OPERATING SYSTEMS

Week 11

Much of the material on these slides comes from the recommended textbook by William Stallings

Detailed content

Weekly program

- ✓ Week 1 – Operating System Overview
- ✓ Week 2 – Processes and Threads
- ✓ Week 3 – Scheduling
- ✓ Week 4 – Real-time System Scheduling and Multiprocessor Scheduling
- ✓ Week 5 – Concurrency: Mutual Exclusion and Synchronization
- ✓ Week 6 – Concurrency: Deadlock and Starvation
- ✓ Week 7 – Memory Management
- ✓ Week 8 – Memory Management II
- ✓ Week 9 – Disk and I/O Scheduling
- ✓ Week 10 – File Management
- ❑ **Week 11 – Security and Protection**
- ❑ Week 12 – Revision of the course
- ❑ Week 13 – Extra revision (if needed)

Key Concepts From Last Lecture

- File organization: Pile, Sequential, Indexed Sequential, Indexed, Direct or Hashed file
- Indexed structure: B-Trees
- Directories: Two level, Tree Structured, Acyclic-Graph directory
- File sharing: Access rights
- Record blocking: Fixed blocking, Variable-length spanned blocking, variable-length unspanned blocking
- Secondary storage management: Two tasks
- File Allocation methods: contiguous allocation, chained allocation, indexed allocation
- Free space management: bit table, chained free portion, indexing, free block list

Week 11 Lecture Outline

Security and Protection

- ❑ Computer Security: CIA triad
- ❑ Threat consequences and Threat Areas
- ❑ Intruders and malicious software
- ❑ Buffer overflow
- ❑ Defences against buffer overflow
- ❑ Countermeasures
- ❑ Authentication
- ❑ Access control
- ❑ Intrusion Detection
- ❑ Malware Defence

Videos to watch before lecture

Computer Security

- **Computer security: The protection afforded to an automated information system**
  - in order to attain the applicable objectives of preserving the integrity, availability, and confidentiality of information system resources
  - includes hardware, software, firmware, information/data, and telecommunications.

Key Objectives of Computer Security

- **Confidentiality**
  - ***Data confidentiality*** assures that private or confidential information is not made available or disclosed to unauthorized individuals
  - ***Privacy*** assures that individuals control or influence what information related to them may be collected and stored and by whom and to whom that information may be disclosed
- **Integrity**
  - ***Data integrity*** assures that information and programs are changed only in a specified and authorized manner
  - ***System integrity*** assures that a system performs its intended function in an unimpaired manner, free from deliberate or inadvertent unauthorized manipulation of the system
- **Availability**
  - assures that systems work promptly and service is not denied to authorized users

CIA Triad

- Security Objectives:
  - Confidentiality
    - a loss of confidentiality is the unauthorized disclosure of information
  - Integrity
    - a loss of integrity is the unauthorized modification or destruction of information
  - Availability
    - a loss of availability is the disruption of access to or use of information or an information system

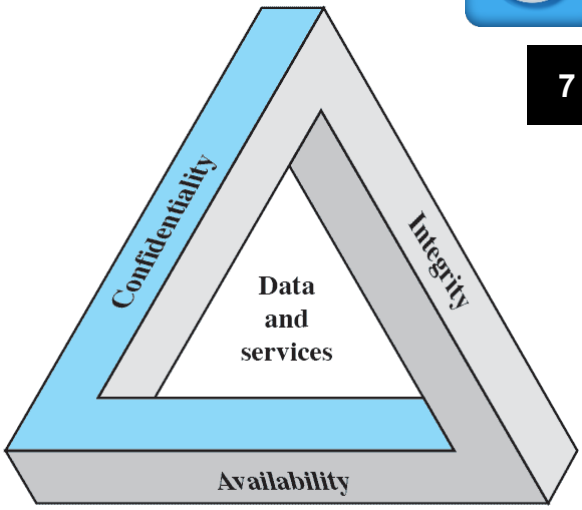


Figure 14.1 The Security Requirements Triad

Additional Concepts

- Two further concepts are often added to the core of computer security:

Authenticity

- The property of being genuine and being able to be verified and trusted; confidence in the validity of a transmission, a message, or message originator
- Verifying that users are who they say they are and that each input arriving at the system came from a trusted source

Accountability

- The security goal that generates the requirement for actions of an entity to be traced uniquely to that entity
- We must be able to trace a security breach to a responsible party
- Systems must keep records of their activities to permit later forensic analysis to trace security breaches or to aid in transaction disputes

Threat consequences

Four basic types of threat consequences are:

[based on RFC2828/RFC4949]

1. Unauthorized Disclosure
2. Deception
3. Disruption
4. Usurpation

Unauthorized Disclosure

Threat Consequences, and the Types of Threat Actions That Cause Each Consequence

A threat to confidentiality

(Based on RFC 2828/RFC 4949)

Threat Consequence	Threat Action (attack)
Unauthorized Disclosure A circumstance or event whereby an entity gains access to data for which the entity is not authorized.	**Exposure:** Sensitive data are directly released to an unauthorized entity. **Interception:** An unauthorized entity directly accesses sensitive data traveling between authorized sources and destinations. **Inference:** A threat action whereby an unauthorized entity indirectly accesses sensitive data (but not necessarily the data contained in the communication) by reasoning from characteristics or byproducts of communications. **Intrusion:** An unauthorized entity gains access to sensitive data by circumventing a system's security protections.

Deception

Threat Consequences, and the Types of Threat Actions That Cause Each Consequence

A threat to either system integrity or data integrity

(Based on RFC 2828 / RFC 4949)

Threat Consequence	Threat Action (attack)
Deception A circumstance or event that may result in an authorized entity receiving false data and believing it to be true.	**Masquerade:** An unauthorized entity gains access to a system or performs a malicious act by posing as an authorized entity. **Falsification:** False data deceive an authorized entity. **Repudiation:** An entity deceives another by falsely denying responsibility for an act.

Disruption

Threat Consequences, and the Types of Threat Actions That Cause Each Consequence

A threat to availability or system integrity

(Based on RFC 2828 / RFC 4949)

Threat Consequence	Threat Action (attack)
Disruption A circumstance or event that interrupts or prevents the correct operation of system services and functions.	**Incapacitation:** Prevents or interrupts system operation by disabling a system component. **Corruption:** Undesirably alters system operation by adversely modifying system functions or data. **Obstruction:** A threat action that interrupts delivery of system services by hindering system operation.

Usurpation

Threat Consequences, and the Types of Threat Actions That Cause Each Consequence

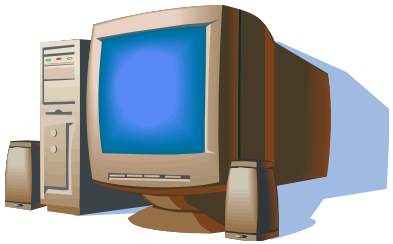
A threat to system integrity

(Based on RFC 2828 / RFC 4949)

Threat Consequence	Threat Action (attack)
Usurpation A circumstance or event that results in control of system services or functions by an unauthorized entity.	**Misappropriation:** An entity assumes unauthorized logical or physical control of a system resource. **Misuse:** Causes a system component to perform a function or service that is detrimental to system security.

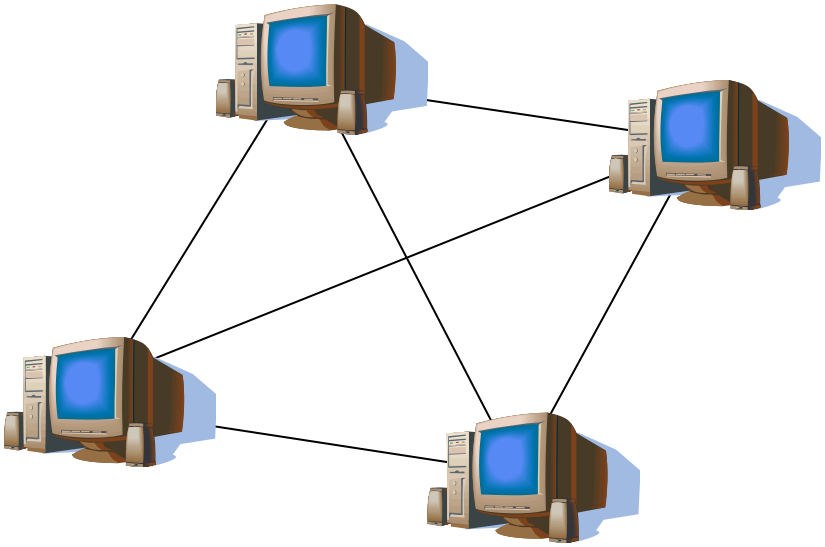
Three circles of concern

- Memory and files of a computer
  - RAM and disk storage
  - Need to prevent processes from encroaching on each other's memory areas



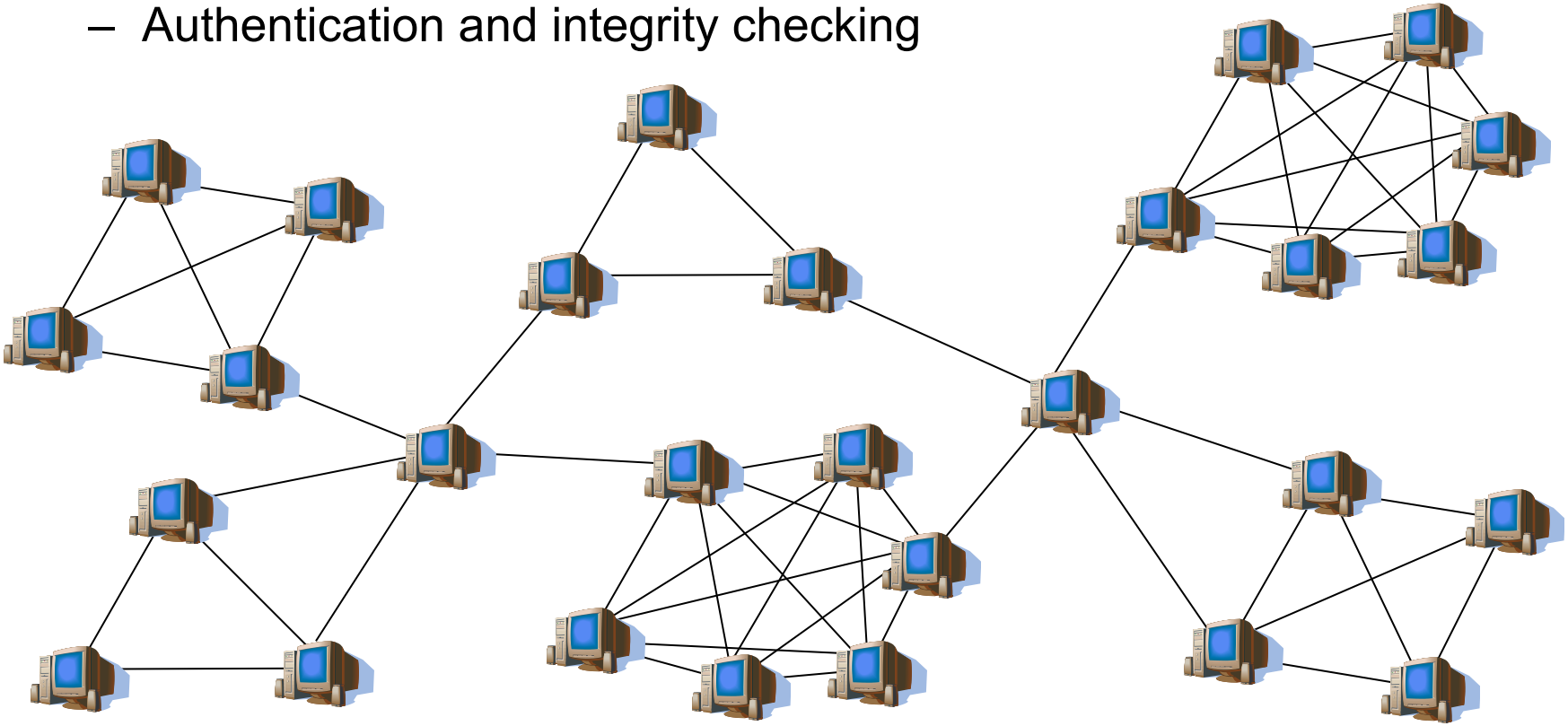
Three circles of concern

- Mutual trusting network – Intranet
  - Concern of user authentication

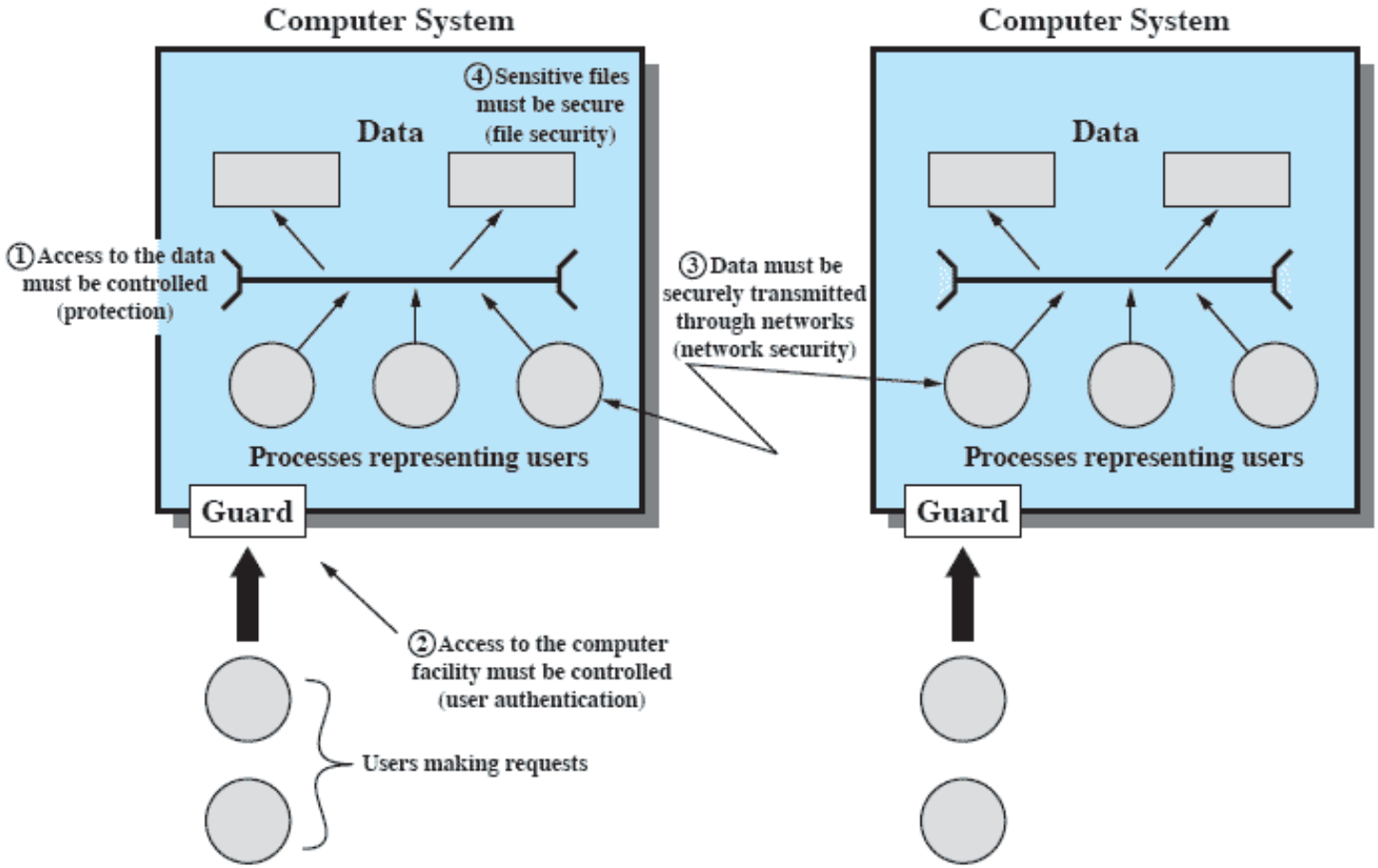


Three circles of concern

- The Internet
  - Need to complete exchange transactions successfully
  - Authentication and integrity checking



Scope of System Security



Examples of Threats

	Availability	Confidentiality	Integrity
Hardware	Equipment is stolen or disabled, thus denying service.		
Software	Programs are deleted, denying access to users.	An unauthorized copy of software is made.	A working program is modified, either to cause it to fail during execution or to cause it to do some unintended task.
Data	Files are deleted, denying access to users.	An unauthorized read of data is performed. An analysis of statistical data reveals underlying data.	Existing files are modified or new files are fabricated.
Communication Lines	Messages are destroyed or deleted. Communication lines or networks are rendered unavailable.	Messages are read. The traffic pattern of messages is observed.	Messages are modified, delayed, reordered, or duplicated. False messages are fabricated.

Threat areas: Hardware

- A major threat to computer system hardware is the threat to availability.
- Hardware is the most vulnerable to attack and the least susceptible to automated controls.
- Threats include accidental and deliberate damage to equipment as well as theft.
- The proliferation of personal computers and workstations and the widespread use of LANs increase the potential for losses in this area.
- Theft of CD-ROMs and DVDs can lead to loss of confidentiality. Physical and administrative security measures are needed to deal with these threats.

Threat areas: Software

- Software includes the operating system, utilities, and application programs. A key threat to software is an attack on availability.
- Software, especially application software, is often easy to delete. Software can also be altered or damaged to render it useless.
- Careful software configuration management, which includes making backups of the most recent version of software, can maintain high availability.
- A more difficult problem to deal with is software modification that results in a program that still functions but that behaves differently than before, which is a threat to integrity/authenticity.
  - Computer viruses and related attacks fall into this category.
- A final problem is protection against software piracy. Although certain countermeasures are available, by and large the problem of unauthorized copying of software has not been solved.

Threat areas: Data

- Hardware and software security are typically concerns of computing centre professionals or individual concerns of personal computer users.
- A much more widespread problem is data security, which involves files and other forms of data controlled by individuals, groups, and business organizations.

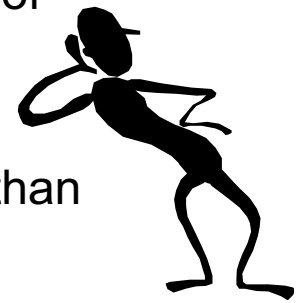
- Security concerns with respect to data are broad, encompassing availability, secrecy, and integrity.
  - **Availability**: the concern is with the destruction of data files, which can occur either accidentally or maliciously.

  - **Confidentiality:** the unauthorized reading of data files or databases and the analysis of data and the use of so-called statistical databases, which provide summary or aggregate information. Presumably, the existence of aggregate information does not threaten the privacy of the individuals involved. However, as the use of statistical databases grows, there is an increasing potential for disclosure of personal information. In essence, characteristics of constituent individuals may be identified through careful analysis.

  - **Integrity**: A major concern in most installations. Modifications to data files can have consequences ranging from minor to disastrous

Passive Attacks

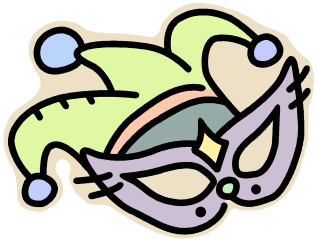
- Attempts to learn or make use of information from the system but does not affect system resources
- Are in the nature of eavesdropping on, or monitoring of, transmissions
- Goal of the attacker is to obtain information that is being transmitted
- Difficult to detect because they do not involve any alteration of the data
  - is feasible to prevent the success of these attacks by means of encryption
- Emphasis in dealing with passive attacks is on prevention rather than detection



Types:

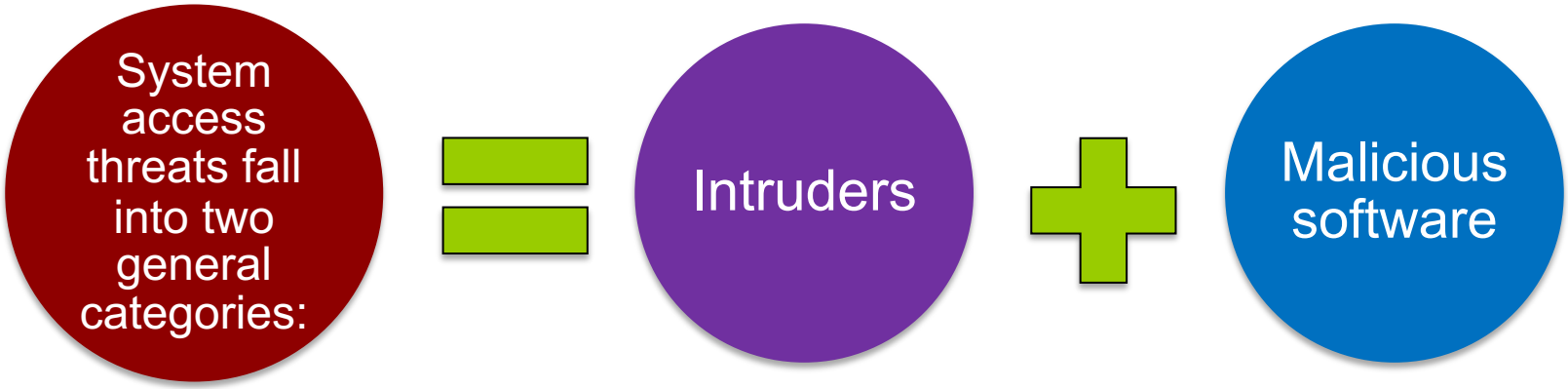
- release of message contents
- traffic analysis

Active Attacks



- Involve some modification of the data stream or the creation of a false stream. Four categories:
- Replay
  - involves the passive capture of a data unit and its subsequent retransmission to produce an unauthorized effect
- Masquerade
  - takes place when one entity pretends to be a different entity
- Modification of messages
  - some portion of a legitimate message is altered, or that messages are delayed or reordered, to produce an unauthorized effect
- Denial of service
  - prevents or inhibits the normal use or management of communications facilities disruption of an entire network either by disabling the network or by overloading it with messages so as to degrade performance

System Access Threats



Intruders

Masquerader

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

Misfeasor

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

Clandestine user

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

Intruders

- Intruders attacks range from the benign to the serious
- The objective of the intruder is:
  - to gain access to a system OR
  - to increase the range of privileges accessible on a system
- Most attacks use system or software vulnerabilities that allow user to execute code that opens a backdoor into the system
- Alternatively intruders attempts to acquire information that should have been protected.
  - E.g. user password

Intruder Patterns of Behavior: Hackers

- Traditionally those who hack do so for the thrill of it or for status
- Attackers often look for targets of opportunity and then share the information with others
- Benign intruders consume resources and may slow performance for legitimate users
- Intrusion detection systems (IDSs) and intrusion prevention systems (IPSs) are designed to counter this type of hacker threat
- Computer emergency response teams (CERTs) are cooperative ventures who collect information about system vulnerabilities and disseminate it to systems managers

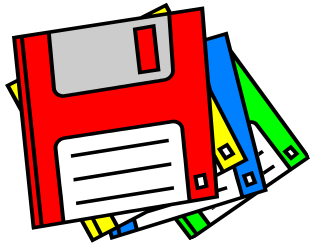
Intruder Patterns of Behavior: Criminal Enterprise

- Organized groups of hackers
- They meet in underground forums to trade tips and data and coordinate attacks
- Usually have specific targets, or at least classes of targets in mind
  - A common target is a credit card file at an e-commerce server
- Quick in and quick out attacks

Intruder Patterns of Behavior: Insider Attacks

- Among the most difficult to detect and prevent
    - Can be motivated by revenge or simply a feeling of entitlement
- Employees already have access to and knowledge of the structure and content of corporate databases

Malware



General term for any malicious software

Software designed to cause damage to or use up the resources of a target computer

Frequently concealed within or masquerades as legitimate software

In some cases it spreads itself to other computers via e-mail or infected discs

THE UNIVERSITY OF NEWCASTLE AUSTRALIA

Terminology of Malicious Programs

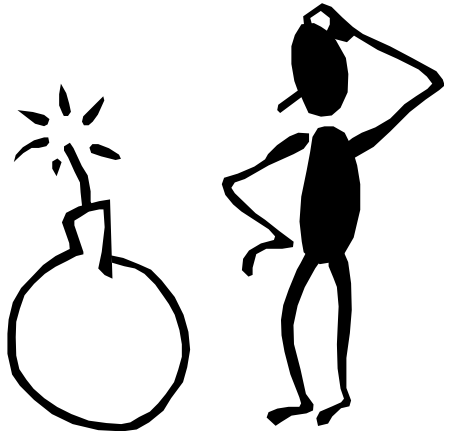
Name	Description
Virus	Malware that, when executed, tries to replicate itself into other executable code; when it succeeds the code is said to be infected. When the infected code is executed, the virus also executes.
Worm	A computer program that can run independently and can propagate a complete working version of itself onto other hosts on a network.
Logic bomb	A program inserted into software by an intruder. A logic bomb lies dormant until a predefined condition is met; the program then triggers an unauthorized act.
Trojan horse	A computer program that appears to have a useful function, but also has a hidden and potentially malicious function that evades security mechanisms, sometimes by exploiting legitimate authorizations of a system entity that invokes the Trojan horse program.
Backdoor (trapdoor)	Any mechanisms that bypasses a normal security check; it may allow unauthorized access to functionality.
Mobile code	Software (e.g., script, macro, or other portable instruction) that can be shipped unchanged to a heterogeneous collection of platforms and execute with identical semantics.
Exploits	Code specific to a single vulnerability or set of vulnerabilities.
Downloaders	Program that installs other items on a machine that is under attack. Usually, a downloader is sent in an e-mail.
Auto-rooter	Malicious hacker tools used to break into new machines remotely.
Kit (virus generator)	Set of tools for generating new viruses automatically.
Spammer programs	Used to send large volumes of unwanted e-mail.
Flooders	Used to attack networked computer systems with a large volume of traffic to carry out a denial-of-service (DoS) attack.
Keyloggers	Captures keystrokes on a compromised system.
Rootkit	Set of hacker tools used after attacker has broken into a computer system and gained root-level access.
Zombie, bot	Program activated on an infected machine that is activated to launch attacks on other machines.
Spyware	Software that collects information from a computer and transmits it to another system.
Adware	Advertising that is integrated into software. It can result in pop-up ads or redirection of a browser to a commercial site.

Backdoor

- Also known as a trapdoor
- A secret entry point into a program that allows someone to gain access without going through the usual security access procedures
- A ***maintenance hook*** is a backdoor that programmers use to debug and test programs
- Become threats when unscrupulous programmers use them to gain unauthorized access
- It is difficult to implement operating system controls for backdoors

Logic Bomb

- One of the oldest types of program threat
- Code embedded in some legitimate program that is set to “explode” when certain conditions are met
- Once triggered a bomb may alter or delete data or entire files, cause a machine halt, or do some other damage



Trojan Horse

- Useful, or apparently useful, program or command procedure that contains hidden code that, when invoked, performs some unwanted or harmful function

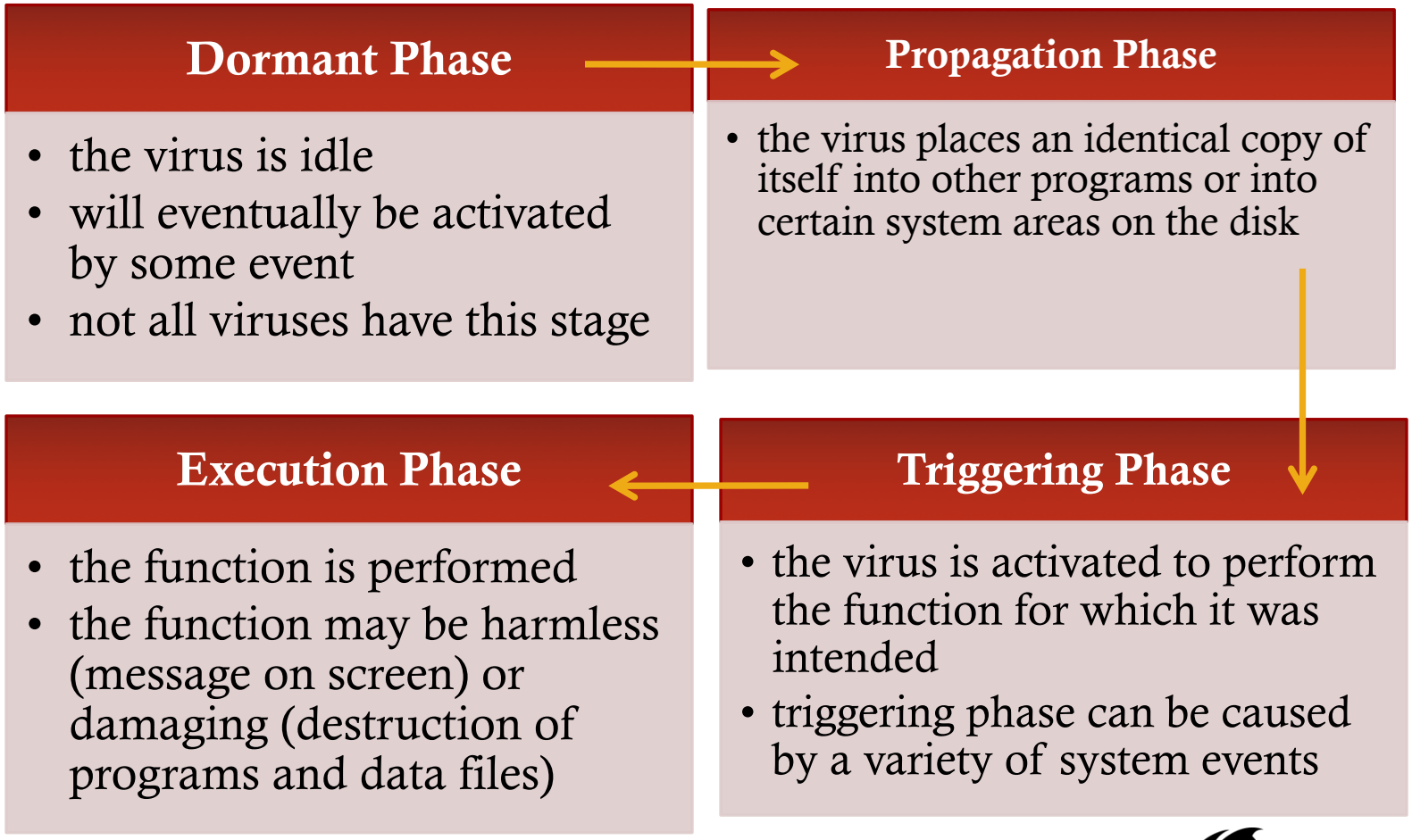
- Trojan horses fit into one of three models:
    1) continuing to perform the function of the original program and additionally performing a separate malicious activity
    2) continuing to perform the function of the original program but modifying the function to perform malicious activity or to disguise other malicious activity
    3) performing a malicious function that completely replaces the function of the original program

Viruses

- Software that “infects” other programs by modifying them
  - carries instructional code to **self duplicate**
  - becomes **embedded** in a program on a computer
  - when the **infected** computer comes into contact with an uninfected piece of software, a fresh copy of the virus passes into the new program
  - infection can be spread by swapping disks from computer to computer or through a network

- A computer virus has **three parts**:
  - an infection mechanism
  - trigger
  - payload

Virus Phases



Simple Virus

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

```
   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:

}
```

Compression Virus

- In this example the virus does nothing other than propagate
- One way to thwart detection is to compress the executable file so that both the infected and uninfected versions are of identical length

```
    program CV :=

{goto main;
    01234567;

    subroutine infect-executable :=
          {loop:
               file := get-random-executable-file;
          if (first-line-of-file = 01234567) then goto loop;
       (1)    compress file;
       (2)    prepend CV to file;
       }

main:  main-program :=
          {if ask-permission then infect-executable;
       (3)    uncompress rest-of-file;
       (4)    run uncompressed file;}
       }
```

Virus Classification

- There is no universally agreed upon classification scheme for viruses
- 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

Virus Concealment Strategy

- A virus **classification by concealment strategy** includes:

Encrypted virus

random encryption key encrypts remainder of virus

Stealth virus

hides itself from detection of antivirus software

Polymorphic virus

mutates with every infection

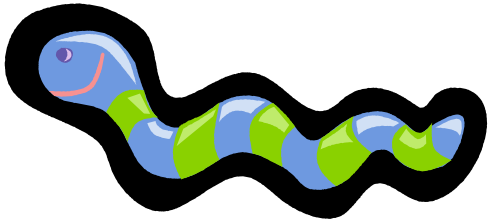
mutation engine is the portion of the virus that is responsible for generating keys and performing encryption/decryption

Metamorphic virus

mutates with every infection

rewrites itself completely after every iteration

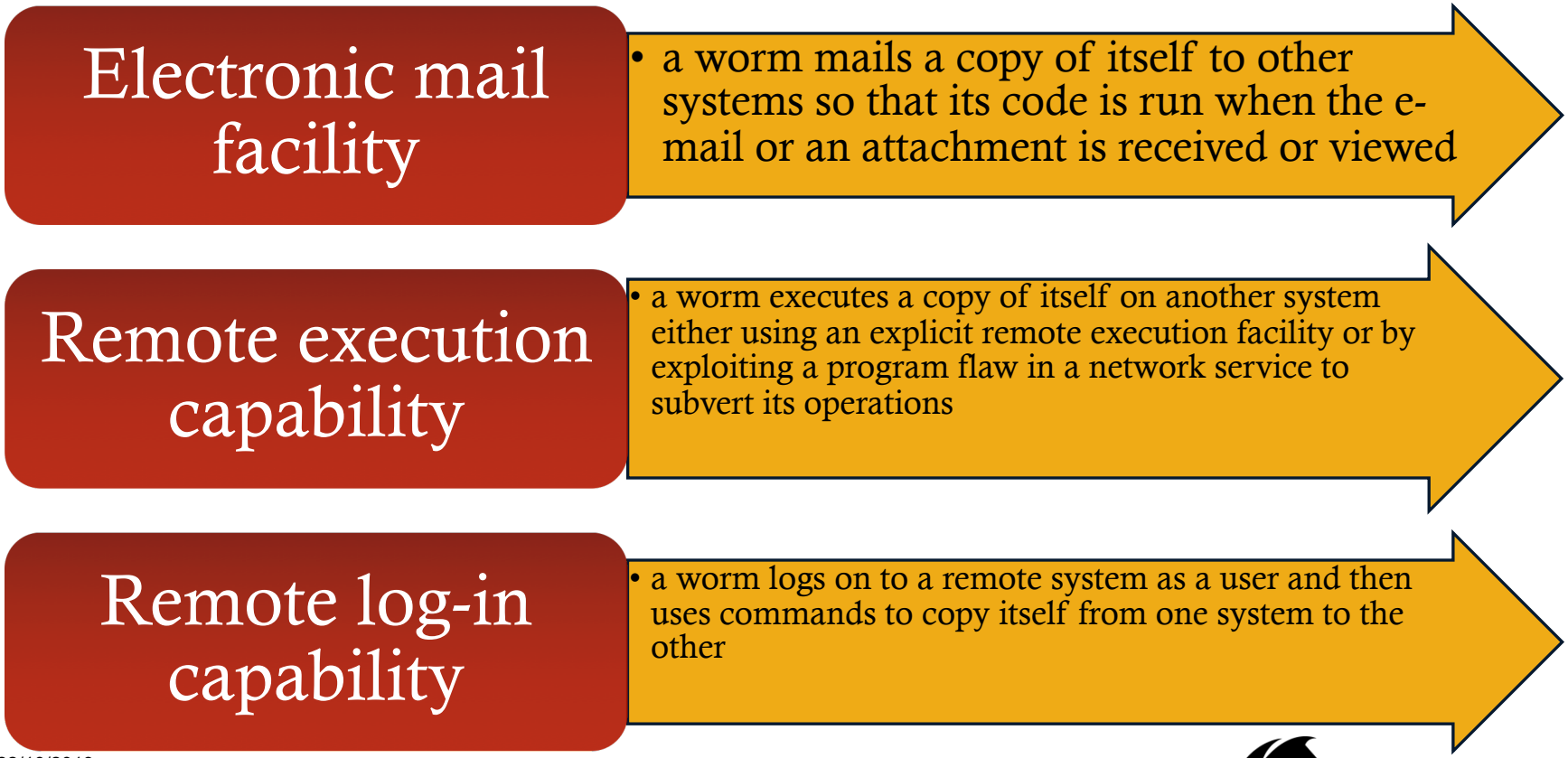
Worms



- A program that can replicate itself and send copies from computer to computer across network connections
- Upon arrival the worm may be activated to replicate and propagate again
- In addition to propagation the worm usually performs some unwanted function
- Actively seeks out more machines to infect and each machine that is infected serves as an automate launching pad for attacks on other machines

Worm Propagation

- To replicate itself a network worm uses some sort of network vehicle



Uses of Bots

Distributed denial-of-service (DDoS) attacks

- causes a loss of service to users

Spamming

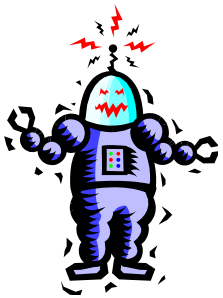
- sending massive amounts of bulk e-mail (spam)

Sniffing traffic

- a packet sniffer is used to retrieve sensitive information like user names and passwords

Keylogging

- captures keystrokes



Spreading new malware

- botnets are used to spread new bots

Installing advertisement add-ons and browser helper objects (BHOs)

- set up a fake Web site and negotiate a deal with hosting companies that pay for clicks on ads

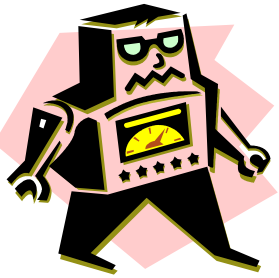
Attacking Internet Relay chat (IRC) chat networks

- victim is flooded with requests, bringing down the IRC network; similar to a DDoS attack

Manipulating online polls/games

- every bot has a distinct IP address so it appears to be a real person

Remote Control Facility



- Distinguishes a bot from a worm
    - a worm propagates and activates itself, whereas a bot is controlled from some central facility (at least initially)

- A typical means of implementing the remote control facility is on an IRC server
    - all bots join a specific channel on this server and treat incoming messages as commands

- More recent botnets tend to use covert communication channels via protocols such as HTTP

- Distributed control mechanisms are also used to avoid a single point of failure

Buffer Overflow Attacks

- Also known as a ***buffer overrun***
- Defined in the NIST (National Institute of Standards and Technology) *Glossary of Key Information Security Terms* as:

> "A condition at an interface under which more input can be placed into a buffer or data-holding area than the capacity allocated, overwriting other information. Attackers exploit such a condition to crash a system or to insert specially crafted code that allows them to gain control of the system"

- One of the most prevalent and dangerous types of security attacks

```
int main(int argc, char *argv[]) {
    int valid = FALSE;
    char str1[8];
    char str2[8];

    next_tag(str1);
    gets(str2);
    if (strncmp(str1, str2, 8) == 0)
        valid = TRUE;
    printf("buffer1: str1(%s), str2(%s), valid(%d)\n", str1, str2, valid);
}
```

(a) Basic buffer overflow C code

```
$ cc -g -o buffer1 buffer1.c
$ ./buffer1
START
buffer1: str1(START), str2(START), valid(1)
$ ./buffer1
EVILINPUTVALUE
buffer1: str1(TVALUE), str2(EVILINPUTVALUE), valid(0)
$ ./buffer1
BADINPUTBADINPUT
buffer1: str1(BADINPUT), str2(BADINPUTBADINPUT), valid(1)
```

(b) Basic buffer overflow example runs

Figure 15.1 Basic Buffer Overflow Example

Memory Address	Before gets(str2)	After gets(str2)	Contains Value of
. . . .	. . . .	. . . .	
bffffbf4	34fcffbf 4 . . .	34fcffbf 3 . . .	argv
bffffbf0	01000000	01000000	argc
bffffbec	c6bd0340 . . . @	c6bd0340 . . . @	return addr
bffffbe8	08fcffbf	08fcffbf	old base ptr
bffffbe4	00000000	01000000	valid
bffffbe0	80640140 . d . @	00640140 . d . @	
bffffbdc	54001540 T . . @	4e505554 N P U T	str1[4-7]
bffffbd8	53544152 S T A R	42414449 B A D I	str1[0-3]
bffffbd4	00850408	4e505554 N P U T	str2[4-7]
bffffbd0	30561540 0 V . @	42414449 B A D I	str2[0-3]
. . . .	. . . .	. . . .	

Figure 15.2 Basic Buffer Overflow Stack Values

THE UNIVERSITY OF NEWCASTLE AUSTRALIA

Exploiting Buffer Overflow

- To exploit any type of buffer overflow the attacker needs:
  - To identify a buffer overflow vulnerability in some program that can be triggered using externally sourced data under the attackers control
  - To understand how that buffer will be stored in the processes memory, and hence the potential for corrupting adjacent memory locations and potentially altering the flow of execution of the program

Compile-Time Defenses

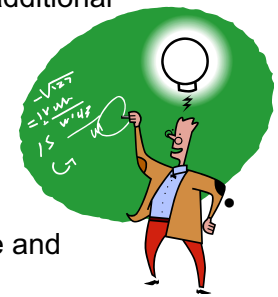
- Countermeasures can be broadly classified into two categories:

    1) Compile-time defenses, which aim to harden programs to resist attacks

    2) Runtime defenses, which aim to detect and abort attacks in executing programs

Compile-time Techniques

- **Choice of programming language**
  - one possibility is to write the program using a modern high-level programming language that has a strong notion of variable type and what constitutes permissible operations on them
  - the flexibility and safety provided by these languages does come at a cost in resource use, both at compile time and also in additional code that must execute at runtime



- **Safe coding techniques**
  - programmers need to inspect the code and rewrite any unsafe coding constructs
  - an example is the OpenBSD project which produces a free, multiplatform 4.4BSD-based UNIX-like operating system
  - among other technology changes, programmers have under-taken an extensive audit of the existing code base, including the operating system, standard libraries, and common utilities

- **Language extensions and use of safe libraries**
  - there have been a number of proposals to augment compilers to automatically insert range checks on pointer references
  - Libsafe is an example that implements the standard semantics but includes additional checks to ensure that the copy operations do not extend beyond the local variable space in the stack frame

- **Stack protection mechanisms**
  - an effective method for protecting programs against classic stack overflow attacks is to instrument the function entry and exit code to set up and then check its stack frame for any evidence of corruption
  - Stackguard, one of the best-known protection mechanisms, is a GNU Compile Collection (GCC) compiler extension that inserts additional function entry and exit code

Runtime Techniques

- **Executable address space protection**
  - a possible defense is to block the execution of code on the stack, on the assumption that executable code should only be found elsewhere in the processes address space
  - extensions have been made available to Linux, BSD, and other UNIX-style systems to support the addition of the no-execute bit
- **Guard pages**
  - gaps are placed between the ranges of addresses used for each of the components of the address space
  - these gaps, or guard pages, are flagged in the MMU as illegal addresses and any attempt to access them results in the process being aborted
  - a further extension places guard pages between stack frames or between different allocations on the heap
- **Address space randomization**
  - a runtime technique that can be used to thwart attacks involves manipulation of the location of key data structures in the address space of a process
  - moving the stack memory region around by a megabyte or so has minimal impact on most programs but makes predicting the targeted buffer's address almost impossible
  - another technique is to use a security extension that randomizes the order of loading standard libraries by a program and their virtual memory address locations

Authentication

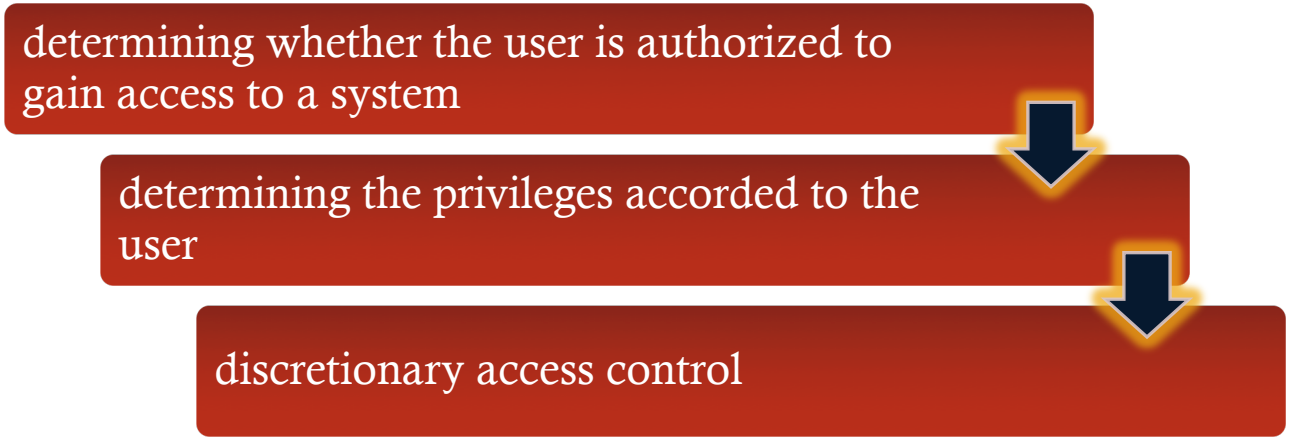
- In most computer security contexts, user authentication is the fundamental building block and the primary line of defense
- RFC 4949 defines user authentication as the process of verifying an identity claimed by or for a system entity
- An authentication process consists of two steps:
    - identification step
        - » presenting an identifier to the security system
    - verification step
        - » presenting or generating authentication information that corroborates the binding between the entity and the identifier

Means of Authentication

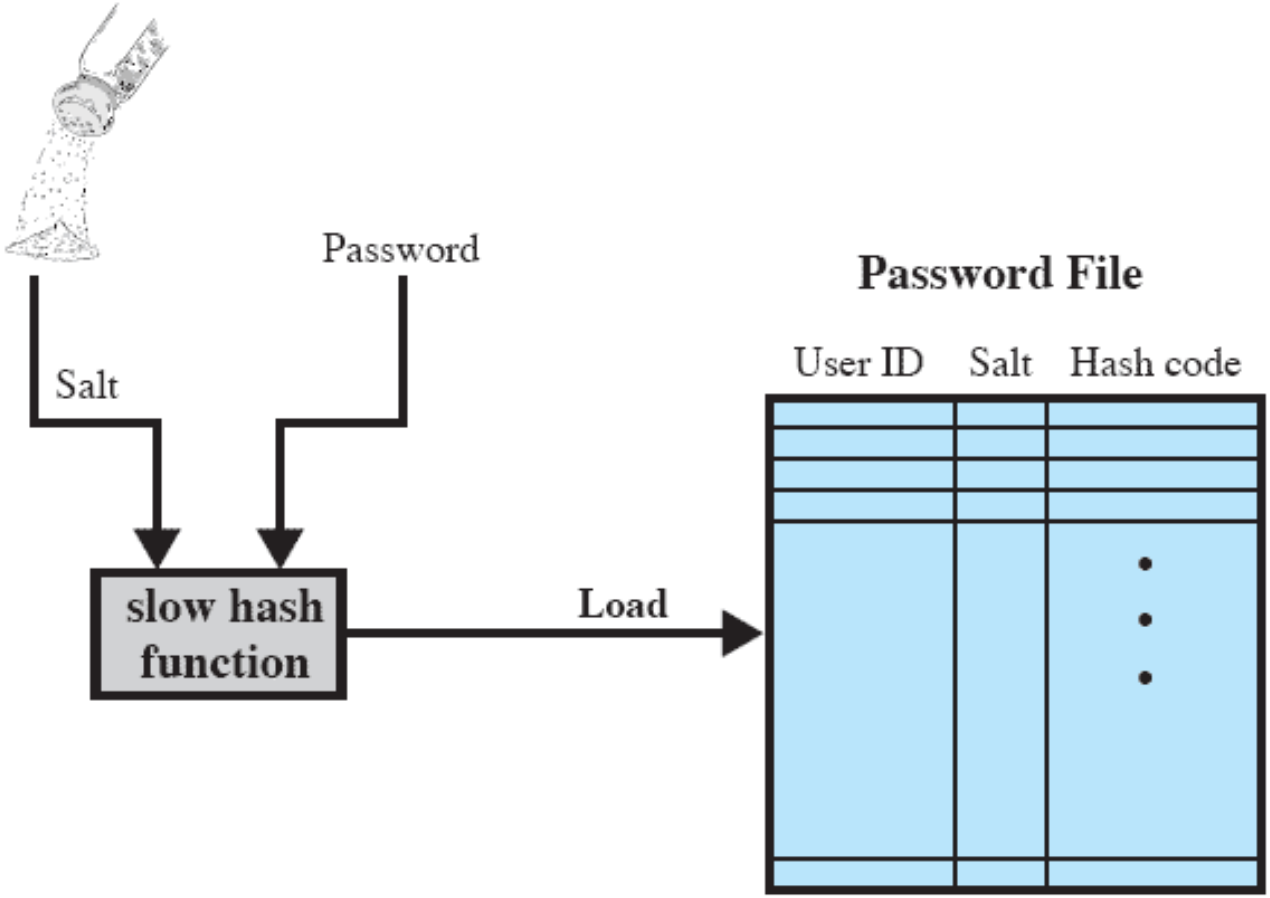
- **Something the individual knows**
  - examples include a password, a personal identification number (PIN), or answers to a prearranged set of questions
- **Something the individual possesses**
  - examples include electronic keycards, smart cards, and physical keys
  - referred to as a ***token***
- **Something the individual is (static biometrics)**
  - examples include recognition by fingerprint, retina, and face
- **Something the individual does (dynamic biometrics)**
  - examples include recognition by voice pattern, handwriting characteristics, and typing rhythm

Password-based authentication

- A widely used line of defense against intruders is the password system
- The password serves to authenticate the ID of the individual logging on to the system
- The ID provides security by:



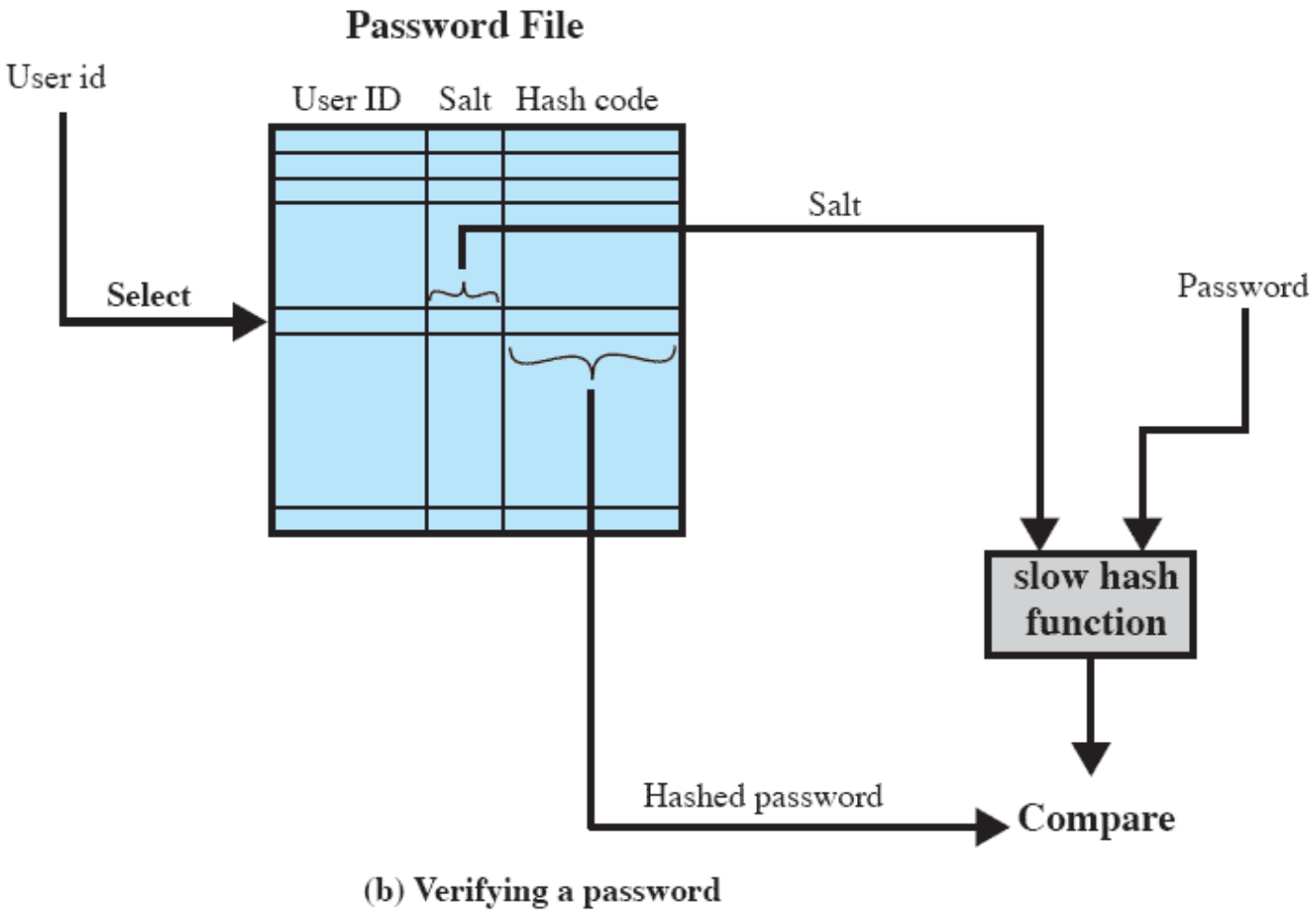
Hashed passwords/salt value



(a) Loading a new password

THE UNIVERSITY OF NEWCASTLE AUSTRALIA

UNIX password scheme



(b) Verifying a password

Salt

- Serves three purposes:

1. prevents duplicate passwords from being visible in the password file
   - even if two users choose the same password, the passwords will be assigned different salt values
2. greatly increases the difficulty of offline dictionary attacks
3. it becomes nearly impossible to find out whether a person with passwords on two or more systems has used the same password on all of them

UNIX Password Scheme

- There are two threats to the UNIX password scheme:
  - a user can gain access on a machine using a *guest account*
  - **Password cracker** – password guessing program

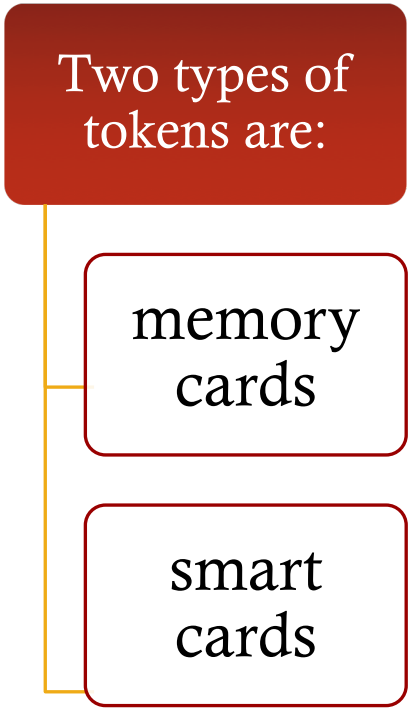
- If an opponent is able to obtain a copy of the password file, a cracker program can be run on another machine at leisure
  - this enables the opponent to run through millions of possible passwords in a reasonable period

UNIX Implementations

- Most implementations have relied on a password scheme where each user selects a password of up to eight printable characters in length which is converted into a 56-bit value that serves as the key input to an encryption routine
- The hash routine, known as crypt(3) is based on DES (Data Encryption Standard)
- The crypt(3) routine is designed to discourage guessing attacks
- Software implementations of DES are slow compared to hardware versions
- The recommended hash function for many UNIX systems, including Linux, Solaris, and FreeBSD is based on the MD5 secure hash algorithm
- The most secure version of the UNIX hash/salt scheme was developed for OpenBSD and uses a hash function (Bcrypt) based on the Blowfish symmetric block cipher
  - Slow [user set] function

Token-Based Authentication

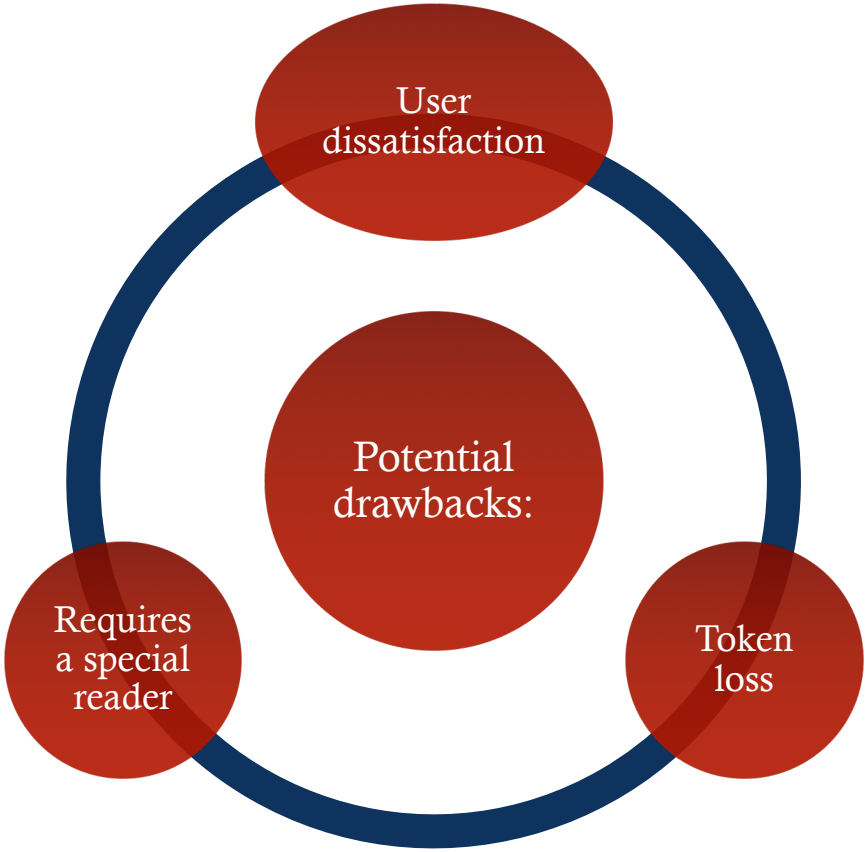
- Objects that a user possesses for the purpose of user authentication are called tokens



Memory Cards

- Memory cards can store but not process data
  - the most common is the bank card with a magnetic stripe on the back
  - a magnetic stripe can store only a simple security code which can be read and reprogrammed by an inexpensive card reader
  - there are also memory cards that include an internal electronic memory

- Can be used alone for physical access or with some form of password or personal identification number (PIN)



Smart Cards

Physical characteristics:

- include an embedded processor
- a smart token that looks like a bank card is called a smart card
- smart tokens can look like calculators, keys, or other small portable objects

Interface

- manual interfaces include a keypad and display for human/token interaction
- smart tokens with an electronic interface communicate with a compatible reader/writer

Authentication protocol

- static
- dynamic password generator
- challenge-response

Static Biometric Authentication

- Attempts to authenticate an individual based on his or her unique physical characteristics

- Includes static characteristics such as:
  - fingerprints
  - hand geometry
  - facial characteristics
  - retinal and iris patterns

- Dynamic characteristics such as:
  - voiceprint
  - signature

Physical Characteristics

Facial characteristics

- most common means of human-to-human identification
- examples: eyes, eyebrows, nose, lips, and chin shape
- alternative approach is to use an infrared camera to produce a face thermogram

Fingerprints

- pattern of ridges and furrows on the surface of the fingertip
- unique across the entire human population

Hand geometry

- identify features of the hand, including shape, and lengths and widths of fingers

Retinal pattern

- the pattern formed by veins beneath the retinal surface is unique
- a retinal biometric system obtains a digital image of the retinal pattern by projecting a low-intensity beam of visual or infrared light into the eye

Iris

- the detailed structure of the iris is unique

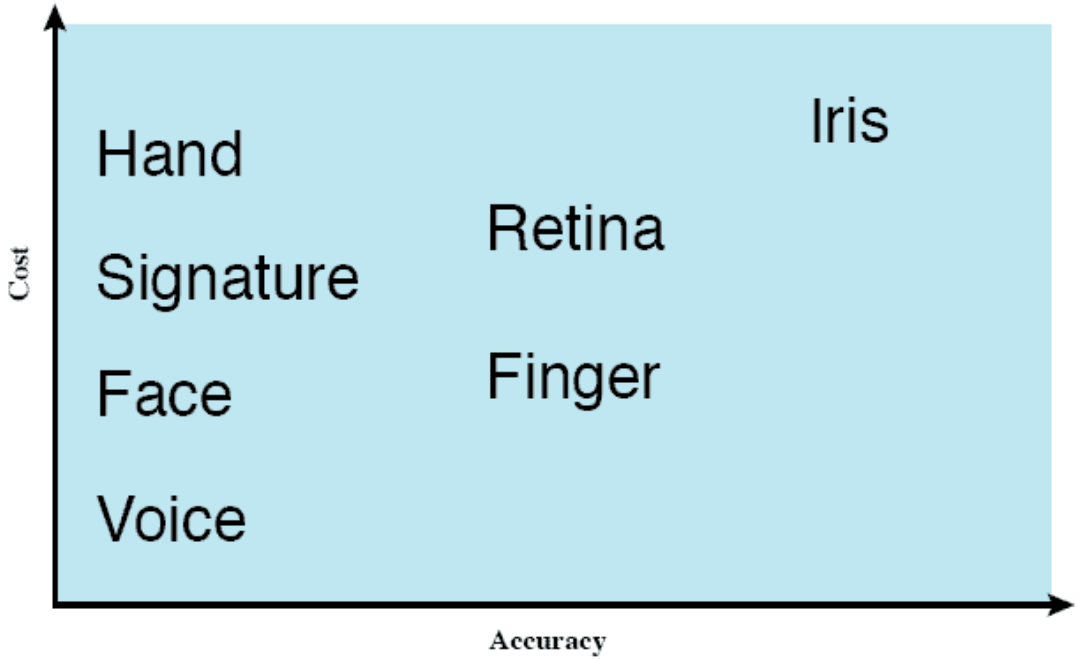
Signature

- each individual has a unique style of handwriting

Voice

- voice patterns are closely tied to the physical and anatomical characteristics of the speaker

Cost versus Accuracy



THE UNIVERSITY OF NEWCASTLE AUSTRALIA

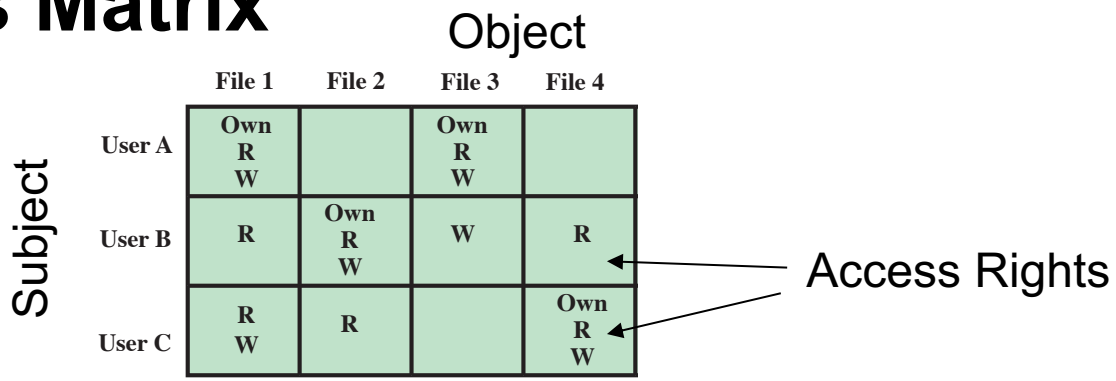
Access Control

- Dictates what types of access are permitted, under what circumstances, and by whom
- Access control is exercised by the OS, by the file system or at both levels
- Principles that have been typically applied are the same at the both levels

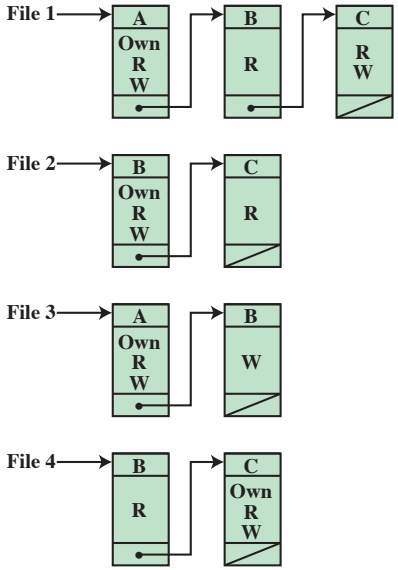
File System Access Control

- Identifies a user to the system
- Associated with each user there can be a profile that specifies permissible operations and file accesses
- The operating system can then enforce rules based on the user profile
- The database management system, however, must control access to specific records or even portions of records
- The database management system decision for access depends not only on the user's identity but also on the specific parts of the data being accessed and even on the information already divulged to the user

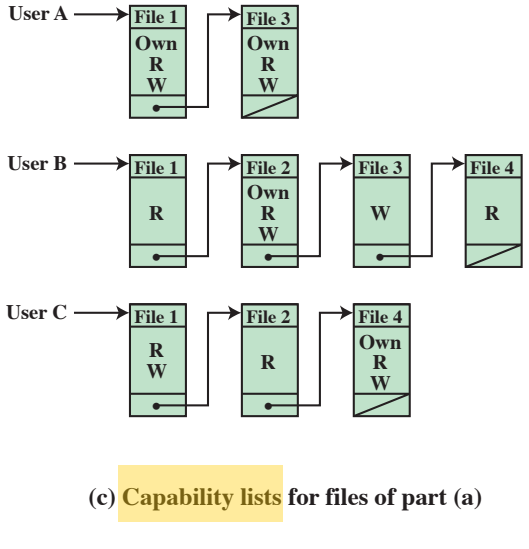
Access Matrix



(a) Access matrix



(b) Access control lists for files of part (a)



(c) Capability lists for files of part (a)

THE UNIVERSITY OF NEWCASTLE AUSTRALIA

Access Control Policies

- **Discretionary access control (DAC)**
  - controls access based on the identity of the requestor and on access rules stating what requestors are (or are not) allowed to do
- **Mandatory access control (MAC)**
  - controls access based on comparing security labels with security clearances
- **Role-based access control (RBAC)**
  - controls access based on the roles that users have within the system and on rules stating what accesses are allowed to users in given roles

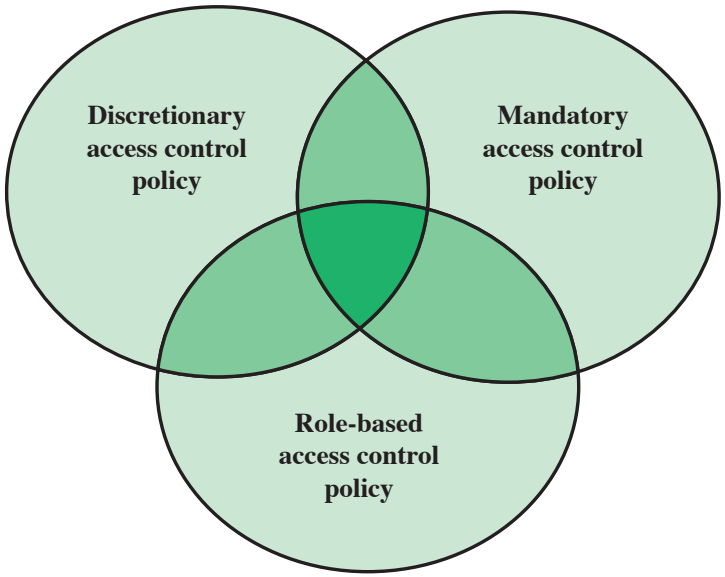


Figure 15.4 Access Control Policies

THE UNIVERSITY OF NEWCASTLE AUSTRALIA

Extended Access Control Matrix for DAC

	OBJECTS								
	subjects			files		processes		disk drives	
SUBJECTS	S_1	S_2	S_3	F_1	F_2	P_1	P_2	D_1	D_2
S_1	control	owner	owner control	read *	read owner	wakeup	wakeup	seek	owner
S_2		control		write *	execute			owner	seek *
S_3			control		write	stop			

* - copy flag set

Figure 15.5 Extended Access Control Matrix

THE UNIVERSITY OF NEWCASTLE AUSTRALIA

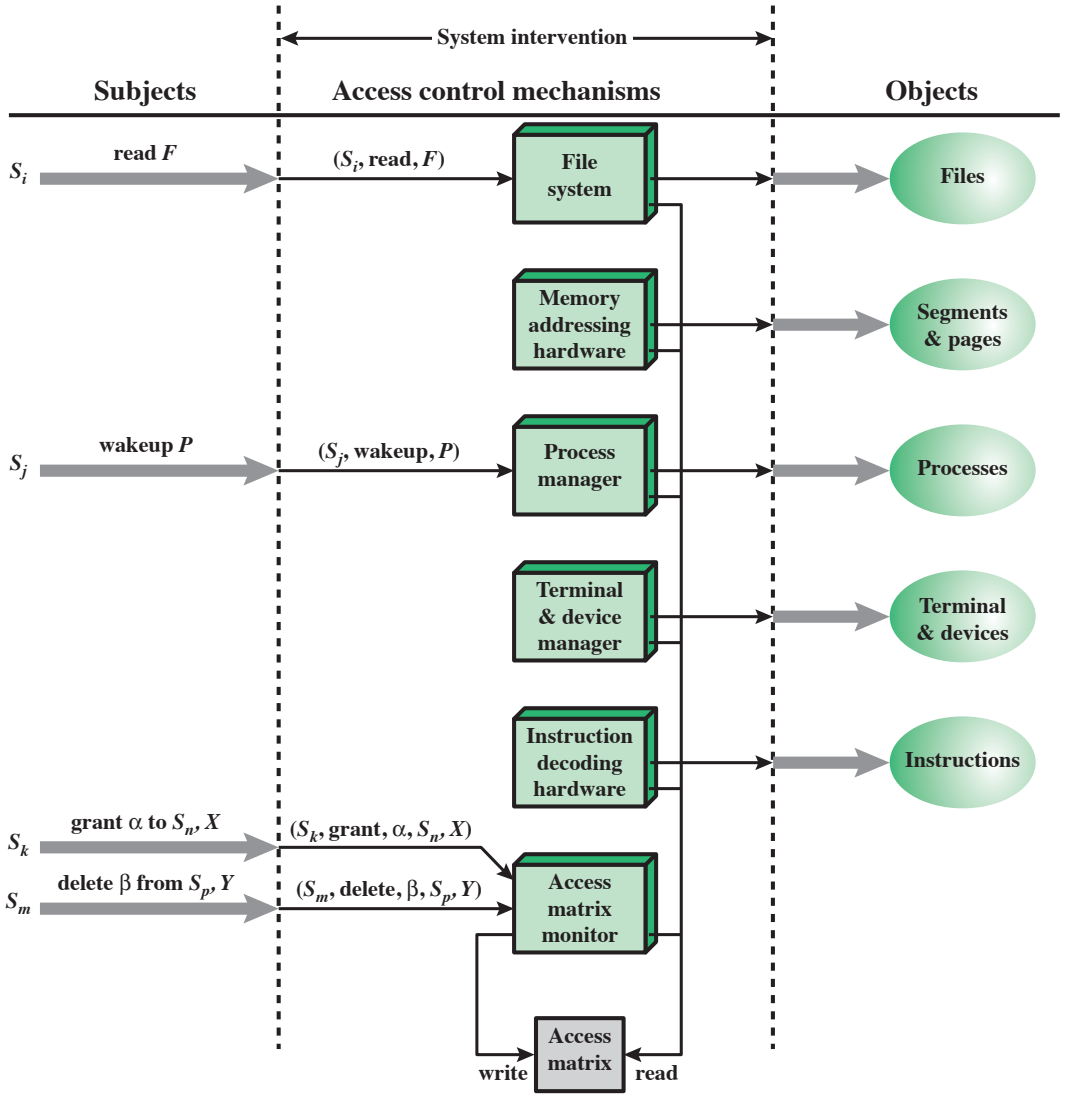


Figure 15.6 An Organization of the Access Control Function

THE UNIVERSITY OF NEWCASTLE AUSTRALIA

	OBJECTS								
	subjects			files		processes		disk drives	
SUBJECTS	S_1	S_2	S_3	F_1	F_2	P_1	P_2	D_1	D_2
S_1	control	owner	owner control	read *	read owner	wakeup	wakeup	seek	owner
S_2		control		write *	execute			owner	seek *
S_3			control		write	stop			

Rule	Command (by S_o)	Authorization	Operation
R1	**transfer** $\begin{Bmatrix}\alpha* \\ \alpha\end{Bmatrix}$ **to** S, X	'α*' in $A[S_o, X]$	store $\begin{Bmatrix}\alpha* \\ \alpha\end{Bmatrix}$ in $A[S, X]$
R2	**grant** $\begin{Bmatrix}\alpha* \\ \alpha\end{Bmatrix}$ **to** S, X	'owner' in $A[S_o, X]$	store $\begin{Bmatrix}\alpha* \\ \alpha\end{Bmatrix}$ in $A[S, X]$
R3	**delete** α **from** S, X	'control' in $A[S_o, S]$ or 'owner' in $A[S_o, X]$	delete α from $A[S, X]$
R4	$w \leftarrow$ **read** S, X	'control' in $A[S_o, S]$ or 'owner' in $A[S_o, X]$	copy $A[S, X]$ into w
R5	**create object** X	None	add column for X to A; store 'owner' in $A[S_o, X]$
R6	**destroy object** X	'owner' in $A[S_o, X]$	delete column for X from A
R7	**create subject** S	none	add row for S to A; execute **create object** S; store 'control' in $A[S, S]$
R8	**destroy subject** S	'owner' in $A[S_o, S]$	delete row for S from A; execute **destroy object** S

Role-based Access Control (RBAC)

- Based on the roles that user assume in a system rather than the user's identity
- Defines a role as a job function with an organization
- Access rights are assigned to roles and users are assigned to different roles (statically or dynamically)
- Relationship of users to roles (and roles to resources) is many to many

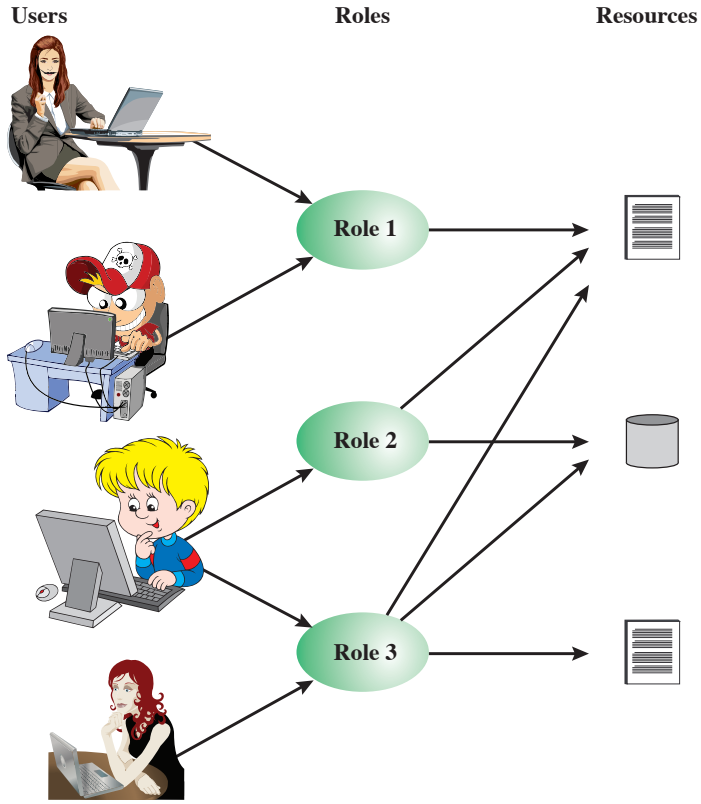
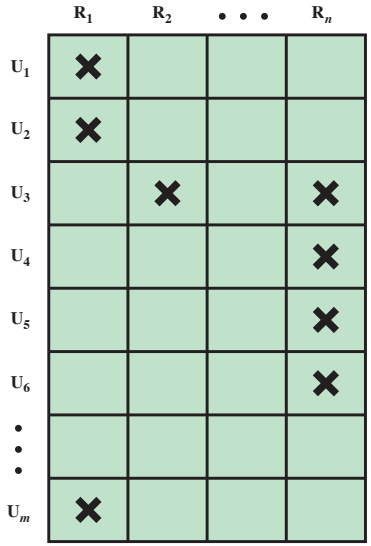


Figure 15.7 Users, Roles, and Resources

THE UNIVERSITY OF NEWCASTLE AUSTRALIA

Access Matrices for RBAC



Principle of least privilege

	OBJECTS								
ROLES	R1	R2	Rn	F1	F1	P1	P2	D1	D2
R1	control	owner	owner control	read *	read owner	wakeup	wakeup	seek	owner
R2		control		write *	execute			owner	seek *
• • •									
Rn			control		write	stop			

Figure 15.8 Access Control Matrix Representation of RBAC

Intrusion Detection

- RFC 4949 (*Internet Security Glossary)* defines intrusion detection as a security service that monitors and analyzes system events for the purpose of finding, and providing real-time or near real-time warning of, attempts to access system resources in an unauthorized manner
- Intrusion detection systems (IDSs) can be classified as:
    - **host-based IDS**
        - » monitors the characteristics of a single host and the events occurring within that host for suspicious activity
    - **network-based IDS**
        - » monitors network traffic for particular network segments or devices and analyzes network, transport, and application protocols to identify suspicious activity

Intrusion Detection - Basic Principles

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

- An effective IDS can serve as a deterrent, thus acting to prevent intrusions
- Intrusion detection enables the collection of information about intrusion techniques that can be used to strengthen instrusion prevention measures

Profiles of Behavior

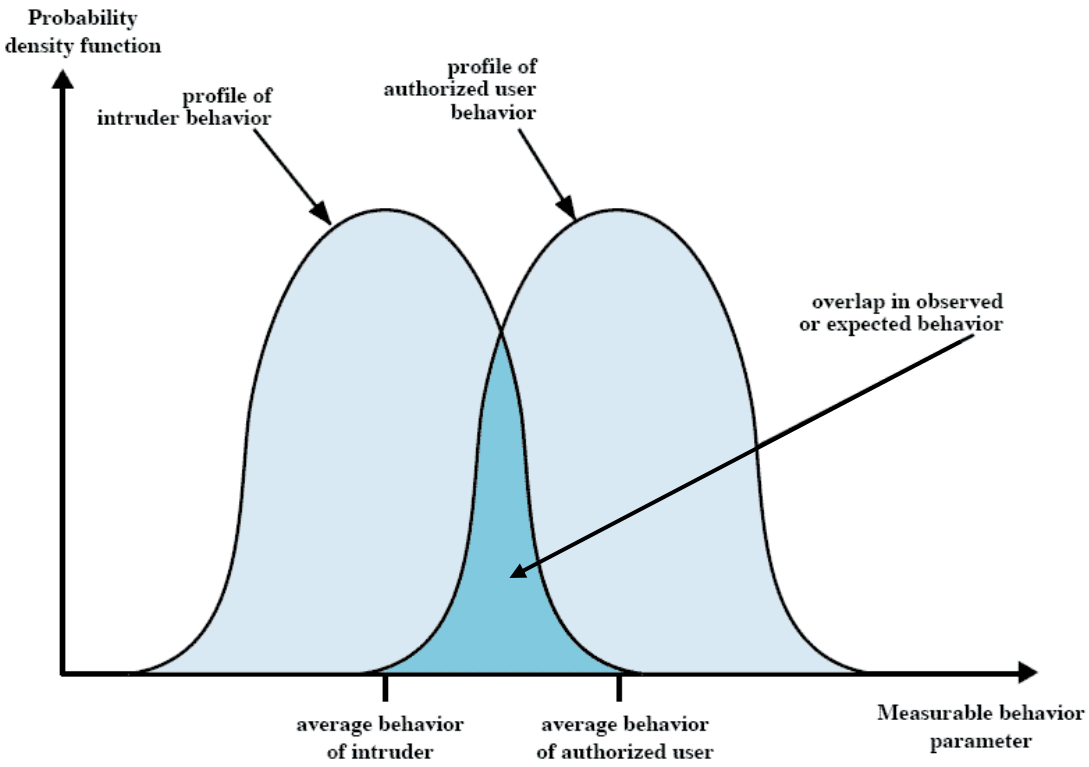


Figure 15.8 Profiles of Behavior of Intruders and Authorized Users

THE UNIVERSITY OF NEWCASTLE AUSTRALIA

IDS Components

Sensors

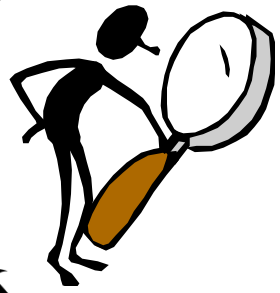
- responsible for collecting data
- the input for a sensor may be any part of a system that could contain evidence of an intrusion
- types of input to a sensor include network packets, log files, and system call traces

Analyzers

- receive input from one or more sensors or from other analyzer
- responsible for determining if an intrusion has occurred
- may provide guidance about what actions to take as a result of the intrusion

User interface

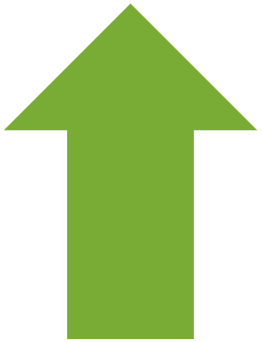
- enables a user to view output from the system or control the behavior of the system
- may equate to a manager, director, or console component



Audit Records

Native

Virtually all multi-user OS include accounting software that collects information on user activity



Advantage

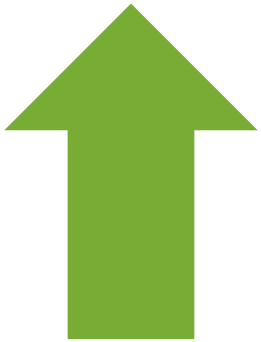
- no additional collection software is needed

Disadvantage

- the native audit records may not contain the needed information or may not contain it in a convenient form

Detection-specific

A collection facility that generates audit record containing only that information required by the IDS



Advantage

- it could be made vendor independent and ported to a variety of systems

Disadvantage

- the extra overhead involved in having, in effect, two accounting packages running on a machine

Detection-specific audit records: Example

- **Fields**: Subject, Action, Object, Exception-condition, Resource-usage, Time-stamp
- Suppose Smith issue the command

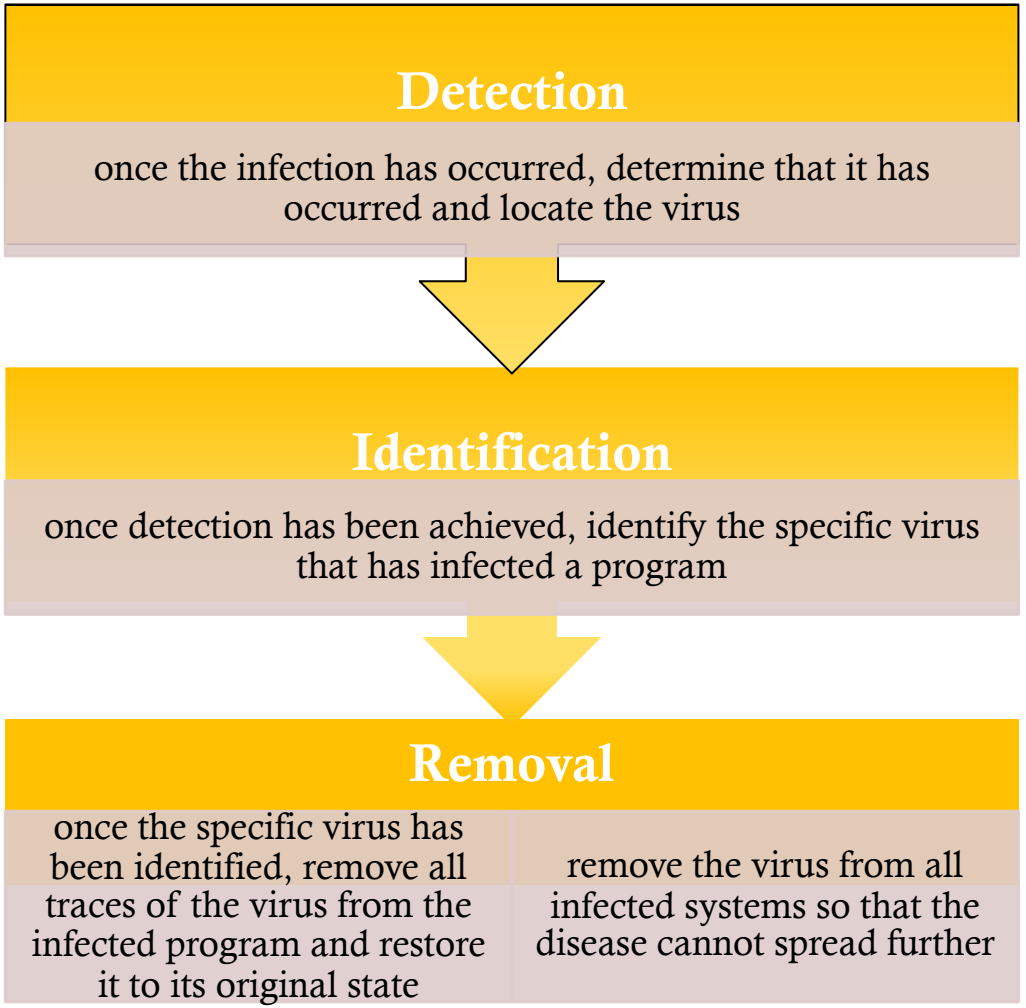
  COPY GAME.EXE TO <Library>GAME.EXE

  The following audit records may be generated

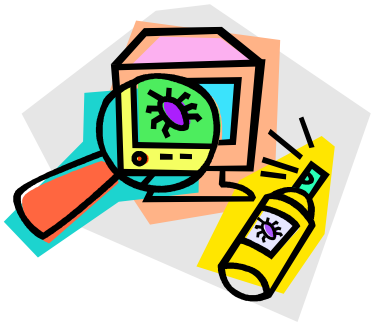
Smith	execute	<Library>COPY.EXE	0	CPU = 00002	11058721678
Smith	Read	<Smith>GAME.EXE	0	RECORDS = 0	11058721679
Smith	execute	<Library>COPY.EXE	write-viol	RECORDS = 0	11058721680

Antivirus Approaches

- Ideal solution to the threat of viruses is prevention, don't allow a virus onto the system in the first place!
- That goal is, in general, impossible to achieve, although prevention can reduce the number of successful viral attacks
- If detection succeeds but either identification or removal is not possible, then the alternative is to discard the infected program and reload a clean backup version



Generic Decryption (GD)

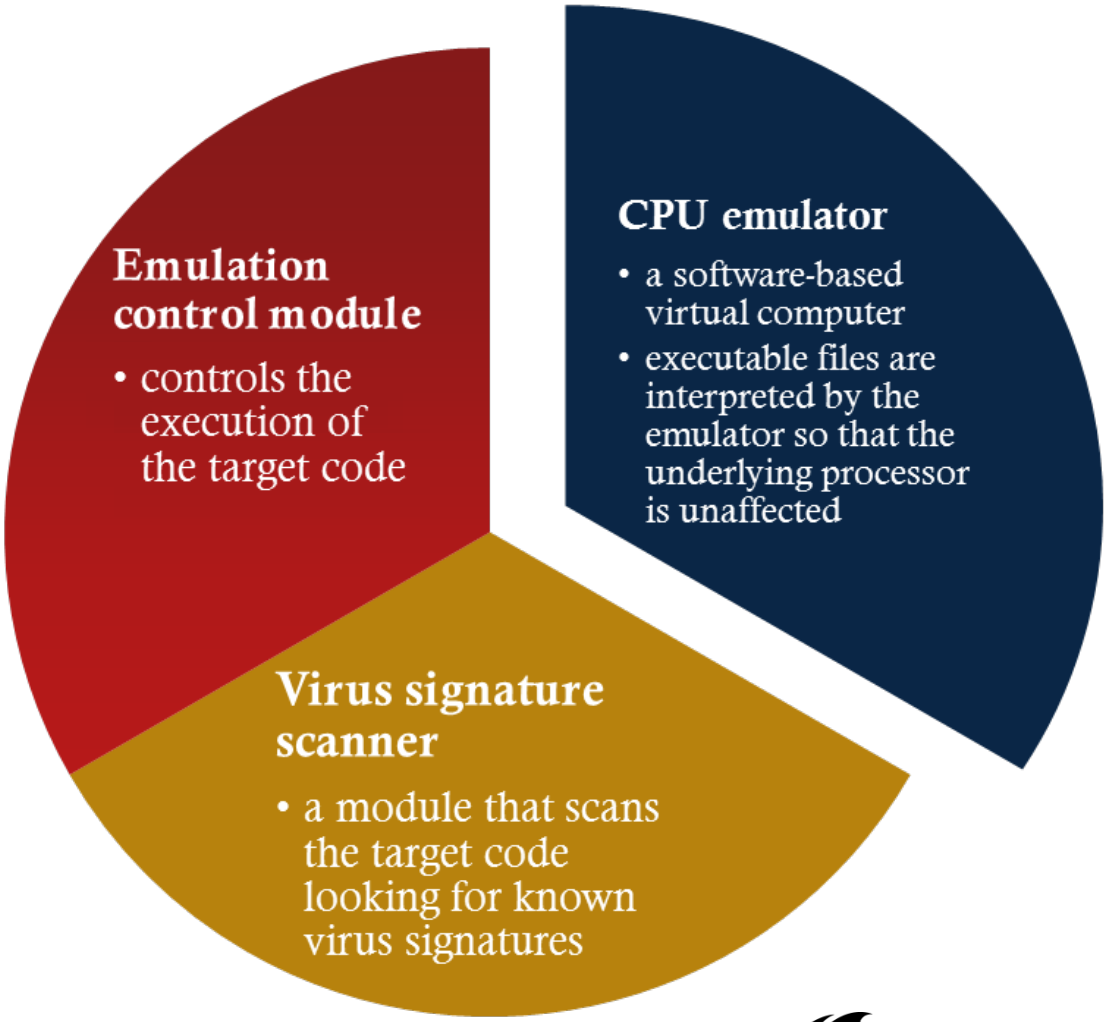


- Enables the antivirus program to easily detect even the most complex polymorphic viruses while maintaining fast scanning speeds
- When a file containing a polymorphic virus is executed, the virus must decrypt itself to activate
- Executable files are run through a GD scanner

GD Scanner

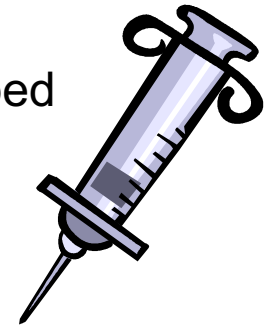
- GD scanner contains:

- Most difficult design issue with a GD scanner is to determine how long to run each interpretation



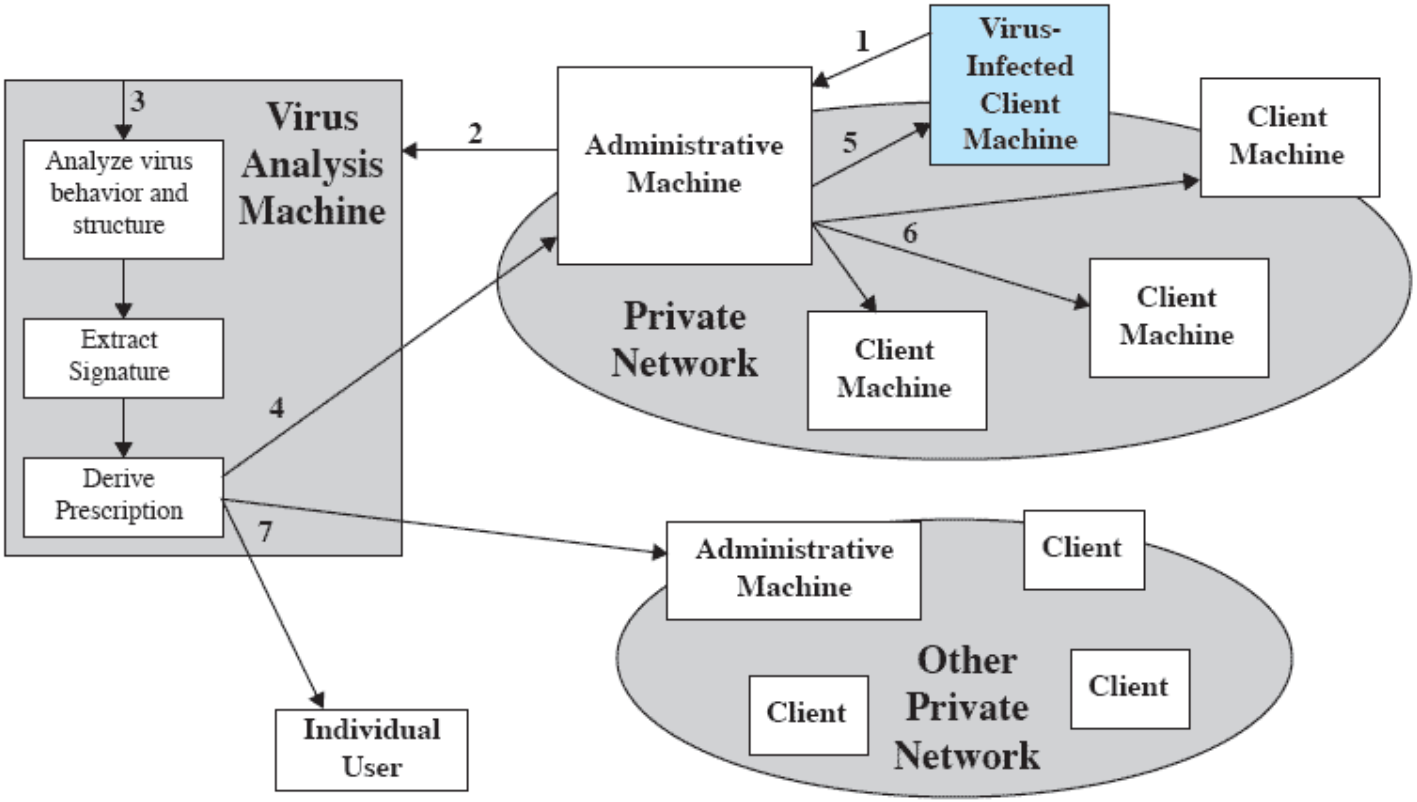
Digital Immune System

- A comprehensive approach to virus protection developed by IBM and refined by Symantec



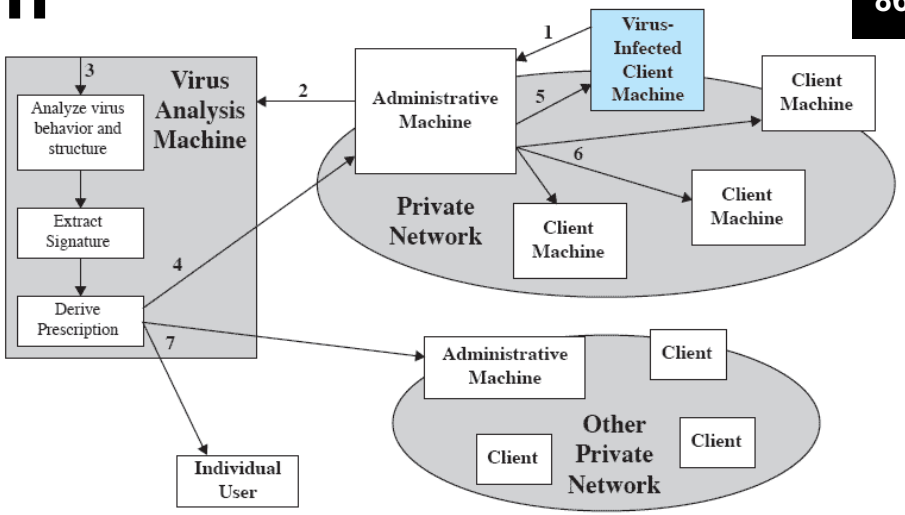
- Motivation for development has been the rising threat of Internet-based virus propagation
- Two major trends in Internet technology have had an increasing impact on the rate of virus propagation in recent years:
  - integrated mail systems
  - mobile-program systems
- Objective of the system is to provide rapid response time so that viruses can be stamped out almost as soon as they are introduced

Digital Immune System



THE UNIVERSITY OF NEWCASTLE AUSTRALIA

Digital Immune System



1. A monitoring program infers that a virus may be present. A copy of any program thought to be infected is forwarded 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.

THE UNIVERSITY OF NEWCASTLE AUSTRALIA

Summary

- Intruders and malicious software
  - system access threats
  - countermeasures
- Buffer overflow
  - buffer overflow attacks
  - compile time defenses
  - runtime defenses
- Access control
  - file system access control
  - access control policies
- Operating systems hardening
  - OS installation: initial setup and patching
  - remove unnecessary services, application, and protocols
  - configure users, groups and authentication
  - install additional security controls
  - test the system security

References

- **Operating Systems – Internal and Design Principles (Eighth Edition/Ninth Edition)**
  - By William Stallings
  - Chapter 15

- **Operating Systems – Internal and Design Principles (Seventh Edition)**
  - By William Stallings
  - Chapter 14, 15