any hardware platform and operating system that supports these applications can be infected.
  2. Macro viruses infect documents, not executable portions of code. Most of the 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.
  4. Because macro viruses infect user documents rather than system programs, traditional file system access controls are of limited use in preventing their spread.
- 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.
- Successive releases of MS Office products 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 on the user's system.

- In 1999, a more powerful version of the e-mail virus appeared. This newer version can be activated merely by opening an e-mail that contains the virus rather than opening an attachment.
- The virus uses the Visual Basic scripting language supported by the e-mail package. This is a new generation of malware that arrives via e-mail and uses e-mail software features to replicate itself across the Internet.
- The virus propagates itself as soon as it is activated (either by opening an e-mail attachment or by opening the e-mail) to all of the e-mail addresses known to the infected host.
- As a result, whereas viruses used to take months or years to propagate, they now do so in hours. This makes it very difficult for antivirus software to respond before much damage is done.
- Ultimately, a greater degree of security must be built into Internet utility and application software on PCs to counter the growing threat.

➢ VIRUS COUNTERMEASURES

Antivirus Approaches

- The ideal solution to the threat of viruses is prevention: Do not allow a virus to get into the system in the first place, or block the ability of a virus to modify any files containing executable code or macros.
- This goal is, in general, impossible to achieve, although prevention can reduce the number of successful viral attacks. 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 virus cannot spread further.

- If detection succeeds but either identification or removal is not possible, then the alternative is to discard the infected file and reload a clean backup version.
- Advances in virus and antivirus technology go hand in hand. Early viruses were relatively simple code fragments and could be identified and purged with relatively simple antivirus software packages.
- As the virus arms race has evolved, both viruses and, necessarily, antivirus software have grown more complex and sophisticated. The 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. The virus may contain "wildcards" but has essentially the same structure and bit pattern in all copies. 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. For example, a scanner may look for the beginning of an encryption loop used in a polymorphic virus and discover the encryption key. Once the key is discovered, the scanner can decrypt the virus to identify it, then remove the infection and return the program to service.
- 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.

➢ Advanced Antivirus Techniques

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

a) GENERIC DECRYPTION

- Generic decryption (GD) technology enables the antivirus program to easily detect even the most complex polymorphic viruses while maintaining fast scanning speeds . When a file containing a polymorphic virus is executed, the virus must decrypt itself to activate. 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.
- At the start of each simulation, the emulator begins interpreting instructions in the target code, one at a time. Thus, if the code includes a decryption routine that decrypts and hence exposes the virus, that code is interpreted.
- In effect, the virus does the work for the antivirus program by exposing the virus. Periodically, the control module interrupts interpretation to scan the target code for virus signatures.
- During interpretation, the target code can cause no damage to the actual personal computer environment, because it is being interpreted in a completely controlled environment.
- The most difficult design issue with a GD scanner is to determine how long to run each interpretation. The longer the scanner emulates a particular program, the more likely it is to catch any hidden viruses.

b) DIGITAL IMMUNE SYSTEM

- The digital immune system is an approach to virus protection developed by IBM. The motivation for this development has been the rising threat of Internet-based virus propagation.
- Traditionally, the virus threat was characterized by the relatively slow spread of new viruses and new mutations. Antivirus software was typically updated on a monthly basis, and this was sufficient to control the problem.
- Also traditionally, the Internet played a comparatively small role in the spread of viruses. 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.
- In response to the threat posed by these Internet-based capabilities, IBM has developed a prototype digital immune system.
- The objective of this system is to provide rapid response time so that viruses can be stamped out almost as soon as they are introduced.
- When a new virus enters an organization, the immune system automatically captures it, analyzes it, adds detection and shielding for it, removes it, and passes information about that virus to systems running IBM AntiVirus so that it can be detected before it is allowed to run elsewhere.

Figure 4.6 illustrates the typical steps in digital immune system operation:

1. 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.
2. The administrative machine encrypts the sample and sends it to a central virus analysis machine.
3. 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.
4. The resulting prescription is sent back to the administrative machine.
5. The administrative machine forwards the prescription to the infected client.
6. The prescription is also forwarded to other clients in the organization.

7. Subscribers around the world receive regular antivirus updates that protect them from the new virus.

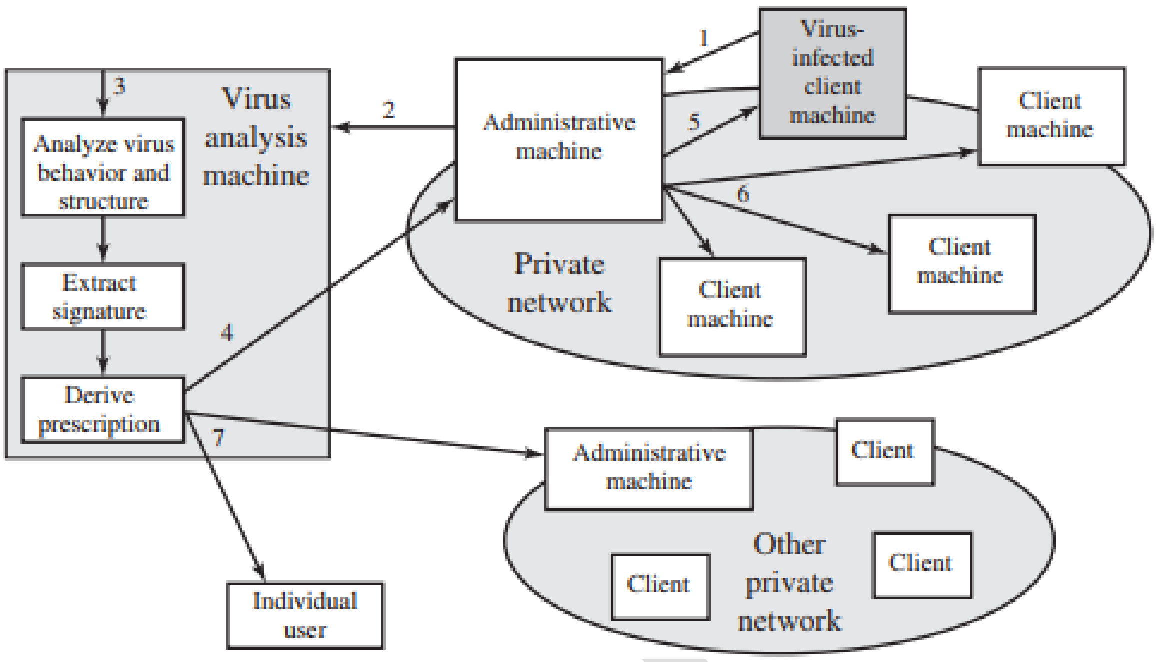


Fig 4.6 Digital Immune System

- 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.

c) BEHAVIOR-BLOCKING SOFTWARE

- Behavior-blocking software integrates with the operating system of a host computer and monitors program behavior in real-time for malicious actions. The behavior blocking software blocks potentially malicious actions before they have a chance to affect the system. Monitored behaviors can include
  - ✓ 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.

- Figure 4.7 illustrates the operation of a behavior blocker. Behavior-blocking software runs on server and desktop computers and is instructed through policies set by the network administrator to let benign actions take place but to intercede when unauthorized or suspicious actions occur.
- The module blocks any suspicious software from executing. A blocker isolates the code in a sandbox, which restricts the code's access to various OS resources and applications. The blocker then sends an alert.
- Because a behavior blocker can block suspicious software in real-time, it has an advantage over such established antivirus detection techniques as fingerprinting or heuristics.
- The behavior blocker can identify and block malicious actions regardless of how complicated the program logic appears to be.
- Behavior blocking alone has limitations. Because the malicious code must run on the target machine before all its behaviors can be identified, it can cause harm before it has been detected and blocked.
- For example, a new virus might shuffle a number of seemingly unimportant files around the hard drive before infecting a single file and being blocked.
- Even though the actual infection was blocked, the user may be unable to locate his or her files, causing a loss to productivity or possibly worse.

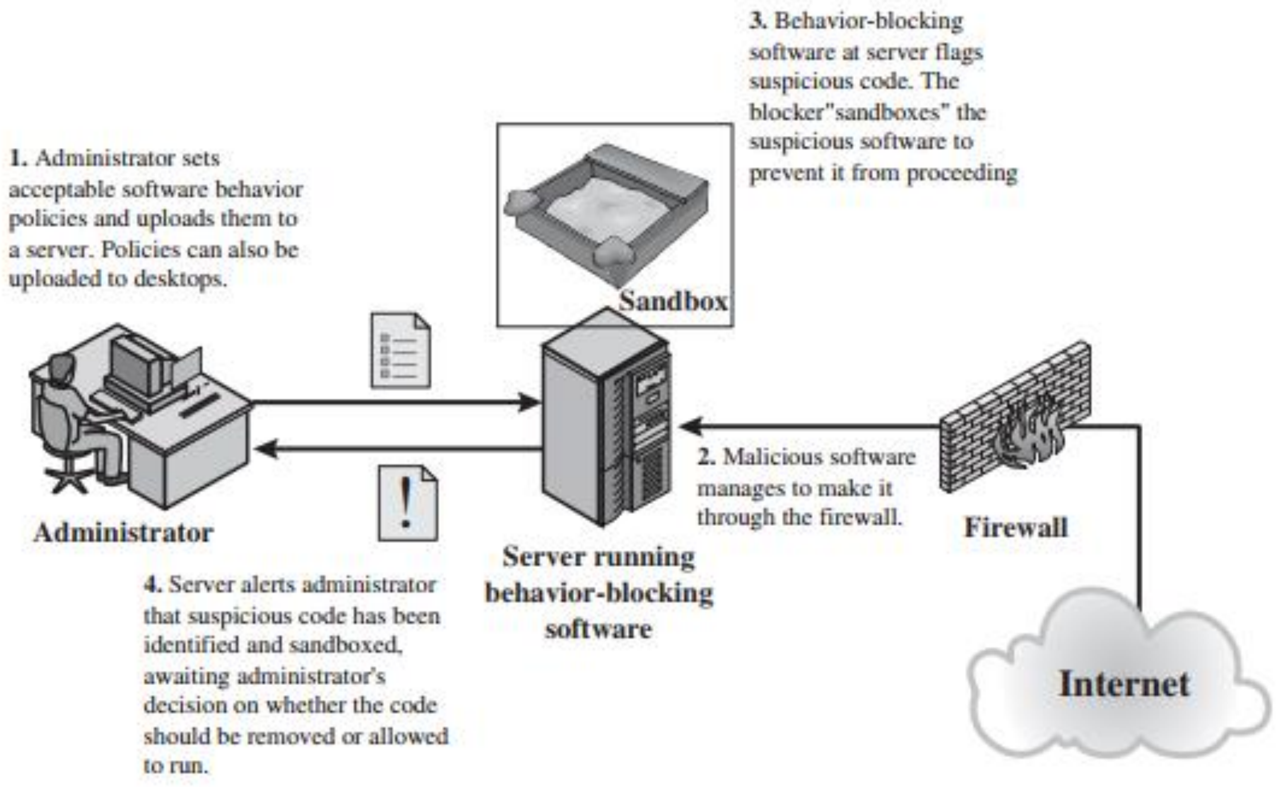

1. Administrator sets acceptable software behavior policies and uploads them to a server. Policies can also be uploaded to desktops.
3. Behavior-blocking software at server flags suspicious code. The blocker"sandboxes" the suspicious software to prevent it from proceeding
Sandbox
Administrator
2. Malicious software manages to make it through the firewall.
Firewall
Server running behavior-blocking software
4. Server alerts administrator that suspicious code has been identified and sandboxed, awaiting administrator's decision on whether the code should be removed or allowed to run.
Internet

Fig 4.7 Behavior-Blocking Software Operation