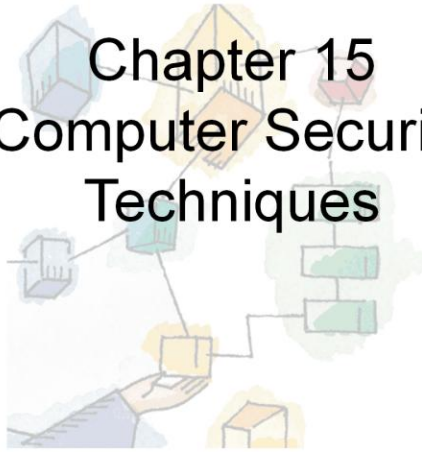


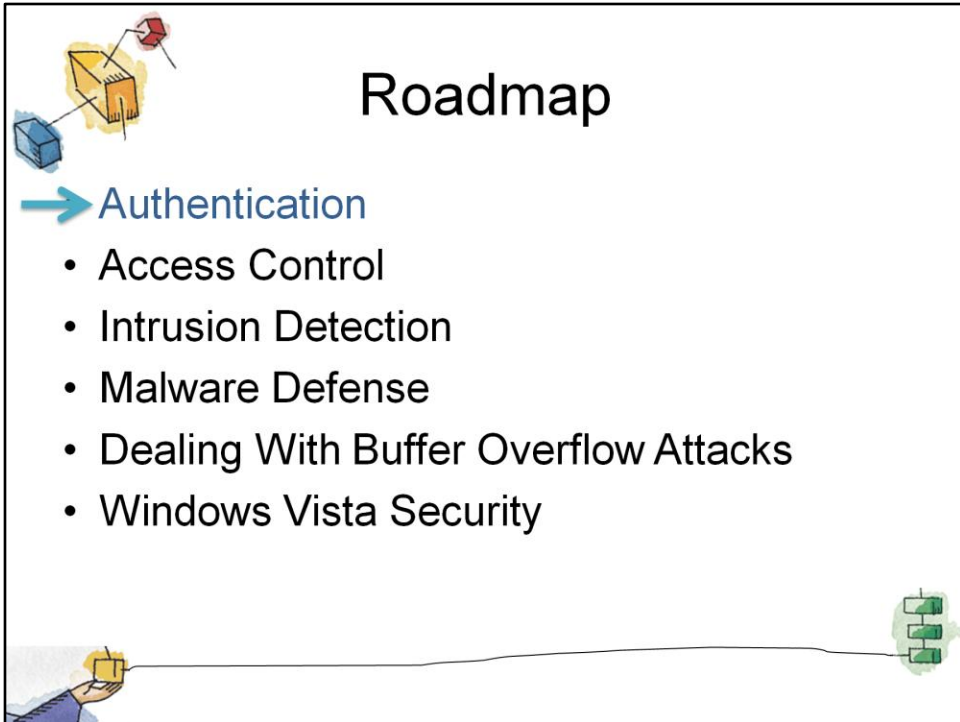
*Operating Systems:  
Internals and Design Principles, 6/E*  
William Stallings

# Chapter 15 Computer Security Techniques



Dave Bremer  
Otago Polytechnic, N.Z.  
©2008, Prentice Hall

These slides are intended to help a teacher develop a presentation. This PowerPoint covers the entire chapter and includes too many slides for a single delivery. Professors are encouraged to adapt this presentation in ways which are best suited for their students and environment.



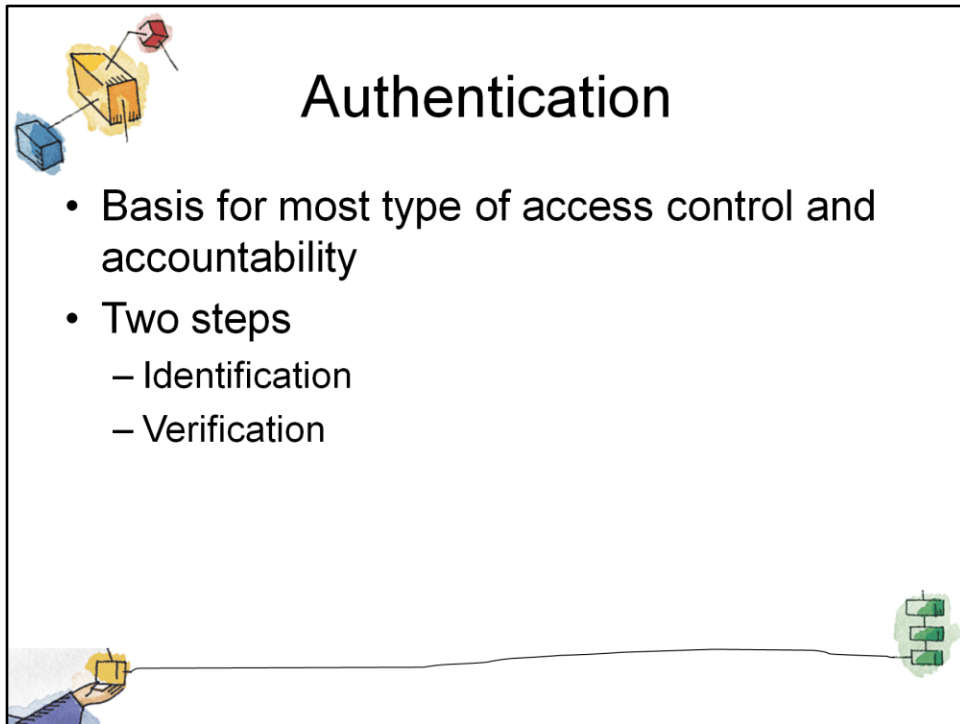
The diagram is enclosed in a black rectangular border. In the top-left corner, there are three icons: a blue cube, a yellow cube with a red dot on top, and a red cube. In the bottom-left corner, a hand is shown holding a yellow cube. In the bottom-right corner, there is a green icon representing a server rack. A thin black line starts from the hand and extends horizontally across the bottom of the diagram towards the server rack icon.

# Roadmap

→ Authentication

- Access Control
- Intrusion Detection
- Malware Defense
- Dealing With Buffer Overflow Attacks
- Windows Vista Security

This chapter introduces common measures used to counter the security threats discussed in Chapter 14



In most computer security contexts, user authentication is the fundamental building block and the primary line of defense.

User authentication is the basis for most types of access control and for user accountability.


An authentication process consists of two steps:

**Identification step:**

- Presenting an identifier to the security system.
- Identifiers should be assigned carefully, because authenticated identities are the basis for other security services, such as access control service.


**Verification step:**

- Presenting or generating authentication information that corroborates the binding between the entity and the identifier.



# Means of Authentication

- Traditionally listed as three factors
- Something you **know**
  - Password, PIN
- Something you **have**
  - Card, RFID badge
- Something you **are**
  - Biometrics



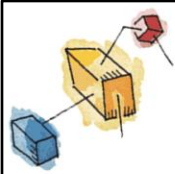
All of these methods, properly implemented and used, can provide secure user authentication.

However, each method has problems.

- An adversary may be able to guess or steal a password.
- or may be able to forge or steal a token.
- or a user may forget a password or lose a token.

Further, there is a significant administrative overhead for managing password and token information on systems and securing such information on systems.

With respect to biometric authenticators, there are a variety of problems, including dealing with false positives and false negatives, user acceptance, cost, and convenience.



## A different take

- Nick Mathewson is attributed with turning these factors into:
  - Something you had,
  - Something you forgot,
  - Something you were!




Source of quote attribution <http://www.links.org/?p=326>



## Biometrics expanded


- Recently Biometrics (something you are) has been expanded into:
- Something the individual *is*
  - **Static Biometrics:** Fingerprint, face
- Something the individual *does*
  - **Dynamic Biometrics:** handwriting, voice recognition, typing rhythm





# Password-Based Authentication

- Determines if user is authorized to access the system
- Determines privileges for the user
- Discretionary access control may be applied



Identification is the means by which a user provides a claimed identity to the system;

- user authentication is the means of establishing the validity of the claim.

The ID determines:

**Whether the user is authorized** to gain access to a system.

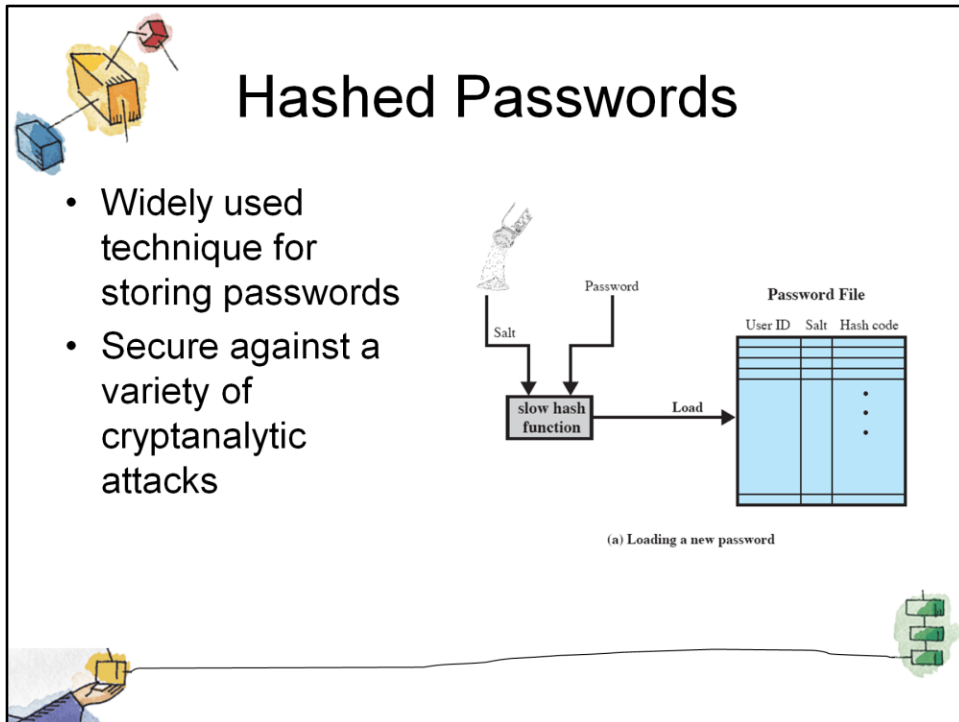
- In some systems, only those who already have an ID filed on the system are allowed to gain access.

**The privileges** accorded to the user.

- A few users may have supervisory or “superuser” status that enables them to read files and perform functions that are especially protected by the operating system.
- Some systems have guest or anonymous accounts, and users of these accounts have more limited privileges than others.

**The discretionary access control.**

- For example, by listing the IDs of the other users, a user may grant permission to them to read files owned by that user.



A widely used password security technique is the use of hashed passwords and a salt value.

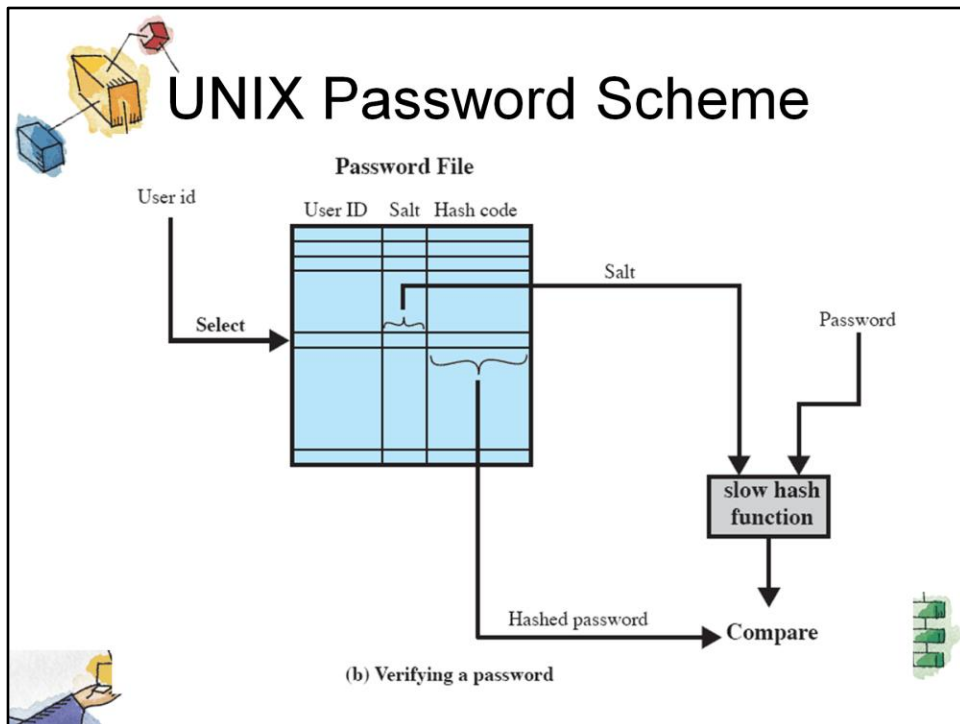
- This scheme is found on virtually all UNIX variants as well as on a number of other operating systems.

The following procedure is employed

- To load a new password into the system, the user selects or is assigned a password.
- This password is combined with a fixed-length salt value
- The password and salt serve as inputs to a hashing algorithm to produce a fixed-length hash code.
- The hash algorithm is designed to be slow to execute to thwart attacks.
- The hashed password is then stored, together with a plaintext copy of the salt, in the password file for the corresponding user ID.

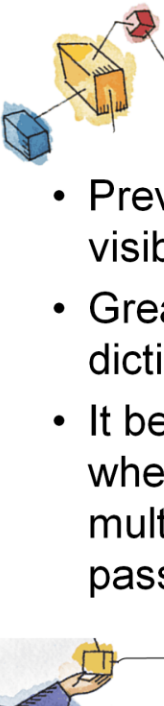
The hashed-password method has been shown to be secure against a variety of cryptanalytic attacks






When a user attempts to log on to a UNIX system,

- the user provides an ID and a password.
- The operating system uses the ID to index into the password file and retrieve the plaintext salt and the encrypted password.
- The salt and user-supplied password are used as input to the encryption routine.
- If the result matches the stored value, the password is accepted.



# Salt

- Prevents duplicate passwords from being visible in the password file.
- Greatly increases the difficulty of offline dictionary attacks.
- It becomes nearly impossible to find out whether a person with an account on multiple systems has used the same password for all.



The salt serves three purposes:


It prevents duplicate passwords from being visible in the password file.

- Even if two users choose the same password, those passwords will be assigned different salt values. Hence, the hashed passwords of the two users will differ.

It greatly increases the difficulty of offline dictionary attacks.


- For a salt of length  $b$  bits, the number of possible passwords is increased by a factor of  $2^b$ , increasing the difficulty of guessing a password in a dictionary attack.

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.

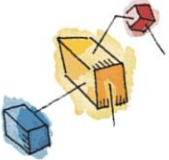


# Token-Based Authentication

- Objects that a user possesses for the purpose of user authentication are called tokens.
- Examples include
  - Memory cards
  - Smart cards





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



# Memory Cards

- Memory cards can store but not process data.
- Often used in conjunction with password or ping
- Drawbacks include
  - Requires a special reader
  - Token loss
  - User dissatisfaction



Memory cards can store but not process data.

- The most common such card 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 unfortunately reprogrammed) by an inexpensive card reader.
- There are also memory cards that include an internal electronic memory.

The memory card, when combined with a PIN or password, provides significantly greater security than a password alone.

- An adversary must gain physical possession of the card (or be able to duplicate it) plus must gain knowledge of the PIN.

Among the potential drawbacks are the following

**Requires special reader:**

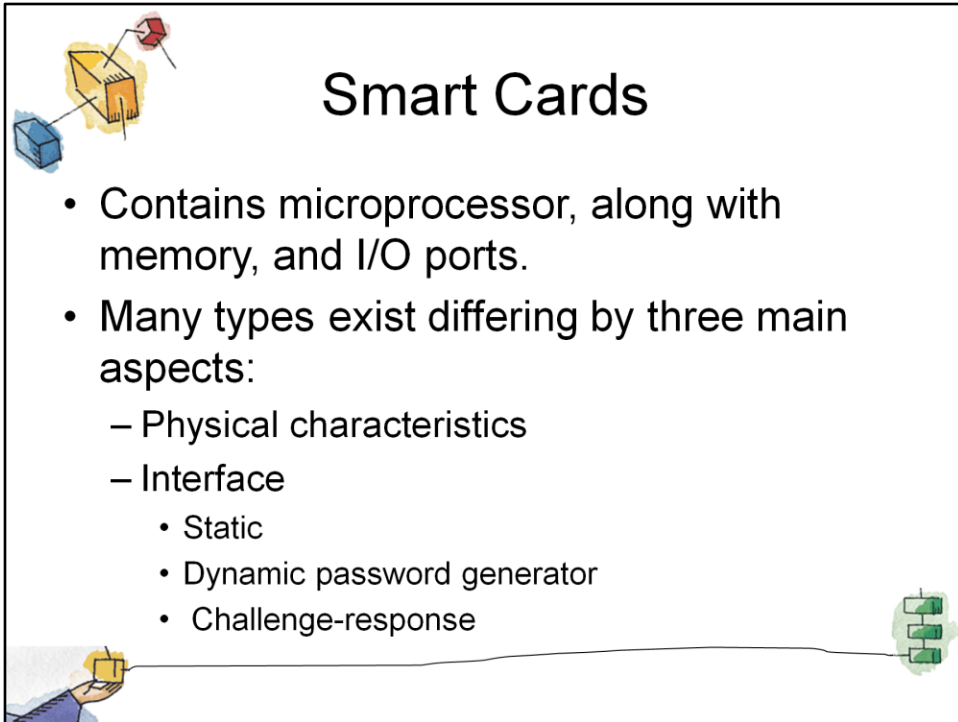
- This increases the cost of using the token and creates the requirement to maintain the security of the reader's hardware and software.

**Token loss:**

- A lost token temporarily prevents its owner from gaining system access giving an administrative cost in replacing the lost token.
- Also, if the token is found, stolen, or forged, then an adversary now need only determine the PIN to gain unauthorized access.

**User dissatisfaction:**

- Although users may have no difficulty in accepting the use of a memory card for ATM access, its use for computer access may be deemed inconvenient.



The diagram is enclosed in a black rectangular border. In the top-left corner, there is an illustration of a yellow smart card being inserted into a blue reader device. In the bottom-left corner, a hand is shown holding a yellow smart card. In the bottom-right corner, there is a small green icon representing a microprocessor or chip. The title 'Smart Cards' is centered at the top in a large, black, sans-serif font.

# Smart Cards

- Contains microprocessor, along with memory, and I/O ports.
- Many types exist differing by three main aspects:
  - Physical characteristics
  - Interface
    - Static
    - Dynamic password generator
    - Challenge-response

A smart card contains within it an entire microprocessor, including processor, memory, and I/O ports.

- Some versions incorporate a special co-processing circuit for cryptographic operation to speed the task of encoding and decoding messages or generating digital signatures to validate the information transferred.

A wide variety of devices qualify as smart tokens.

These can be categorized along three dimensions that are not mutually exclusive:

**Physical characteristics:**

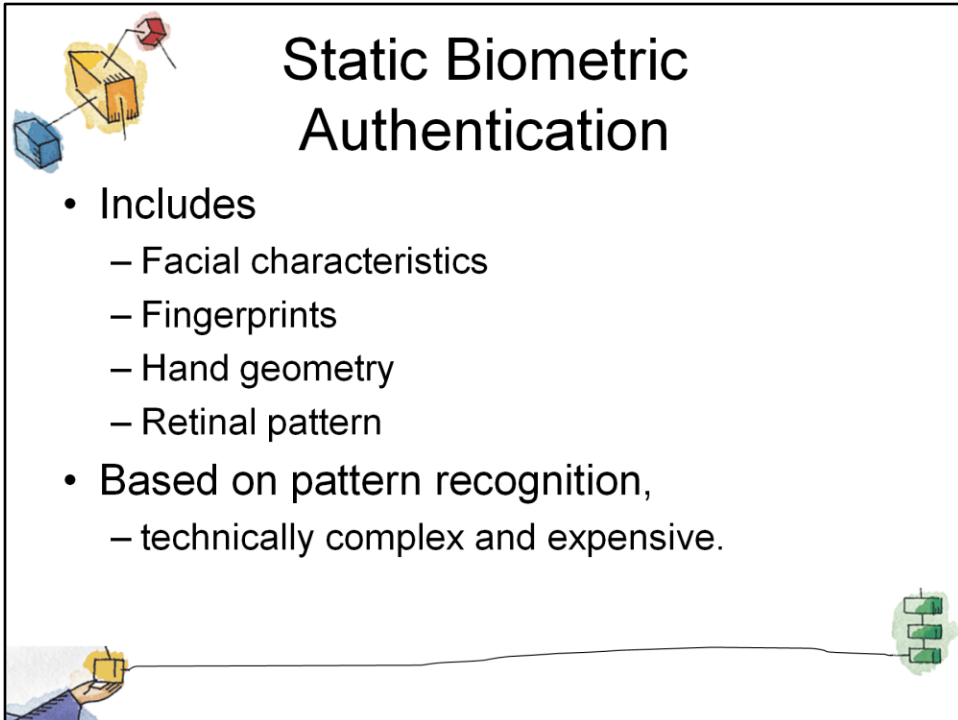
- Smart tokens include an embedded microprocessor.
- A smart token that looks like a bank card is called a smart card.
- Other smart tokens can look like calculators, keys, or other small portable objects.

**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:**

- to provide a means for user authentication.
  - **Static:** the user authenticates himself or herself to the token and then the token authenticates the user to the computer. The latter half of this protocol is similar to the operation of a memory token.
  - **Dynamic password generator:** the token generates a unique password periodically (e.g., every minute). This password is then entered into the computer system for authentication, either manually by the user or electronically via the token. The token and the computer system must be initialized and kept synchronized so that the computer knows the password that is current for this token.
  - **Challenge-response:** The computer system generates a challenge, such as a random string of numbers. The smart token generates a response based on the challenge.



# Static Biometric Authentication

- Includes
  - Facial characteristics
  - Fingerprints
  - Hand geometry
  - Retinal pattern
- Based on pattern recognition,
  - technically complex and expensive.

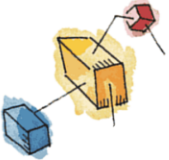
Attempts to authenticate an individual based on his or her unique physical characteristics.

Static characteristics include, fingerprints, hand geometry, facial characteristics, and retinal and iris patterns;

In essence, biometrics is based on pattern recognition.



Compared to passwords and tokens, biometric authentication is both technically complex and expensive.

- While it is used in a number of specific applications, biometrics has yet to mature as a standard tool for user authentication to computer systems.

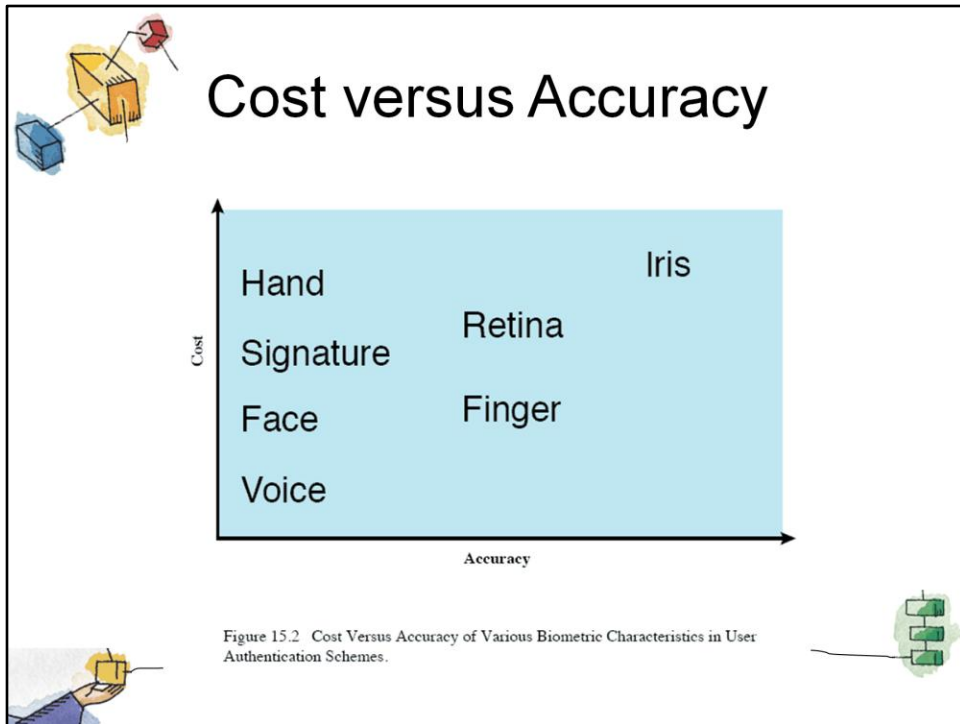


# Dynamic Biometric Authentication

- Patterns may change
- Includes
  - Iris
  - Signature
  - Voice
  - Typing rhythm



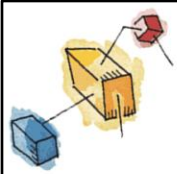
and dynamic characteristics, such as voiceprint and signature.



This figure gives a rough indication of the relative cost and accuracy of these biometric measures.

The concept of accuracy does not apply to user authentication schemes using smart cards or passwords

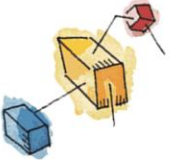




# Roadmap


- Authentication
- Access Control
- Intrusion Detection
- Malware Defense
- Dealing With Buffer Overflow Attacks
- Windows Vista Security





# Access Control

- Dictates what types of access are permitted, under what circumstances, and by whom.
  - Discretionary access control
  - Mandatory access control
  - Role-based access control



An access control policy dictates what types of access are permitted, under what circumstances, and by whom.

### **Discretionary access control (DAC):**

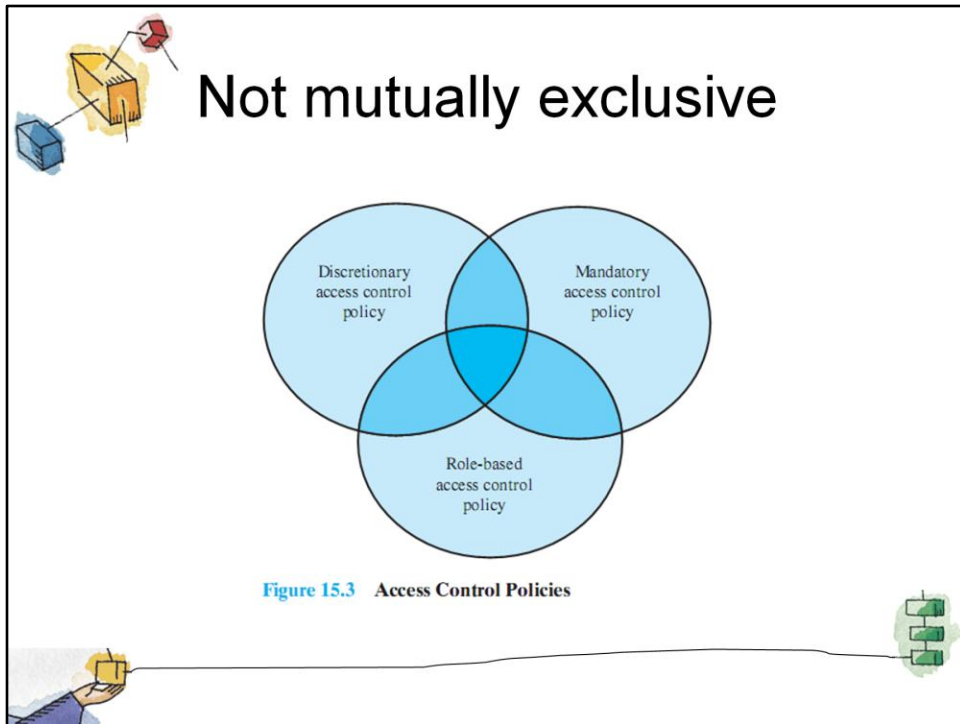
- Controls access based on the identity of the requestor and on access rules (authorizations) stating what requestors are (or are not) allowed to do.
- This policy is termed discretionary because an entity might have access rights that permit the entity, by its own volition, to enable another entity to access some resource.

### **Mandatory access control (MAC):**

- Security labels indicate how sensitive or critical system resources are
- Security clearances indicate which system entities are eligible to access certain resources
- MAC controls access based on comparing security labels with security clearances
- This policy is termed mandatory because an entity that has clearance to access a resource may not, just by its own volition, enable another entity to access that resource.

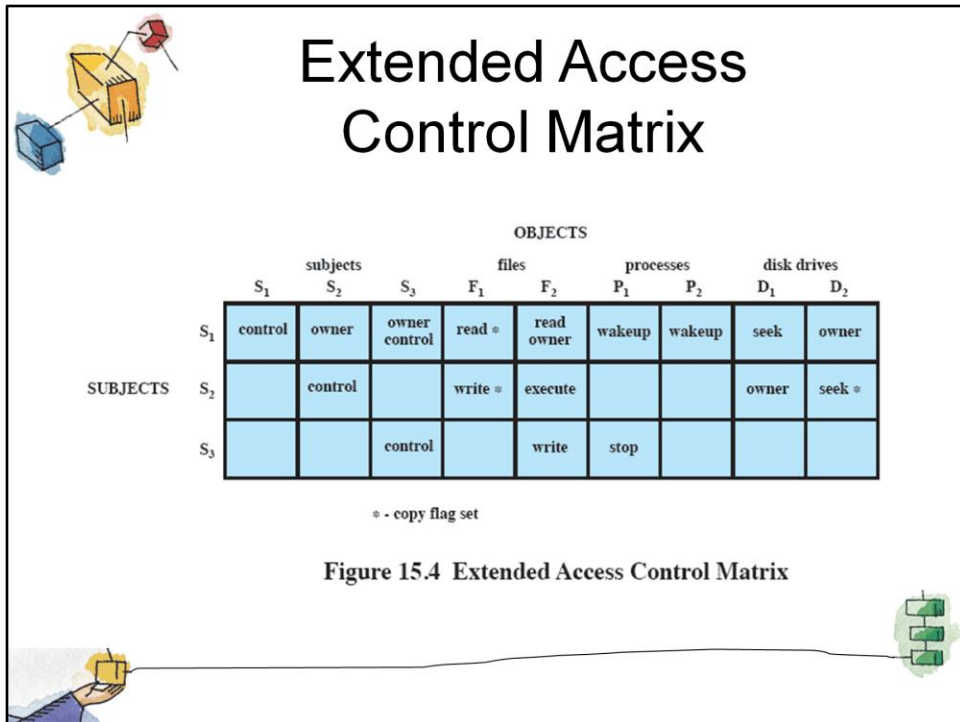
### **Role-based access control (RBAC):**

- 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.



These three policies are not mutually exclusive.

An access control mechanism can employ two or even all three of these policies to cover different classes of system resources.



To represent the protection state, we extend the universe of objects in the access control matrix to include the following:

**Processes:**

- Access rights include the ability to delete a process, stop (block), and wake up a process.

**Devices:**

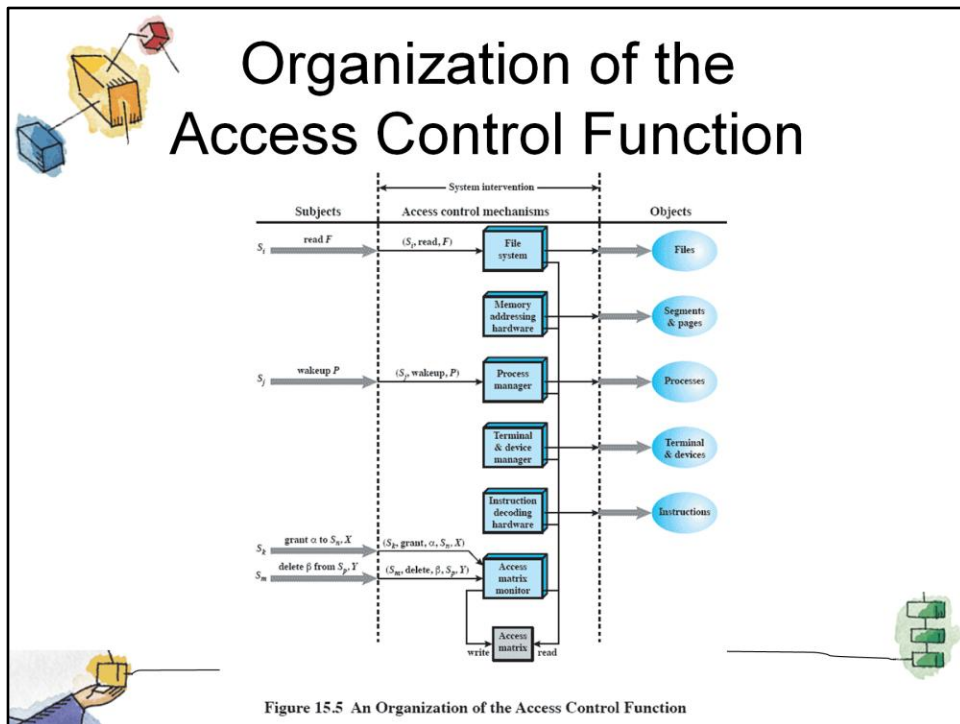
- Access rights include the ability to read/write the device, to control its operation (e.g., a disk seek), and to block/unblock the device for use.

**Memory locations or regions:**

- Access rights include the ability to read/write certain locations of regions of memory that are protected so that the default is that access is not allowed.

**Subjects:**

- Access rights with respect to a subject have to do with the ability to grant or delete access rights of that subject to other objects



From a logical or functional point of view, a separate access control module is associated with each type of object.


The module evaluates each request by a subject to access an object to determine if the access right exists.

An access attempt triggers the following steps:

1. A subject  $S_0$  issues a request of type  $\alpha$  for object  $X$ .
2. The request causes the system to generate a message of the form  $(S_0, \alpha, X)$  to the controller for  $X$ .
3. The controller interrogates the access matrix  $A$  to determine if  $\alpha$  is in  $A[S_0, X]$ .
  - If so, the access is allowed; if not, the access is denied and a protection violation occurs.
  - The violation should trigger a warning and appropriate action.


Every access by a subject to an object is mediated by the controller for that object, and that the controller's decision is based on the current contents of the matrix.

- Also, certain subjects have the authority to make specific changes to the access matrix.
- A request to modify the access matrix is treated as an access to the matrix, with the individual entries in the matrix treated as objects.
- Such accesses are mediated by an access matrix controller, which controls updates to the matrix.



## Role Based Access Control

- Effective implementation of the principle of least privilege
- Each role should contain the minimum set of access rights needed for that role.
- A user is assigned to a role that enables him or her to perform what is required for that role.
  - But only while they are performing that role

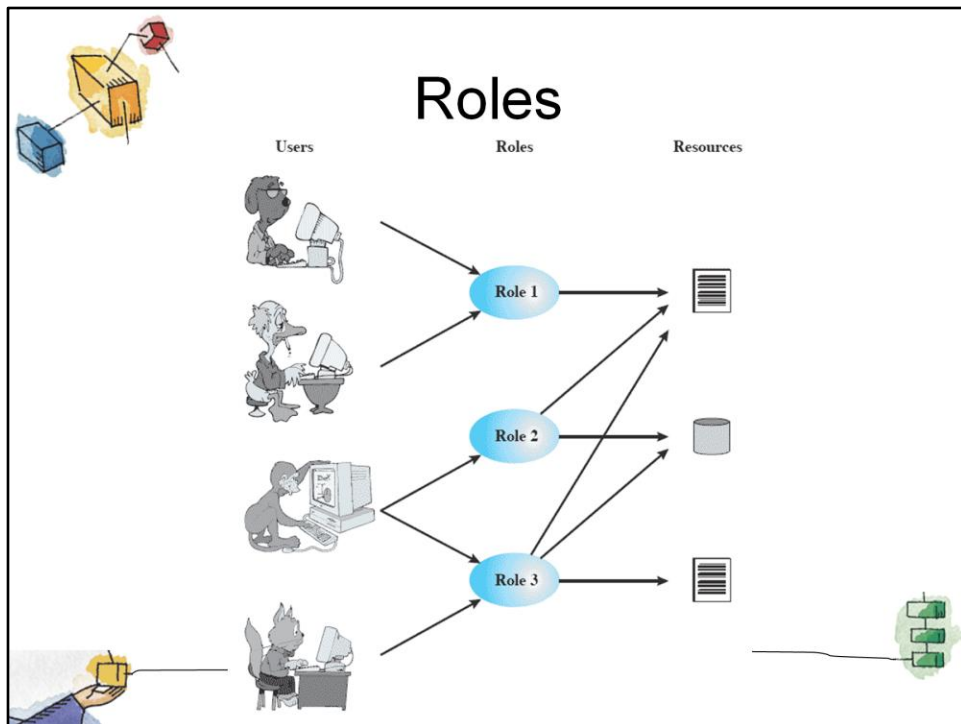


RBAC lends itself to an effective implementation of the principle of least privilege.

Each role should contain the minimum set of access rights needed for that role.

A user is assigned to a role that enables him or her to perform only what is required for that role.

Multiple users assigned to the same role enjoy the same minimal set of access rights.



Based on the roles that users assume in a system rather than the user's identity.

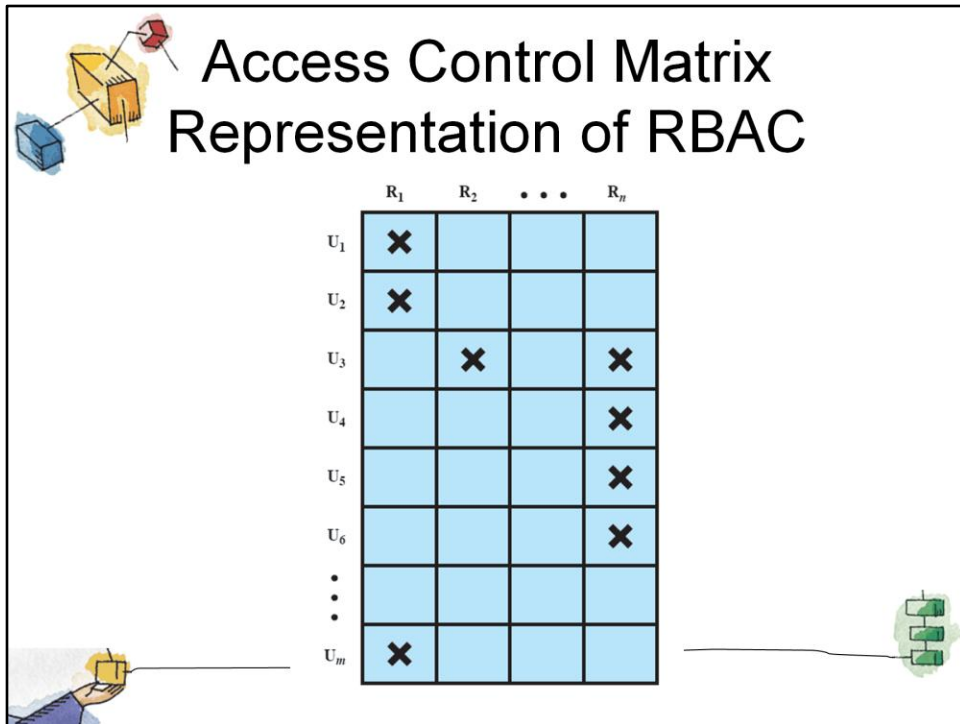
- Typically, RBAC models define a role as a job function within an organization.

RBAC systems assign access rights to roles instead of individual users.

- In turn, users are assigned to different roles, **either statically or dynamically**, according to their responsibilities.

The relationship of users to roles is many to many, as is the relationship of roles to resources, or system objects.

- The set of users changes, sometimes frequently, and the assignment of a user to one or more roles may also be dynamic.
- The set of roles in the system in most environments is likely to be static, with only occasional additions or deletions.
- Each role will have specific access rights to one or more resources.
- The set of resources and the specific access rights associated with a particular role are also likely to change infrequently.

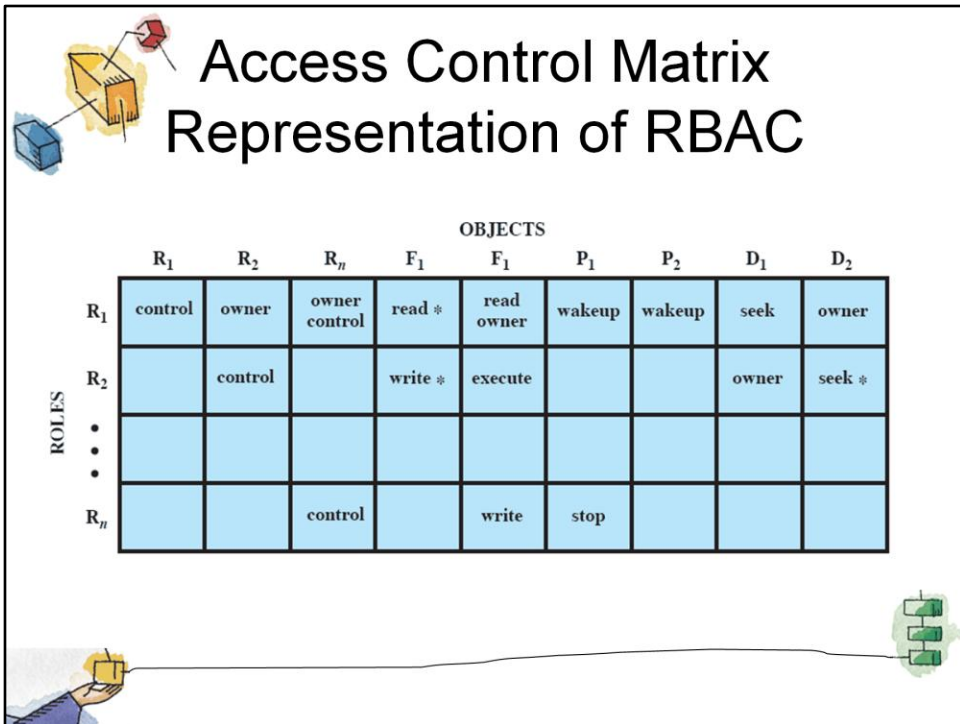


We can use the access matrix representation to depict the key elements of an RBAC system in simple terms

This matrix relates individual users to roles.

- Typically there are many more users than roles.
- Each matrix entry is either blank or marked, the latter indicating that this user is assigned to this role.
- **Note** that a single user may be assigned multiple roles (more than one mark in a row) and that multiple users may be assigned to a single role (more than one mark in a column).





The lower matrix has the same structure as the DAC access control matrix, with roles as subjects.

Typically, there are few roles and many objects or resources.


- In this matrix the entries are the specific access rights enjoyed by the roles.
- **Note** that a role can be treated as an object, allowing the definition of role hierarchies.



# Roadmap



- Authentication
- Access Control
- Intrusion Detection
- Malware Defense
- Dealing With Buffer Overflow Attacks
- Windows Vista Security





# Some Definitions

- **Security intrusion:**
  - A security event in which an intruder gains access to a system without authorization.
- **Intrusion detection:**
  - A security service that monitors and analyzes system events to find intrusions and provide alerts

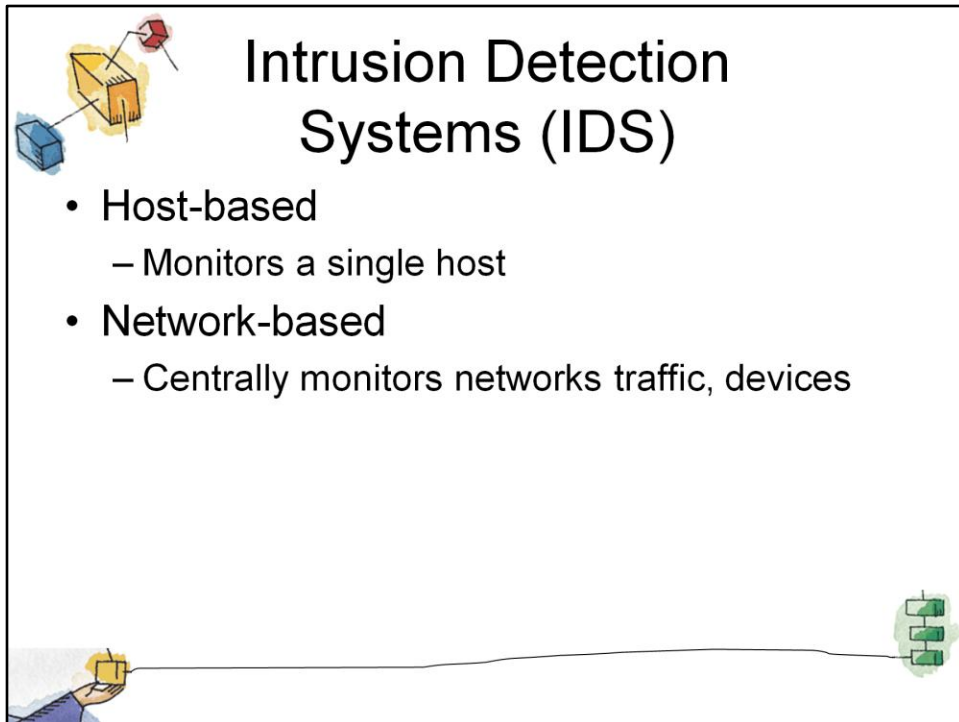


### **Security intrusion:**

- A security event, or a combination of multiple security events, that constitutes a security incident in which an intruder gains, or attempts to gain, access to a system (or system resource) without having authorization to do so.

### **Intrusion detection:**

- 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 (IDS)

- Host-based
  - Monitors a single host
- Network-based
  - Centrally monitors networks traffic, devices

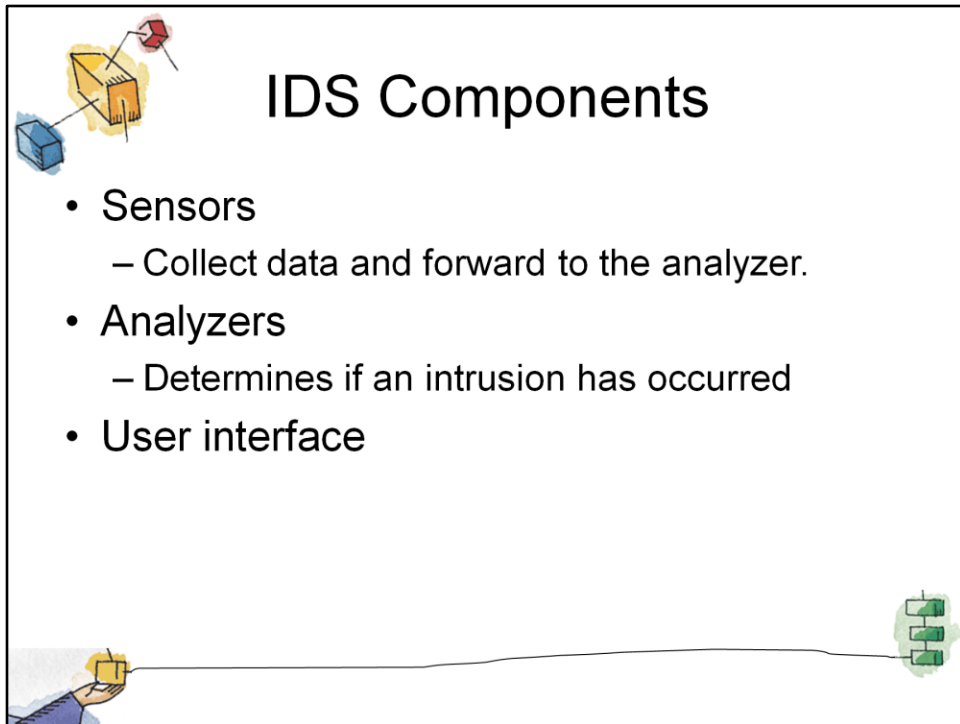
IDSs can be classified as:

**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



# IDS Components

- Sensors
  - Collect data and forward to the analyzer.
- Analyzers
  - Determines if an intrusion has occurred
- User interface

An IDS comprises three logical components:

## **Sensors:**

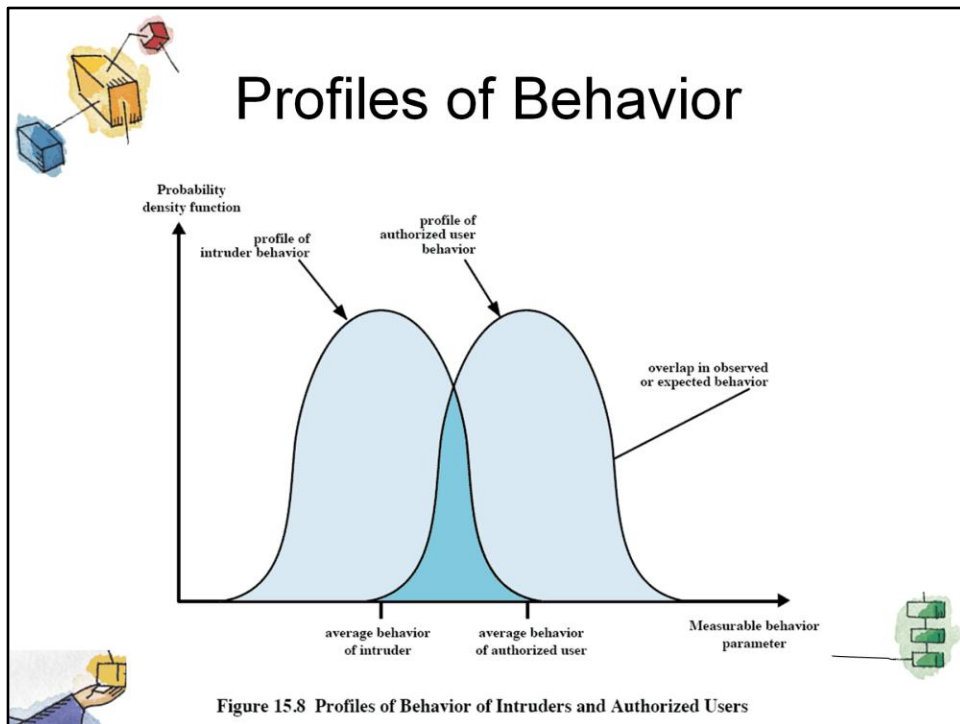
- Sensors are responsible for collecting data.
- The input for a sensor may be any part of a system that could contain evidence of an intrusion.
- Types of input to a sensor include network packets, log files, and system call traces.
- Sensors collect and forward this information to the analyzer.

## **Analyzers:**

- Analyzers receive input from one or more sensors or from other analyzers.
- The analyzer is responsible for determining if an intrusion has occurred.
- The output may include evidence supporting the conclusion that an intrusion occurred.
- The analyzer may provide guidance about what actions to take as a result of the intrusion.

## **User interface:**

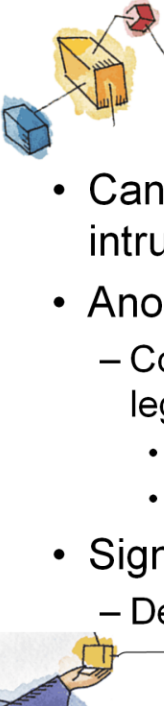
- The user interface to an IDS enables a user to view output from the system or control the behavior of the system.
- In some systems, the user interface may equate to a manager, director, or console component.



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.
- But, 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.



# Host-Based IDSs

- Can detect both external and internal intrusions
- Anomaly detection
  - Collection of data relating to behavior of legitimated users over time may use
    - Threshold detection
    - Profile based detection
- Signature detection
  - Define set of rules or attack patters



Host-based IDSs add a specialized layer of security software to vulnerable or sensitive systems;

The host-based IDS monitors activity on the system in a variety of ways to detect suspicious behaviour.

The primary benefit of a host-based IDS is that it can detect both external and internal intrusions, something that is not possible either with network-based IDSs or firewalls.

### Anomaly detection:

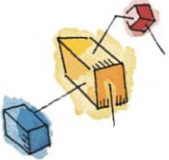
- Involves the collection of data relating to the behavior of legitimate users over a period of time.
- Then statistical tests are applied to observed behavior to determine with a high level of confidence whether that behavior is not legitimate user behavior.
- Two approaches:
  - Threshold detection:** 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.

### Signature detection:

- Involves an attempt to define a set of rules or attack patterns that can be used to decide that a given behavior is that of an intruder.


In essence,

- anomaly approaches attempt to define normal behavior,
- signature-based approaches attempt to define improper behavior.



# Audit Records

- Native audit records
  - Uses the OS accounting software/logs
- Detection-specific audit records
  - Generate audit records required by the IDS



Some record of ongoing activity by users must be maintained as input to an IDS.

### **Native audit records:**

- Virtually all multiuser operating systems include accounting software that collects information on user activity.
- No additional collection software is needed.
- BUT 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 IDS.
- Can be made vendor independent and ported to a variety of systems.
- BUT there is extra overhead involved in having, in effect, two accounting packages running on a machine.

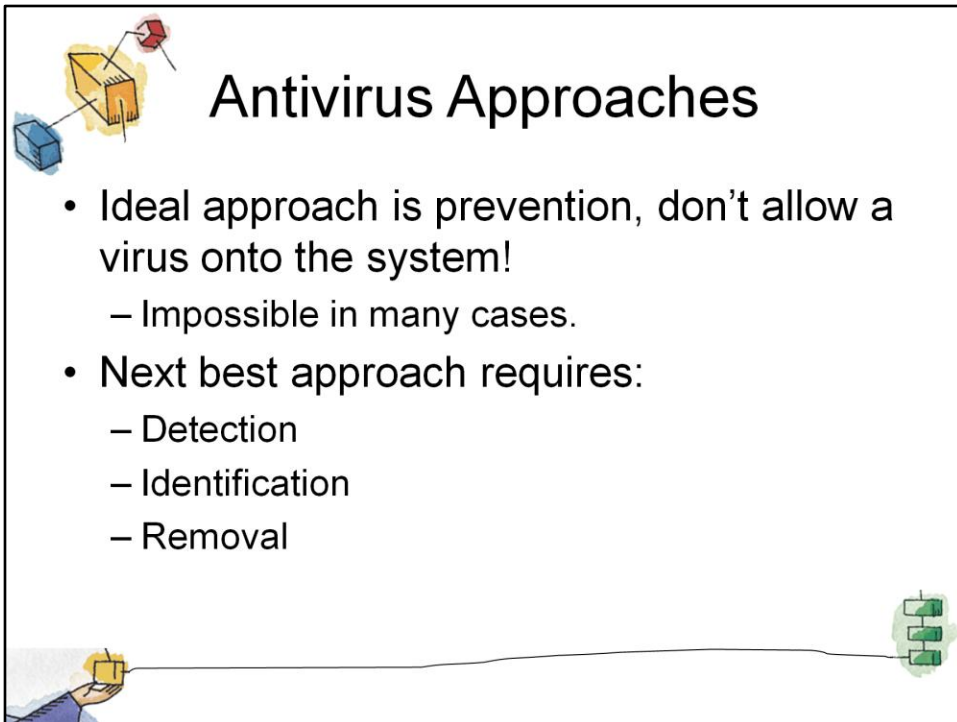




# Roadmap

- Authentication
- Access Control
- Intrusion Detection
- Malware Defense
- Dealing With Buffer Overflow Attacks
- Windows Vista Security



The slide is titled "Antivirus Approaches" and is enclosed in a black rectangular border. In the top-left corner, there is a small illustration of a yellow computer tower with a red virus icon and a blue box. In the bottom-left corner, there is an illustration of a hand holding a yellow computer mouse. In the bottom-right corner, there is a small green icon of a server rack. The main content of the slide is a bulleted list of antivirus approaches.

## Antivirus Approaches

- Ideal approach is prevention, don't allow a virus onto the system!
  - Impossible in many cases.
- Next best approach requires:
  - Detection
  - Identification
  - Removal

The ideal solution to the threat of viruses is prevention:

- Do not allow a virus to get into the system in the first place.

The next best approach is to be able to do the following:

**Detection:**

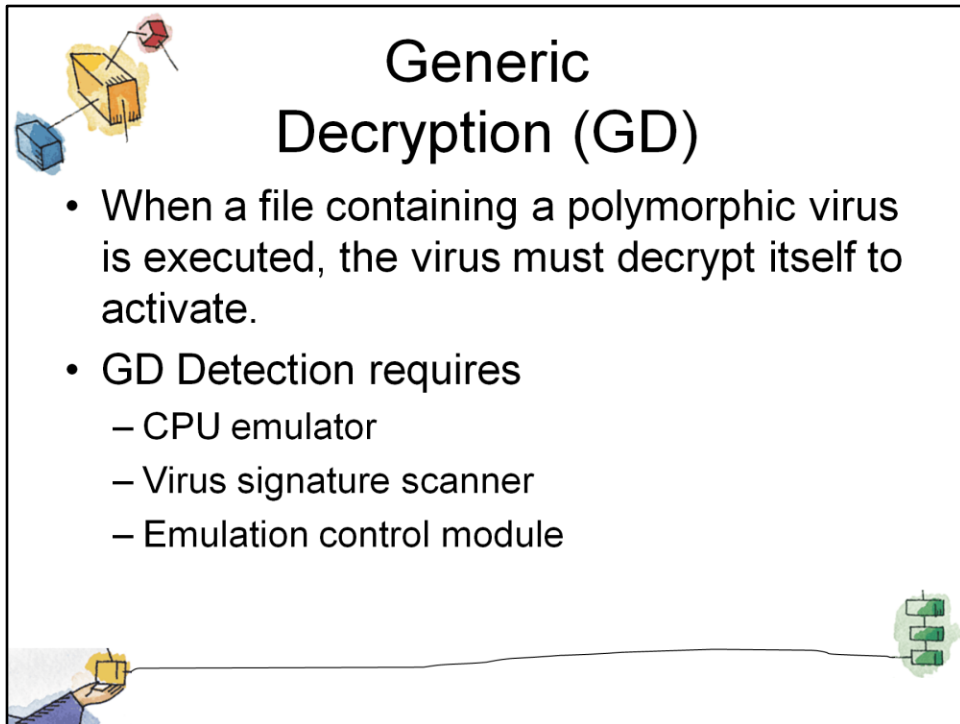
- Once the infection has occurred, determine that it has occurred and locate the virus.

**Identification:**

- Identify the specific virus that has infected a program.

**Removal:**

- Remove all traces of the virus from the infected program and restore it to its original state.
- Remove the virus from all infected systems so that the disease cannot spread further.



## Generic Decryption (GD)

- When a file containing a polymorphic virus is executed, the virus must decrypt itself to activate.
- GD Detection requires
  - CPU emulator
  - Virus signature scanner
  - Emulation control module

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.

To detect such a structure, executable files are run through a GD scanner, which contains the following elements:

**CPU emulator:**

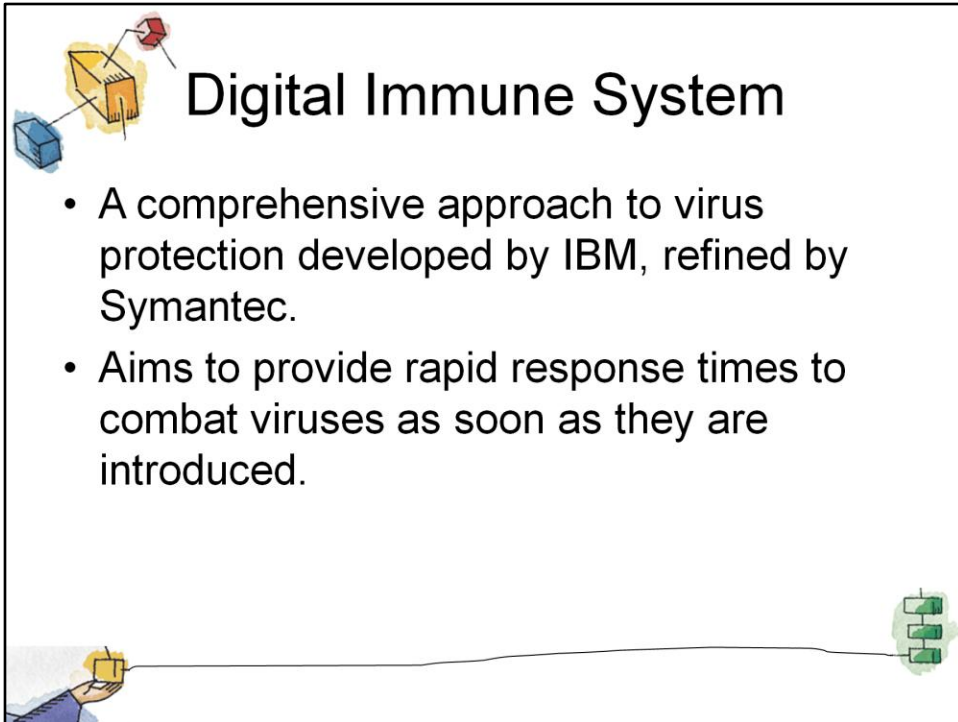
- A software-based virtual computer.
- Instructions in an executable file are interpreted by the emulator rather than executed on the underlying processor.
- The emulator includes software versions of all registers and other processor hardware, so that the underlying processor is unaffected by programs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

The diagram is enclosed in a black rectangular border. In the top-left corner, there are several colorful icons: a blue cube, a yellow cube with a red dot on top, and a red cube. In the top-right corner, the title "Digital Immune System" is written in a large, black, sans-serif font. Below the title, there are two bullet points. In the bottom-left corner, a hand is shown holding a yellow cube. In the bottom-right corner, there is a green icon consisting of three stacked rectangular blocks. A thin black line runs horizontally across the bottom of the diagram, connecting the hand on the left to the green icon on the right.

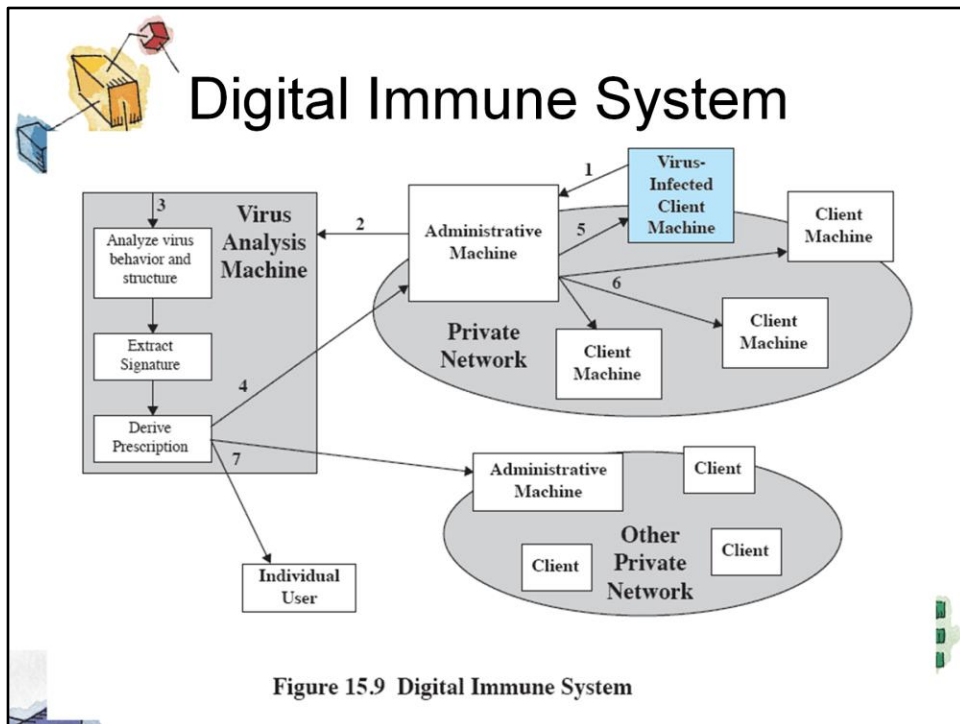
## Digital Immune System

- A comprehensive approach to virus protection developed by IBM, refined by Symantec.
- Aims to provide rapid response times to combat viruses as soon as they are introduced.


This system expands on the use of program emulation discussed in the preceding subsection and provides a general-purpose emulation and virus-detection 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.





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 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.



# Behaviour Blocking Software

- Integrates with the operating system
  - monitors program behavior in real time for malicious actions and blocks them.
- Monitored behaviors may include:
  - opening or modifying certain files
  - formatting disk drives
  - Modifications to executable files or macros
  - Modification of critical system settings
  - Network communication

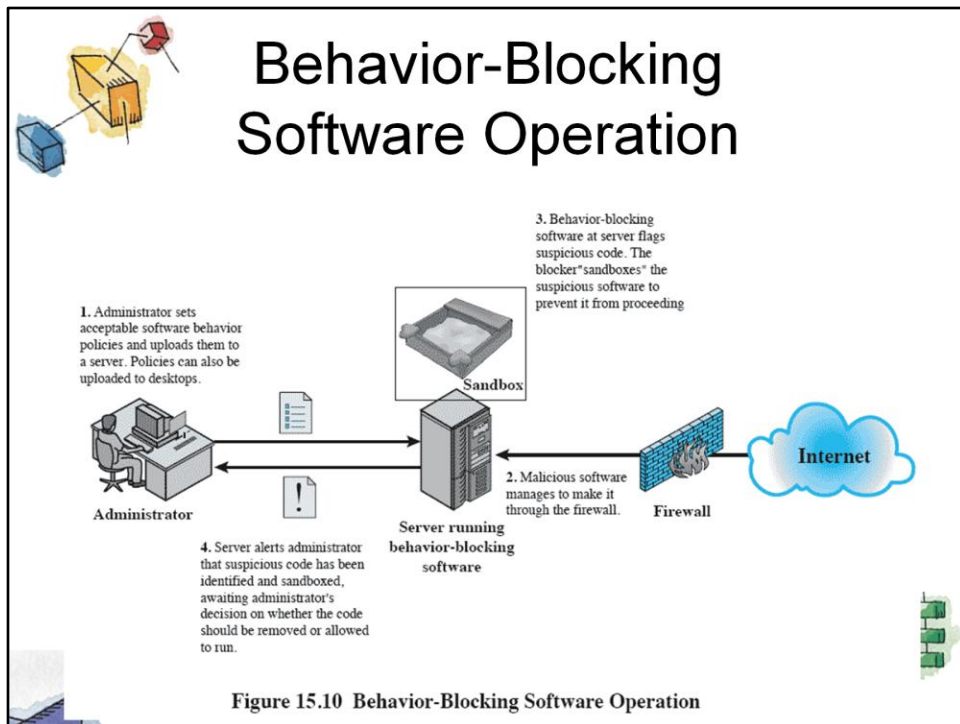


Behavior blocking software integrates with the operating system of a host computer and monitors program behavior in real time for malicious actions.

The behavior-blocking software then 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;
- Initiation of network communications.




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.



# Worm Countermeasures

- a. Signature-based worm scan filters
- b. Filter-based worm containment
- c. Payload-classification-based worm containment
- d. Threshold random walk (TRW) scan detection
- e. Rate limiting
- f. Rate halting

**A. Signature-based worm scan filtering:**

- This type of approach generates a worm signature, which is then used to prevent worm scans from entering/leaving a network/host.
- Typically, this approach involves identifying suspicious flows and generating a worm signature.
- This approach is vulnerable to the use of polymorphic worms

**B. Filter-based worm containment:**

- The filter checks a message to determine if it contains worm code.
- This approach can be quite effective but requires efficient detection algorithms and rapid alert dissemination.

**C. Payload-classification-based worm containment:**

- These network-based techniques examine packets to see if they contain a worm. Various anomaly detection techniques can be used, but care is needed to avoid high levels of false positives or negatives.
- This approach does not generate signatures based on byte patterns but rather looks for control and data flow structures that suggest an exploit.

**D. Threshold random walk (TRW) scan detection:**

- TRW exploits randomness in picking destinations to connect to as a way of detecting if a scanner is in operation
- Suitable for deployment in high-speed, low-cost network devices.
- It is effective against the common behavior seen in worm scans.

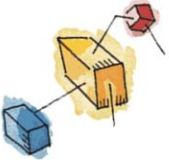
**E. Rate limiting:**

- This class limits the rate of scanlike traffic from an infected host.
- Various strategies can be used, including limiting the number of new machines a host can connect to in a window of time, detecting a high connection failure rate, and limiting the number of unique IP addresses a host can scan in a window of time.
- May introduce longer delays for normal traffic.
- Not suited for slow, stealthy worms that spread slowly to avoid detection based on activity level.

**F. Rate halting:**


- This approach immediately blocks outgoing traffic when a threshold is exceeded either in outgoing connection rate or diversity of connection attempts.
- Must include measures to quickly unblock mistakenly blocked hosts in a transparent way.
- Can integrate with a signature- or filter-based approach so that once a signature or filter is generated, every blocked host can be unblocked.
- Appears to offer a very effective countermeasure.
- Not suitable for slow, stealthy worms.





# Botnet and Rootkit Countermeasures

- IDS and Anti-Viral techniques are useful against bots
  - Main aim is to detect and disable a botnet during its construction
- Rootkits are, by design, difficult to detect
  - Countering rootkits requires a variety of network- and computer-level security tools.



## Bot Countermeasures

A number of the countermeasures discussed in this chapter make sense against bots, including IDSs and digital immune systems. Once bots are activated and an attack is underway, these countermeasures can be used to detect the attack.

But the primary objective is to try to detect and disable the botnet during its construction phase.

## Rootkit Countermeasures

Rootkits can be extraordinarily difficult to detect and neutralize,

- Especially kernel-level rootkits.

Many of the administrative tools that could be used to detect a rootkit or its traces can be compromised by the rootkit precisely so that it is undetectable.

Countering rootkits requires a variety of network- and computer-level security tools.


- Both network-based and host-based intrusion detection systems can look for the code signatures of known rootkit attacks in incoming traffic.
- Host-based antivirus software can also be used to recognize the known signatures.



# Roadmap


- Authentication
- Access Control
- Intrusion Detection
- Malware Defense
- Dealing With Buffer Overflow Attacks
- Windows Vista Security





# Buffer Overflow

- Protection from stack buffer overflows can be broadly classified into two categories:
- **Compile-time defenses**
  - Aims to harden programs to resist attacks in new programs
- **Stack protection mechanisms**
  - Aims to detect and abort attacks in existing programs



Stack buffer overflows can be broadly classified into two categories:

### **Compile-time defenses,**


- Aims to harden programs to resist attacks in new programs

### **Run-time defenses,**

- which aim to detect and abort attacks in existing programs



While suitable defenses have been known for a couple of decades, the very large existing base of vulnerable software and systems hinders their deployment;

- hence the interest in run-time defenses, which can be deployed in operating systems and updates and can provide some protection for existing vulnerable programs.



# Compile Time Defenses


- Choice of Programming Language
  - Some languages do not allow some unsafe coding practices
- Safe Coding Techniques and Auditing
- Language Extensions and Use of Safe Libraries
- Stack Protection Mechanisms



The programmer needs to inspect the code and rewrite any unsafe coding constructs in a safe manner.



Among other technology changes, programmers have begun to undertake extensive audits

There have been a number of proposals to augment compilers to automatically insert range checks on such references.



# Run Time Defenses

- These defenses involve changes to the memory management of the virtual address space of processes.
  - Executable address space protection
  - Address space randomization
  - Guard pages



These defenses involve changes to the memory management of the virtual address space of processes.

These changes act to either alter the properties of regions of memory, or to make predicting the location of targeted buffers sufficiently difficult to thwart many types of attacks.

### **Executable Address Space Protection**

- Many of the buffer overflow attacks involve copying machine code into the targeted buffer and then transferring execution to it.
- 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.

### **Address Space Randomization**

- This involves manipulation of the location of key data structures in a processes address space.
- In order to implement the classic stack overflow attack, the attacker needs to be able to predict the approximate location of the targeted buffer.
- The attacker uses this predicted address to determine a suitable return address to use in the attack to transfer control to the shellcode.
- One technique to greatly increase the difficulty of this prediction is to change the address at which the stack is located in a random manner for each process.

### **Guard Pages**

- This exploits the fact that a process has much more virtual memory available than it typically needs.
- 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.
- This can prevent buffer overflow attacks, typically of global data, which attempt to overwrite adjacent regions in the processes address space.

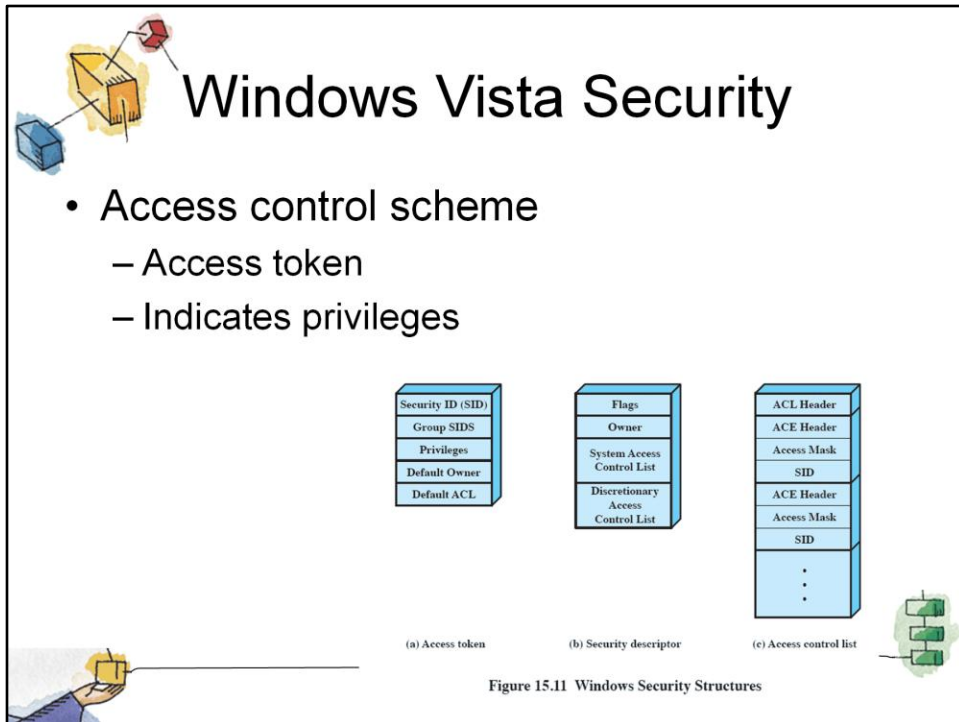


# Roadmap

- Authentication
- Access Control
- Intrusion Detection
- Malware Defense
- Dealing With Buffer Overflow Attacks

→ Windows Vista Security





The access token, include a security ID (SID), which is the identifier by which this user is known to the system for purposes of security.

The access token serves two purposes:

1. It keeps all necessary security information together to speed access validation.
  - When any process associated with a user attempts access, the security subsystem can make use of the token associated with that process to determine the user's access privileges.
2. It allows each process to modify its security characteristics in limited ways without affecting other processes running on behalf of the user.

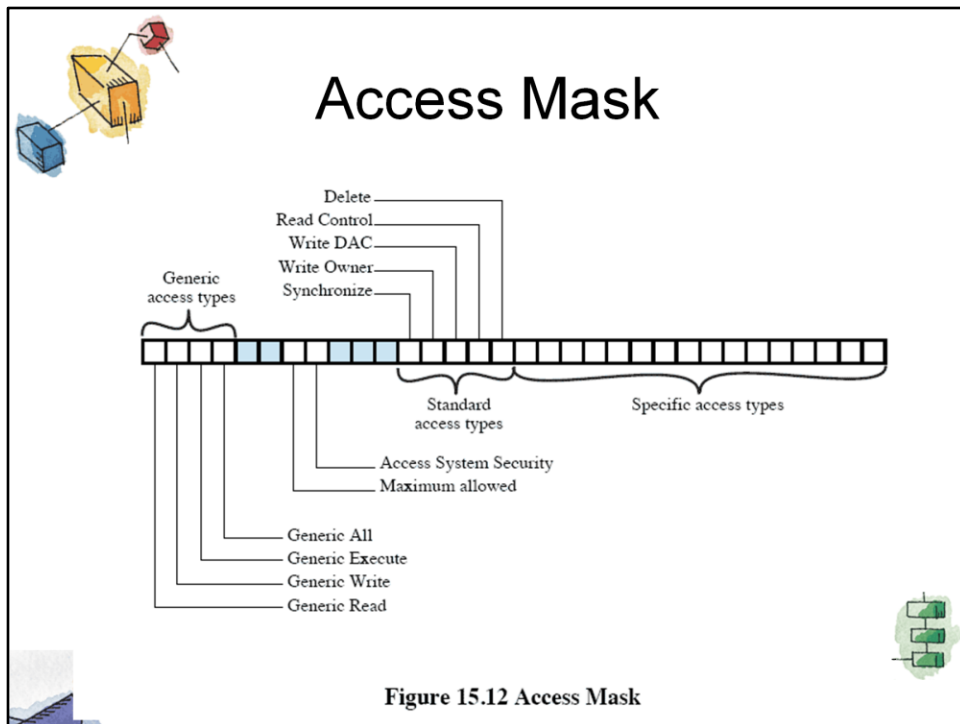


Figure 15.12 Access Mask

This figure shows the contents of the access mask.

The least significant 16 bits specify access rights that apply to a particular type of object.

- E.g. bit 0 for a file object is `File_Read_Data` access and bit 0 for an event object is `Event_Query_Status` access.

The most significant 16 bits of the mask contains bits that apply to all types of objects.

**Synchronize:** Gives permission to synchronize execution with some event associated with this object.

- In particular, this object can be used in a wait function.

**Write\_owner:** Allows a program to modify the owner of the object.

- This is useful because the owner of an object can always change the protection on the object (the owner may not be denied Write DAC access).

**Write\_DAC:** Allows the application to modify the DACL and hence the protection on this object.

**Read\_control:** Allows the application to query the owner and DACL fields of the security descriptor of this object.

**Delete:** Allows the application to delete this object.

The high-order half of the access mask also contains the four generic access types.

- These bits provide a convenient way to set specific access types in a number of different object types.