

# **CHAPTER - 9**

## **INTRUDERS**



# Key Points

- Unauthorized intrusion into a computer system or network is one of the most serious threats to computer security.
- Intrusion detection systems can be used to provide early warning of an intrusion so that defensive action can be taken to prevent or minimize damage.
- Intrusion detection involves detecting unusual patterns of activity or patterns of activity likely to be intrusions.
- One important element of intrusion prevention is password management, to prevent unauthorized users from having access to the passwords of others.



# 9.1 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.
- **Masquerader:** An individual who is not authorized to use the computer but 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



# 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

- Behavior patterns and techniques used by intruders keep shifting to exploit newly discovered weaknesses and to evade countermeasures.
- Following table shows three broad examples of intruder behavior patterns faced by the security administrator.



# Intruder Behavior Patterns

**Table 9.1** Some Examples of Intruder Patterns of Behavior

## **(a) Hacker**

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

## **(b) Criminal Enterprise**

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

## **(c) Internal Threat**

1. Create network accounts for themselves and their friends.
2. Access accounts and applications they wouldn't normally use for their daily jobs.
3. E-mail former and prospective employers.
4. Conduct furtive instant-messaging chats.
5. Visit Web sites that cater to disgruntled employees, such as fdcompany.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.
- Initial attacks use system vulnerabilities to open a back door into the system.
- The intruder may attempt to acquire protected information like users password, log in to the system and exercise all the privileges of the legitimate user.
- So a system must maintain a file that associate a password with its authorized user.
- The password file can be protected in one of two ways:
  - One-way function
  - Access control



# Intrusion Techniques

- **One-way function:** the system stores only the value of a function based on the users password.
- The system performs a one-way transformation(not reversible) of the password to compare with stored value.
- Here password is used to generate a key for the one-way transformation and a fixed length output is produced.
- **Access control:** Access to password file is limited to one or a very few accounts.
- In the presence of one or two measures, an intruder needs more efforts to be put in to learn password.





# Continue...

- A survey on password hackers reports the following techniques for learning passwords.
  - Try default passwords used with standard accounts that are shipped with the system.
  - Exhaustively try all short passwords
  - Try words in the systems online dictionary or likely words
  - Collect users information like full name, family members names/ birth-dates, pictures, hobbies and related books of interest, etc.
  - Try users' phone numbers, Social Security numbers, and room numbers etc.
  - Use a Trojan horse to bypass restrictions on access.



# Continue...

- Other intrusion techniques do not require learning a password. Intruders can exploit conditions such as buffer overflows that runs with certain privileges.
- There are two counter measures:
- **Detection:** learning of an attack either before or after the attack. Then defender must attempt to thwart all the possible attacks.
- **Prevention:** challenging security goal, find weakest link in the defense chain and attack there.



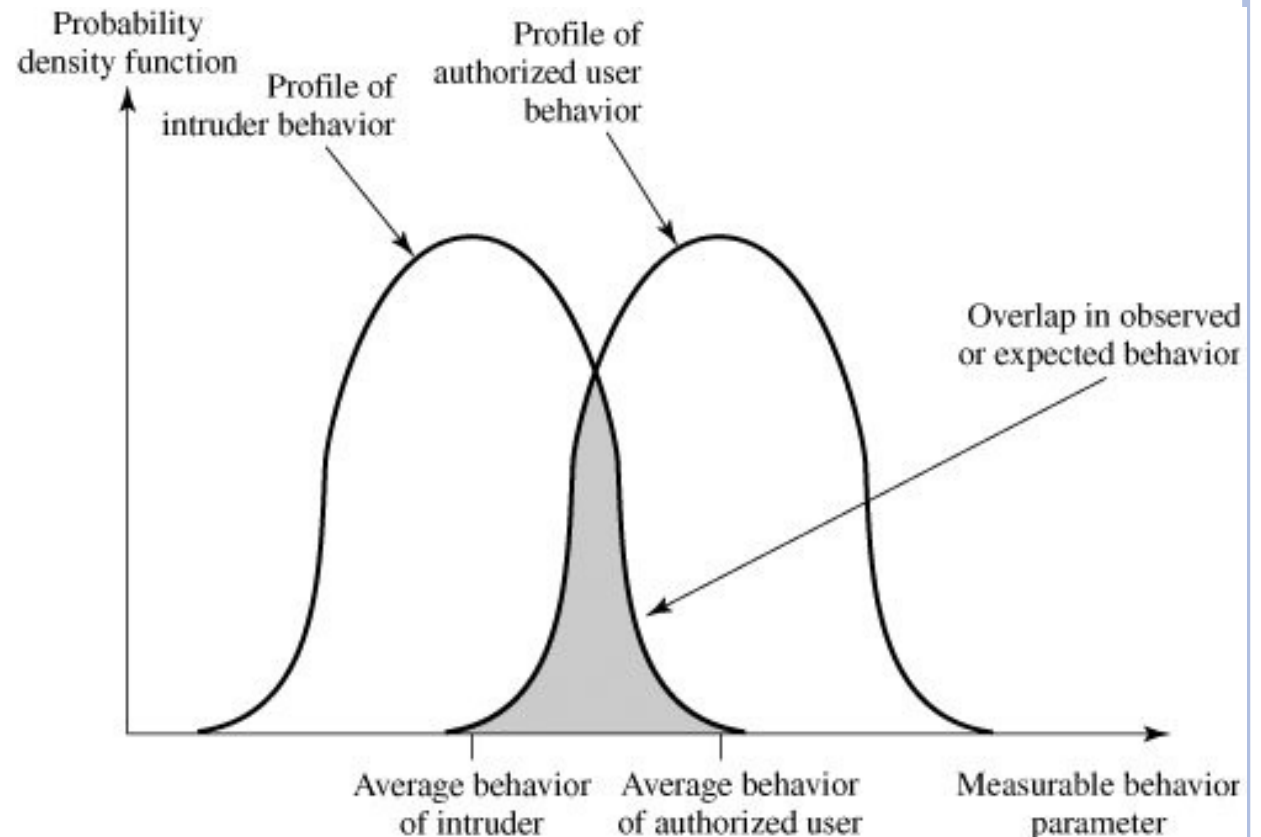
## 9.2 Intrusion Detection

- The best intrusion detection system too can fail.
- Intrusion detection has been the focus of much research in recent years with following considerations:
  - If you quickly identify intrusion, the intruder can be identified and ejected before any damage is done or any data are compromised.
  - Even if the detection can not preempt the intruder, the amount of damage will be less and more quick.
  - Effective intrusion detection system can serve as deterrent, so acting to prevent intrusions.
  - Intrusion detection enables collection of information about intrusion techniques that can be used in intrusion prevention.



# Intrusion Detection

- Intrusion detection is based on the fact that behavior of the intruder differs from that of the legitimate user. But the distinction cannot be crisp if it's an attack or normal use.
- The nature of the task confronting the designer of an intrusion detection system is shown in the following figure.



# Intrusion Detection

- **Statistical anomaly detection:** collect data relating to the behavior of legitimate user over a time and applying statistical tests to determine high level of confidence whether that behavior is legitimate or not.
  - **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
  - **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.



# Intrusion Detection

- **Rule-based detection:** attempt to define a set of rules that can be used to decide that a given behavior of individual accounts.
  - Anomaly detection: rules are developed to detect deviation from previous usage.
  - Penetration identification: an expert system approach that searches for suspicious behavior.
- **Conclusion:** Statistical approaches attempt to define normal, or expected behavior whereas rule-based approaches attempt to define proper behavior.
- Statistical approach is effective against masqueraders while rule-based approach is more suitable for misfeasors.
- 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:** all multi-user OS include accounting s/w that collects user activity records.
  - Advantage: no additional collection s/w is needed
  - Disadvantage: native records may not contain needed information or may not be in the required format.



# Audit Records

- **Detection-specific audit records:** A collection facility can be implemented to collect only the required information by the intrusion detection system.
- Advantage: it can be made vendor independent and ported to a variety of systems.
- Disadvantage: extra overhead with having two accounting systems on machine.





# Audit Records

- Example of detection-specific audit records. Each audit record contains the following fields
  - Subject: typically a terminal user, initiator of the action or might be a process acting on behalf of a user or group of users.
  - Subject can be grouped in different access classes and these classes may overlap
  - Action: Operation performed by the subject on or with object. E.g login, read, I/O, execute, etc.
  - Object: Receptor of actions like files, programs, messages, records, terminals, printers, and user/program created structures.
  - When a subject is recipient of an action, it is considered as object. Objects may be grouped by their type.



# Audit Records

- Exception-Condition: noted if any exception condition is raised on return.
- Resource-Usage: list of quantitative elements with the amount used of some resource. E.g no of lines printed or displayed, no of records read/written, processor time, i/o unit used, session elapsed time.
- Time-Stamp: Unique time-and-date stamp identifying when the action took place.



# Audit Records

- 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.
- E.g User Smith tries to copy an executable file GAME from the current directory to the <Library> directory.

COPY GAME.EXE TO <Libray>GAME.EXE

- 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
-------	---------	-------------------	------------	-------------	-------------



# Audit Records

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



# Statistical Anomaly Detection

- Statistical anomaly detection techniques fall into two broad categories: **Threshold detection** and **Profile-based systems**.
- 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.
- 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.



# Statistical Anomaly Detection

- Examples of metrics that are useful for profile-based intrusion detection are the following:
- **Counter:** A nonnegative integer. E.g 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. E.g 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.



# Statistical Anomaly Detection

- Given these general metrics, various tests can be performed to determine whether current activity fits within acceptable limits. lists the following approaches that may be taken:
  - Mean and standard deviation
  - Multivariate
  - Markov process
  - Time series
  - Operational



# Continue...

- Mean and standard deviation
  - This gives a reflection of the average behavior and its variability
- Multivariate
  - Model is based on correlations between two or more variables.
  - For example, processor time and resource usage
- Markov process: used to establish transition probabilities among various states.
- Model is used to establish transition probabilities among various states.





# Continue...

- Time series
  - Model focuses on time intervals, looking for sequences of events that happen too rapidly or too slowly.
- Operational
  - Model is based on a judgment of what is considered abnormal.



# Rule-Based Intrusion Detection

- Rule-Based Intrusion Detection
  - Rule-based 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.



# Continue...

- Rule-based penetration identification
- 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
- Example of rules the following
  - Users should not read files in other users' personal directories
  - Users must not write other users' files
  - Users should not be logged in more than once to the same system



**Table 9.2 Measures That May Be Used for Intrusion Detection**

Measure	Model	Type of Intrusion Detected
<b>Login and Session Activity</b>		
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.

---

### 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.



# Continue...

- A simple example of the type of rules that can be used is found in NIDX, an early system that used heuristic rules that can be used to assign degrees of suspicion to activities [BAUE88]. Example heuristics are the following:
  1. Users should not read files in other users' personal directories.
  2. Users must not write other users' files.
  3. Users who log in after hours often access the same files they used earlier.
  4. Users do not generally open disk devices directly but rely on higher-level operating system utilities.
  5. Users should not be logged in more than once to the same system.
  6. Users do not make copies of system programs.



# The Base-Rate Fallacy

- 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.



# 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 standalone intrusion detection systems on each host, a more effective defense can be achieved by coordination and cooperation among intrusion detection systems across the network.





# Distributed Intrusion Detection

- Porras points out the following major issues in the design of a distributed intrusion detection system [PORR92]:
- 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.
- We need to assure the integrity and confidentiality of either raw audit data or summary data must be transmitted across the network.
- 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.



# Distributed Intrusion Detection

- 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 easing the task of correlating incoming reports but a potential bottleneck being single point of failure.
- With a decentralized architecture, there are more than one analysis centers, but these must coordinate their activities and exchange information.



# Distributed Intrusion Detection

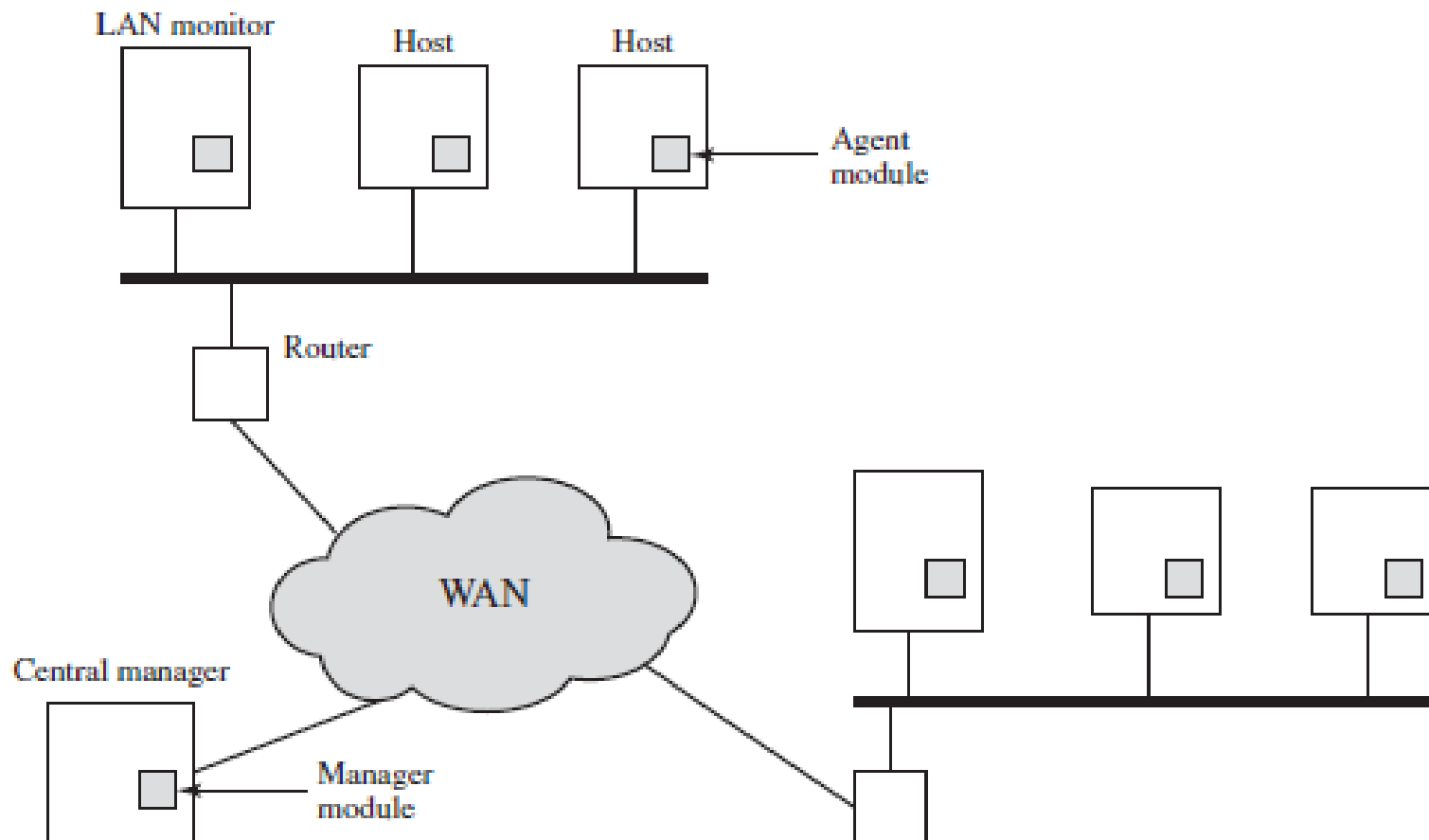


Figure 9.2 Architecture for Distributed Intrusion Detection



# Continue...

- 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
  - One or more nodes in the network will serve as collection and analysis points for the data from the systems on the network
  - Either a centralized or decentralized architecture can be used



# Continue...

## ○ **Components**

- Host agent module
  - Its purpose is to collect data on security related events on the host and transmit these to the central manager.
- LAN monitor agent module
  - 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.



# Continue...

- Figure 9.3 [SNAP91] shows the general approach.
- The agent captures each audit record produced by the native audit collection system and filter only those records that are of security interest.
- Records are reformatted into a standardized format called 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. E.g 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).



## Continue...

- At end the agent looks for anomalous behavior of an individual user based on a historical profile, 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.

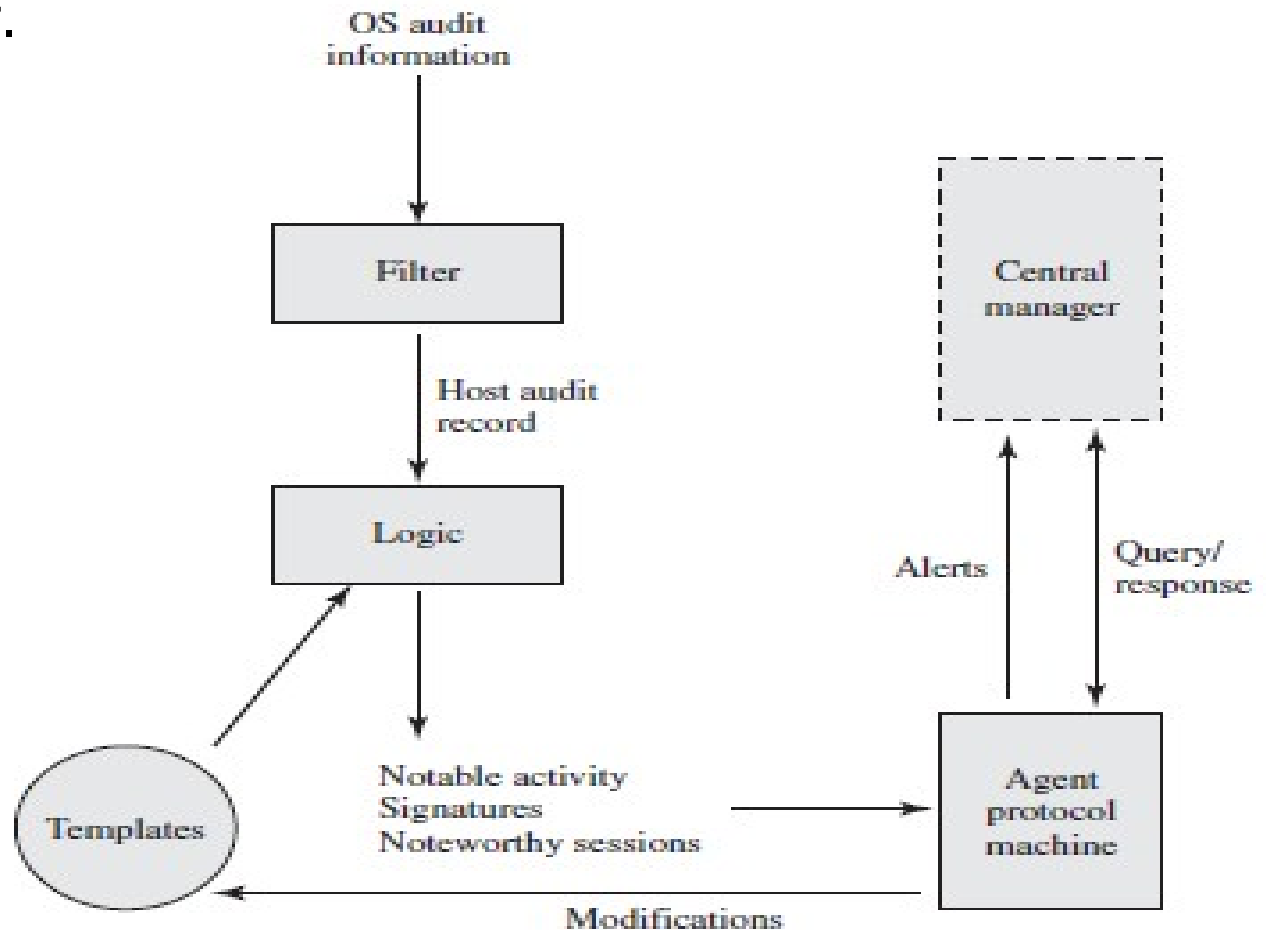


Figure 9.3 Agent Architecture

# Honeypots

- Honeypots are decoy systems that are designed to lure a potential attacker away from critical systems.
- 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.
- 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





# 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.
- IETF Intrusion Detection Working Group defines data formats and exchange procedures for sharing information of interest to intrusion detection and response systems
- The outputs of this working group include
  - A requirements document
  - A common intrusion language specification
  - A framework document



## 9.3 Password Management


### ○ Password Protection

- The front line of defense against intruders is the password system, which authenticates the ID of the individual logging on to the system.
- In turn, the ID provides security in the following ways
  - The ID determines whether the user is authorized to gain access to a system
  - The ID determines the privileges accorded to the user
  - The ID is used in what is referred to as flexible access control



# Continue...

## ○ **The Vulnerability of Password**

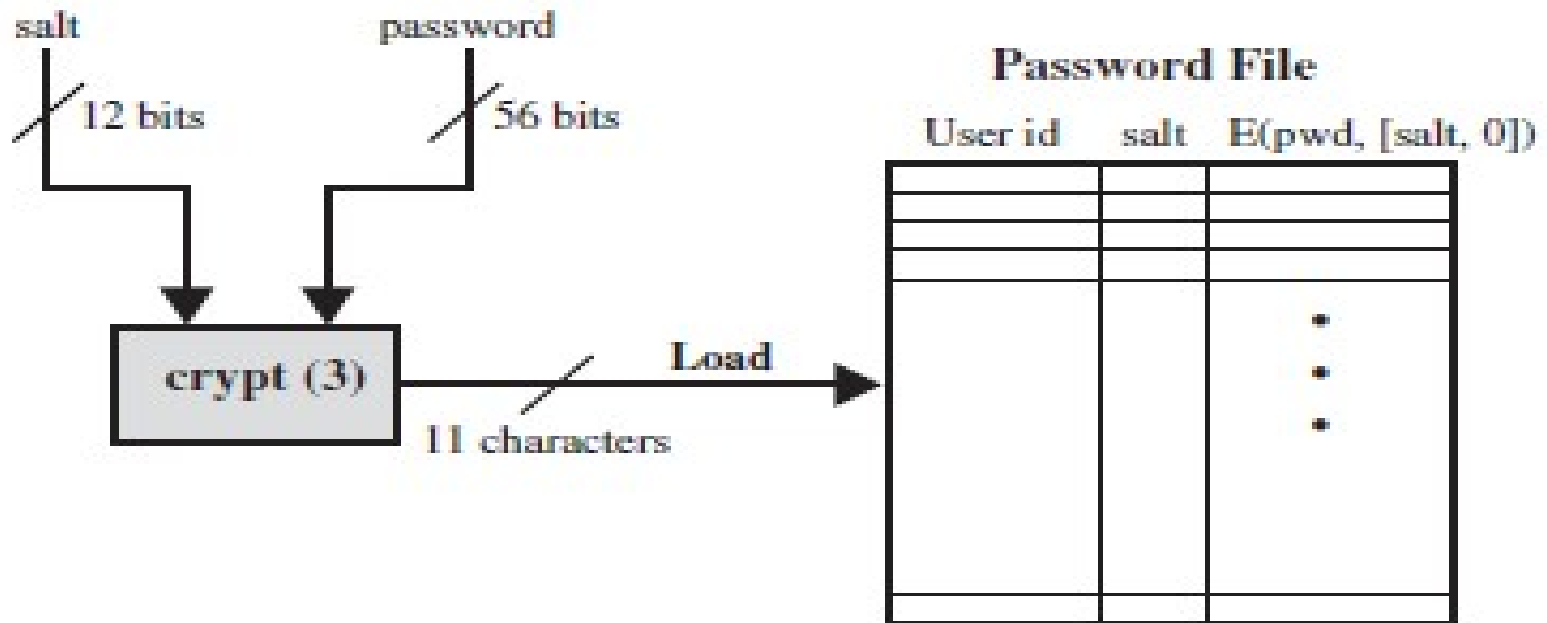
- Understand the nature of the threat to password-based systems.
  - E.g UNIX never stores passwords in the clear format
  - A typical user password is of up to 8 characters, which is converted into a 56-bit value (using 7-bit ASCII) to create i/p for encryption algorithm(here crypt(3) which is based on DES).
  - The DES algorithm is modified using a 12-bit value called salt, typically related to the time at which the password is assigned to the user.
  - The modified DES algorithm is exercised with a 64-bit block of zeros. The output serves as input for a second encryption. This process is repeated for a total of 25 encryptions.
  - The resulting 64-bit output is then translated into an 11-character sequence.
  - The hashed password is then stored, together with a plaintext copy of the salt, in the password file for the corresponding user ID.
- 

# Continue...

- The salt serves three purposes
  - It prevents duplicate passwords
  - It effectively increases the length of the password
  - It prevents brute-force guessing attack



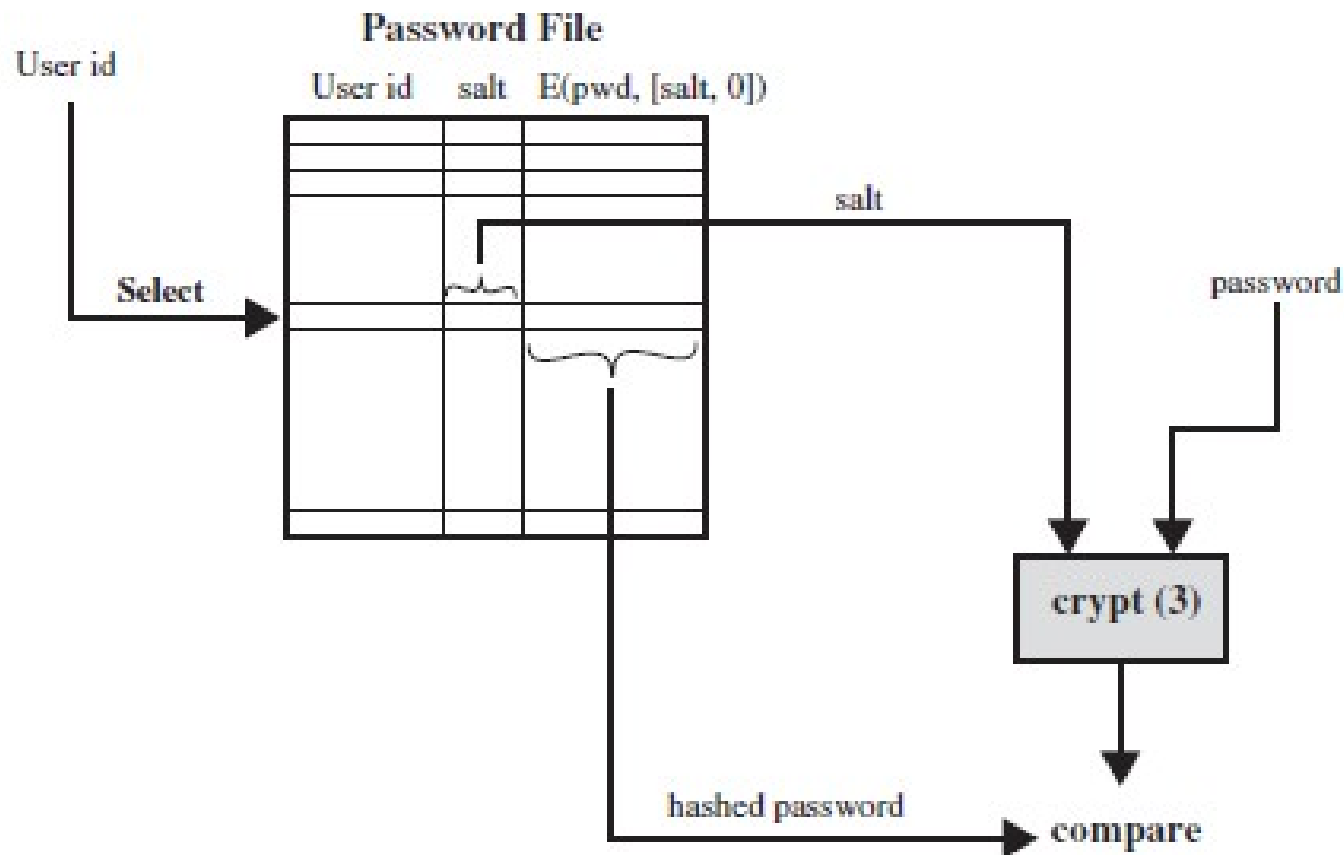
# Continue...



(a) Loading a new password



# Continue...



(b) Verifying a password

Figure 9.4 UNIX Password Scheme



# Password Selection Strategies

- Many users choose too short or too easy password to guess. So if users are assigned passwords consisting of eight randomly selected printable characters, password cracking is effectively impossible.
- But it would be almost as impossible for most users to remember their passwords.
- Our goal is to eliminate guessable passwords while allowing the user to select a memorable password.
- Four basic techniques are in use
  - User education
  - Computer-generated passwords
  - Reactive password checking
  - Proactive password checking



# Password Selection Strategies

- **User education**

- Make users understand importance of hard-to-guess password and provide guidelines for selecting strong passwords.
- Many users will simply ignore the guidelines. Others may not be good judges of what is a strong password.
- For example, many users (mistakenly) believe that reversing a word or capitalizing the last letter makes a password not guessable.





# Password Selection Strategies

- **Computer-generated passwords**

- If the passwords are quite random in nature, users will not be able to remember them.
- Even if the password is pronounceable, the user may have difficulty remembering it and so be tempted to write it down.
- *FIPS PUB 181* defines one of the best-designed automated password generators along with a description of the approach and a complete listing of the C source code of the algorithm.
- The algorithm generates words by forming pronounceable syllables and concatenating them to form a word.



# Password Selection Strategies

- **Reactive password checking**

- The system periodically runs its own password cracker to find guessable passwords, cancels any passwords that are guessed and notifies the user.
- The disadvantage is any existing passwords remain vulnerable until the reactive password checker finds them.



# Password Selection Strategies

- **Proactive password checker**

- A user is allowed to select his or her own password with the system checks to see if the password is allowable and, if not, rejects it.
- With sufficient guidance from the system, users can select memorable passwords from a fairly large password space that are not likely to be guessed in a dictionary attack.
- The trick with a proactive password checker is to strike a balance between user acceptability and strength.
- If the system rejects too many passwords, users will complain that it is too hard to select a password.
- If the system uses some simple algorithm to define what is acceptable, this provides guidance to password crackers to refine their guessing technique.



# Password Selection Strategies

- **Proactive password checker**

- Some proactive approaches are as follows:

- The **first approach** is a simple system for rule enforcement. For example, the following rules could be enforced:

- 1) All passwords must be at least eight characters long.

- 2) In the first eight characters, the passwords must include at least one each of uppercase, lowercase, numeric digits, and punctuation marks.

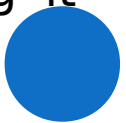
- Another possible procedure is simply to compile a large dictionary of possible “bad” passwords.

- When a user selects a password, the system checks to make sure that it is not on the disapproved list.

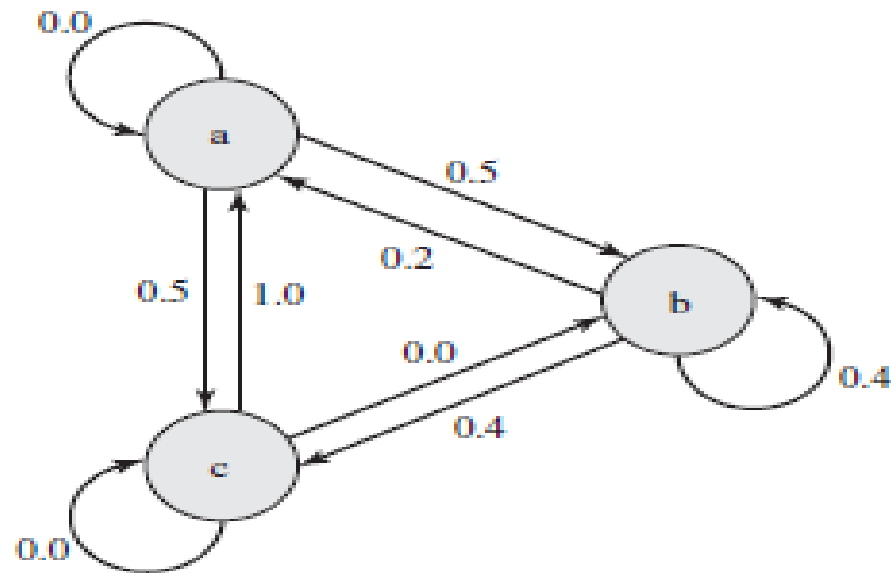
- There are two problems with this approach:

- Space: The dictionary must be very large to be effective.

- Time: The time required to search a large dictionary may itself be large including likely permutations of dictionary words making it take more time in processing.



# Markov model



$M = \{3, \{a, b, c\}, T, 1\}$  where

$$T = \begin{bmatrix} 0.0 & 0.5 & 0.5 \\ 0.2 & 0.4 & 0.4 \\ 1.0 & 0.0 & 0.0 \end{bmatrix}$$

e.g., string probably from this language: abbcacaba

e.g., string probably not from this language: aacccbbaaa

Figure 9.5 An Example Markov Model



# Continue...

- In general, a Markov model is a quadruple  $[m, A, T, k]$  , where  $m$  is the number of states in the model,  $A$  is the state space,  $T$  is the matrix of transition probabilities, and  $k$  is the order of the model.
- For a *2nd-order model following steps*



## Continue...

1. Determine the frequency matrix  $f$ , where  $f(i, j, k)$  is the number of occurrences of the trigram consisting of the  $i$ th,  $j$ th, and  $k$ th character. For example, the password *parsnips* yields the trigrams par, ars, rsni, sni, nip, and ips.
2. For each bigram  $ij$ , calculate  $f(i, j, \infty)$  as the total number of trigrams beginning with  $ij$ . For example,  $f(a, b, \infty)$  would be the total number of trigrams of the form aba, abb, abc, and so on.
3. Compute the entries of  $T$  as follows:

$$T(i, j, k) = \frac{f(i, j, k)}{f(i, j, \infty)}$$



# Bloom Filter

- A Bloom filter of order  $k$  consists of a set of  $k$  independent hash functions  $h_1(x), h_2(x) \dots h_k(x)$ , where each function maps a password into a hash value in the range 0 to  $N-1$ .
- The following procedure is then applied to the dictionary
  - A hash table of  $N$  bits is defined, with all bits initially set to 0.
  - For each password, its  $k$  hash values are calculated, and the corresponding bits in the hash table are set to 1
- **False positives**
- That is, passwords that are not in the dictionary but that produce a match in the hash table.





Continue...

