

Protection & Security

November 15, 2017

Protection in Computer Systems

- ❏ OS consists of and manages a collection of objects
 - ❏ Types: Hardware, Software
 - ❏ Examples: memory; a file or data set on an auxiliary storage device; an executing program in memory; a directory of files; a hardware device; a data structure, such as a stack; a table of the operating system; instructions, especially privileged instructions; passwords; the protection mechanism itself
 - ❏ Each object has a unique name and can be accessed through a well-defined set of operations

Protection in Computer Systems

- ❏ Goal of protection: To ensure that each object is accessed correctly and only by those processes that are allowed to do so.
- ❏ Guiding principle: **Principle of Least Privilege**
 - ❏ Programs, users and systems should be given just enough privileges to perform their tasks.

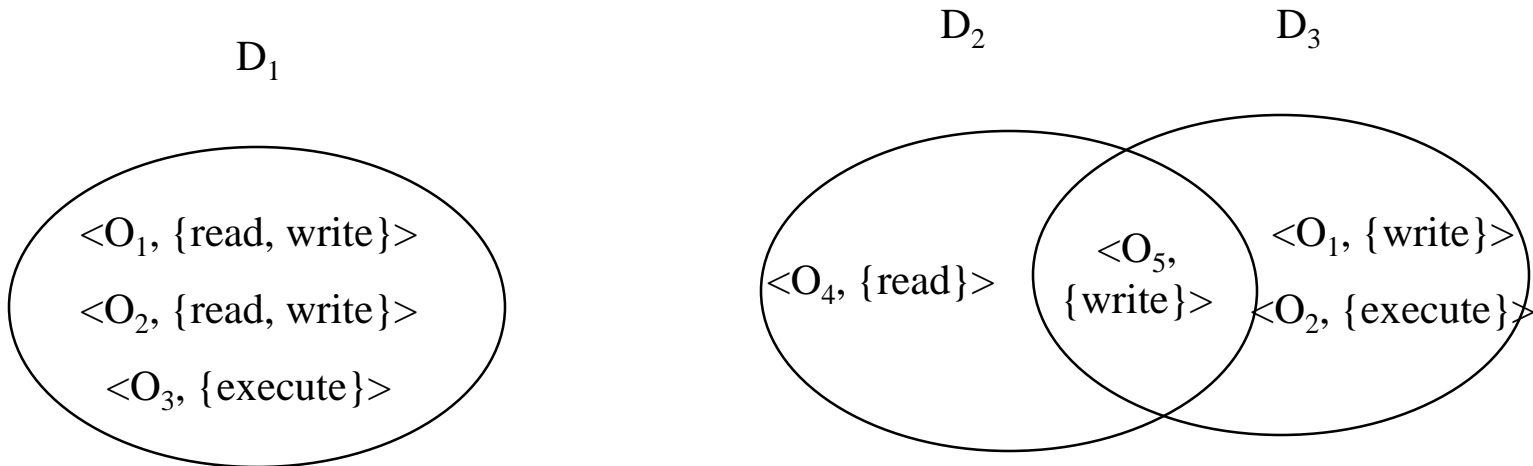
Domain of Protection

Access-right = $\langle \text{object-name, rights-set} \rangle$

Example: $\langle \text{File1, \{read, write\}} \rangle$

Domain: a set of access-rights

Specifying the resources that a process may access and in which manner the resources are accessed.



Domain of Protection: An Example

Dual-mode model of OS execution

User mode

A process running in the user mode can invoke only non-privileged instructions and access the process's own memory space, and cannot directly access shared hardware devices

Kernel mode

A process running in the kernel (i.e., monitor) mode can execute privileged instructions and access the whole memory space and all kinds of hardware devices

Access Matrix

- ❏ A general tool to represent the domains of protection
- ❏ Rows represent domains (associated with subjects)
- ❏ Columns represent objects
- ❏ Access (i,j) is the set of operations that a process executing in Domain_i can invoke on Object_j .

object domain	F_1	F_2	F_3	printer
D_1	read		read	
D_2				print
D_3		read	execute	
D_4	read write		read write	

Access Matrix: Advantage

- ❏ Separating mechanism from policy

 - ❏ Mechanism

 - ❏ OS provides access matrix and rules

 - ❏ Ensuring that the matrix is only manipulated by authorized agents and that rules are strictly enforced

 - ❏ Policy

 - ❏ (Authorized, privileged) User dictates policy

 - ❏ The policy specifies who can access what object and in what mode

Implementation of Access Matrix




- ❏ Access matrix is sparse; keeping the entire matrix is not efficient
- ❏ Different implementations
 - ❏ Capability-based system (stored by rows)
 - ❏ Access control list (stored by columns)

Capability-based Approach

- Keep the capability for each domain
 - Capability: a list of objects and access rights allowed by the domain to the objects in the list (o, access [d, o])
 - Ex: $\langle o1, \{\text{read, write, execute}\} \rangle$
- Major advantages:
 - Given a subject id and its associated domain, authorization can be done fast

Capability-based Approach

Disadvantages:




-  Slow review: Take time to find out all domains (subjects) with an access right to a particular object
-  Difficult to revoke privileges
-  Garbage collection: Garbage --- object that no domain (subjects) have the capability to access

Access Control List Approach

- ❏ Each object is assigned a list of pairs $(d, access[o, d])$ for all domains d that are allowed to access the object
- ❏ Advantages:
 - ❏ Revocation of access rights to an object from a domain is simple and fast
 - ❏ Easy to review (know domains that can access this object)
- ❏ Disadvantage:
 - ❏ Authorization can be slow since the access control list needs to be searched for every access of an object

Access Control List Approach

Solution

-  During the first access of an object, the subject **caches** all the rights to access this object
-  The cache entry is used for subsequent access to the same object
-  **Problem:** When modifying the access right of the object in the access control list, the modification does not take an immediate effect unless some mechanism is provided to validate the cache entries

Problems common to both approaches

- ❏ Capability or access control list can be huge due to a large number of domains, objects, and access rights
- ❏ Solution: Use **protection group**
 - ❏ Group multiple domains into a group
 - ❏ Access control list maintains access right for each group
 - ❏ Capability per group instead of per subject
 - ❏ Drawback: domains in the same group have the same access rights – may not be fine-grained enough

Protection and Security

- ❏ Protection (internal):
 - ❏ How do we provide controlled access to programs and data stored in a computer system?
 - ❏ Limitation: what if user authentication is compromised (thus an attacker can pretend as an authorized user)?
- ❏ Security (internal + external):
 - ❏ Resources are used and accessed as intended under all circumstances.

Security Violations

- ❏ Common (accidental and malicious) security violations
 - ❏ Breach of confidentiality: unauthorized reading of data.
 - ❏ Breach of integrity: unauthorized modification of data.
 - ❏ Breach of availability: unauthorized destruction of data.
 - ❏ Theft of service: unauthorized use of resources.
 - ❏ Denial of service: preventing legitimate use of the system.

Security

❏ Security must occur at four levels to be effective:

❏ Physical

❏ Human

❏ Avoid social engineering, phishing, dumpster diving

❏ Operating System

❏ Network

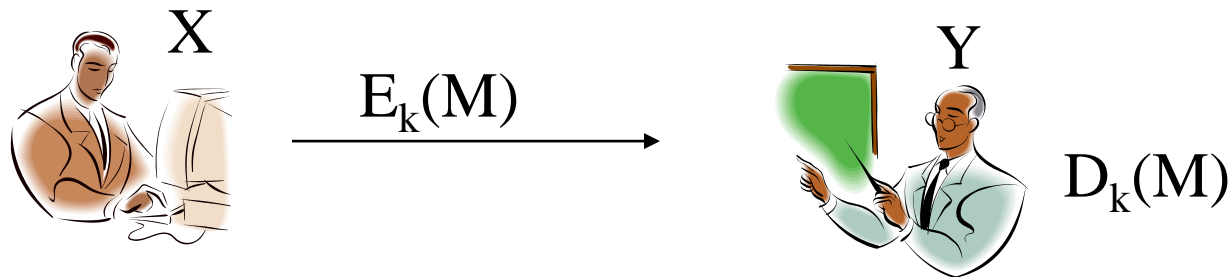
❏ Security is as weak as the weakest chain

Confidentiality: Encryption & Decryption

- ❏ Two types of encryption/decryption systems
 - ❏ Symmetric key systems (also called secret key systems)
 - ❏ Use same key for encryption and decryption
 - ❏ Key is kept secret
 - ❏ Asymmetric key systems (also called public key systems)
 - ❏ Use different keys for encryption and decryption
 - ❏ Key for encryption is known (public key), but key for decryption is kept secret

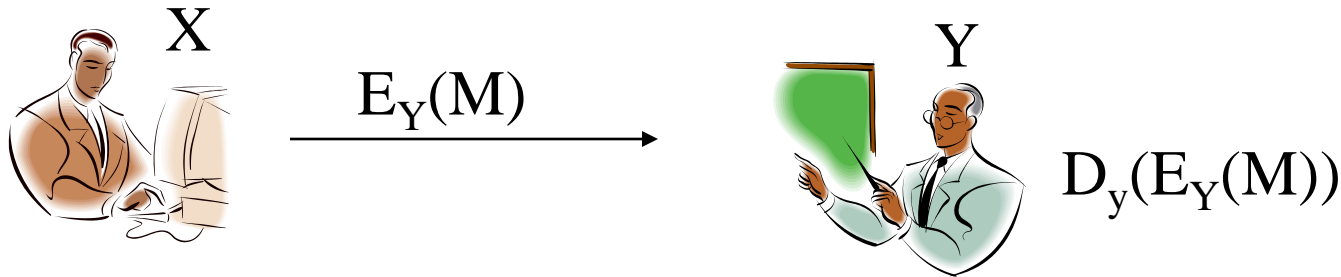
Symmetric Key Systems: DES, AES

- Based on
 - permutation: Permute bits of a word to provide diffusion
 - substitution: To provide confusion
- Use a long key, making it computationally expensive to break
- Drawback:** Need to distribute a secret key securely over an insecure communication network before secure communication (key distribution problem)



Asymmetric Key System: RSA




- Encryption procedure E (and the associated key – public key) is placed in the public domain
- Decryption procedure D is also placed in the public domain
- The key used in D (called private key) is kept in secret






- Main idea: Rely on the fact that it is impractical to derive the private key from the knowledge of the encryption/decryption procedures and the public key

Authentication

Password-based authentication

-  OS stores passwords for users (in encrypted form)
-  User is prompted to enter password
-  Encrypted version of the entered password is compared to the recorded version: authentication succeeds only if match!

Public key based authentication




-  OS keeps public keys for users
-  User is requested to encrypt a random message m provided by the OS into m' with its private key
-  m' is decrypted by OS with recorded public key: authentication succeeds only if m is obtained from decryption.

Integrity and Authenticity: Digital Signatures

- ❏ Digital signature is a way to code an electronic message such that the recipient of the message can certify which sender actually sent the message
- ❏ Basic idea (based on public key system)
 - ❏ Sending user encrypts a message M using his secret key (signing)
 - ❏ Receiving user decrypts $E(M)$ using the sender's public key (verifying)
 - ❏ If the receiver cannot decrypt the message, the sender is not who he claims to be

Program Threats: caused by programs inappropriately developed/downloaded


Trojan Horse

-  Program or code segment appears as useful and not harmful
-  Contains code segment that misuses its environment
-  Example: Faked programs, Spyware, covert channels


Trap Door

-  Specific user identifier or password that circumvents normal security procedures

Logic Bomb

-  Program that initiates a security incident under certain circumstances

Stack and Buffer Overflow

-  Exploiting a bug in a program (overflow either the stack or memory buffers)

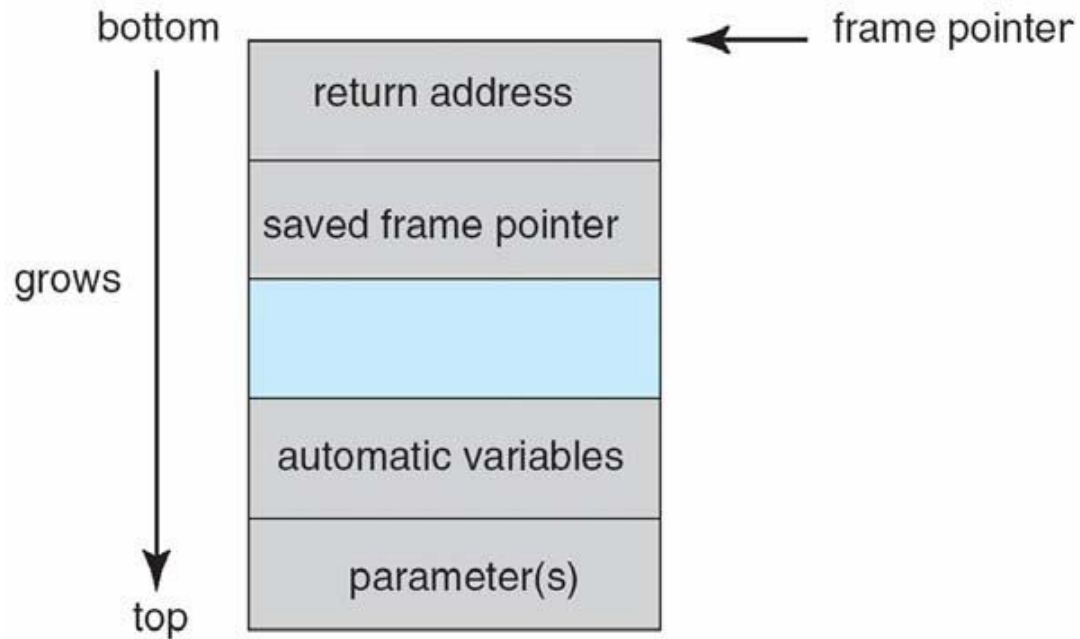
C Program with Buffer-overflow Condition

```
#include <stdio.h>
#define BUFFER_SIZE 256

int main( int argc, char *argv[])
{
    char buffer[BUFFER_SIZE];
    if(argc<2)
        return -1;
    else{
        strcpy(buffer, argv[1]);
        return 0;
    }
}
```

What if the argument is a string with length more than 256 bytes?

Layout of Typical Stack Frame



Attack Code

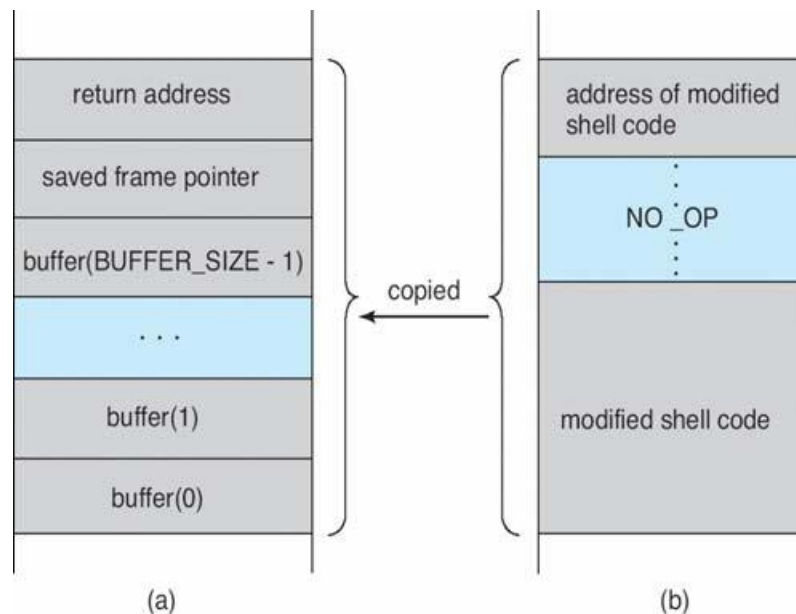
```
#include <stdio.h>

int attack_code( int argc, char *argv[ ])
{
    execvp(“\bin\sh”, “\bin\sh”, NULL);
    return 0;
}
```

The above C source code is compiled ➡ get object code.

Stack Frame

Launch attack: Compose argument to the main function such that the starting address of attack code is put into the “return address” field.

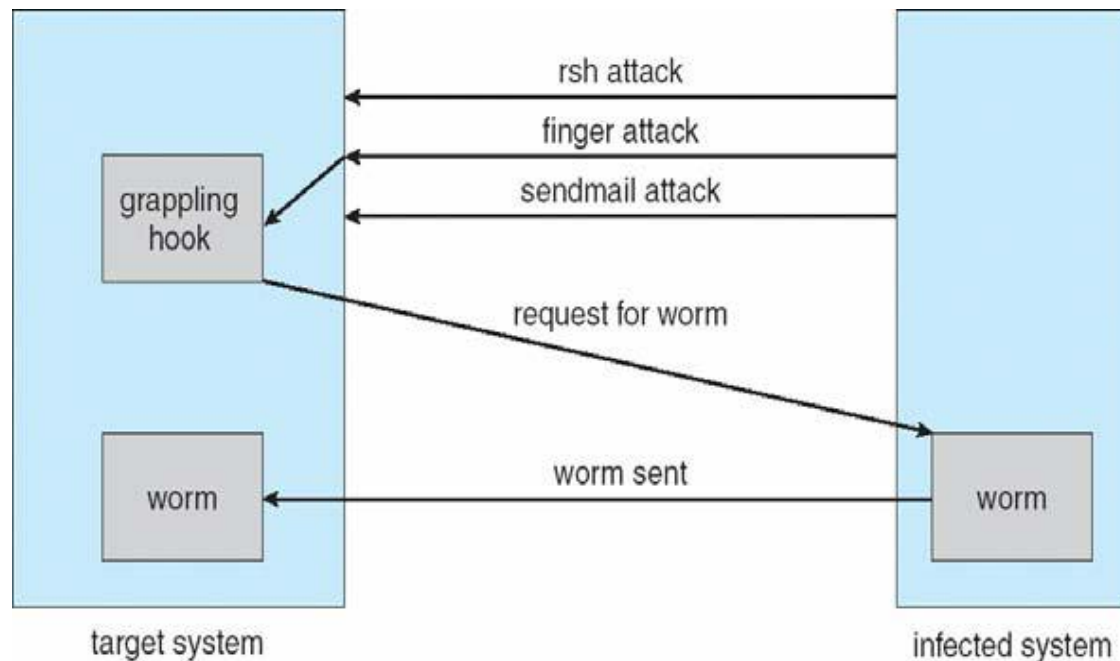


Before attack

After attack



System and Network Threats

- Worms: using spawn mechanism; standalone program
- Morris worm
 - Exploiting known vulnerabilities in Unix sendmail, finger, and rsh/rexec, as well as weak passwords.
 - Grappling hook program uploading main work program





System and Network Threats

Port scanning

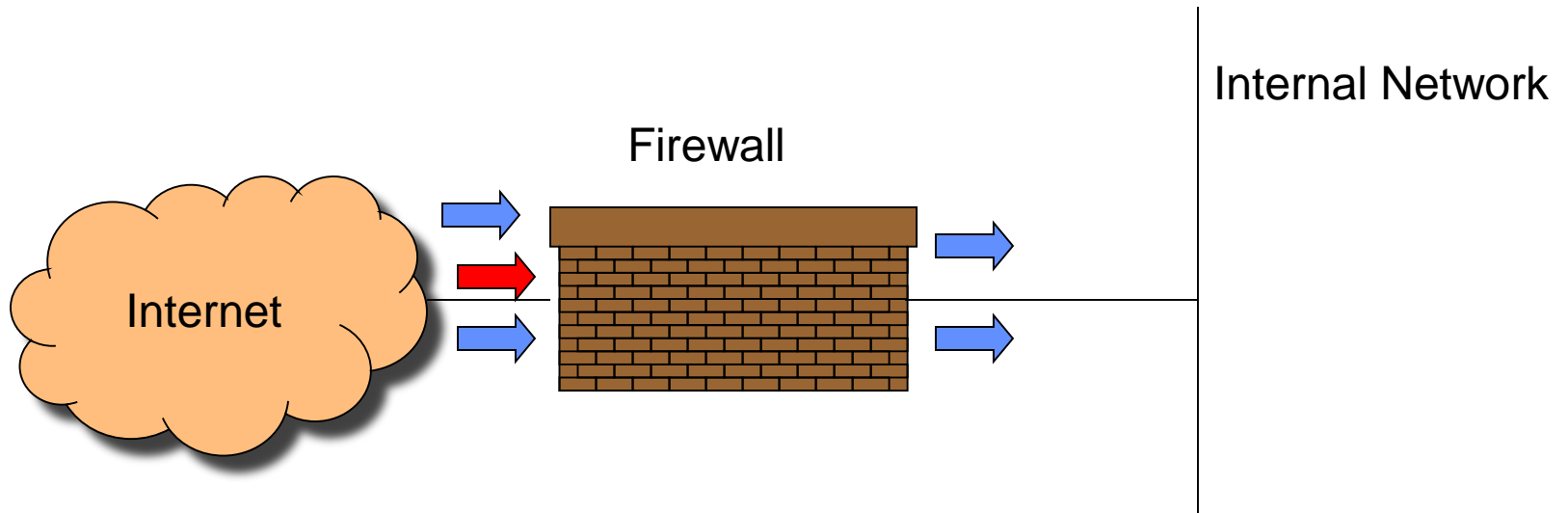
-  Automated attempt to connect to a range of ports on one or a range of IP addresses
-  Intention: connect to a machine and exploit its vulnerability

Denial of service

-  Overload the targeted computer preventing it from doing any useful work
-  Distributed denial-of-service (DDOS) coming from multiple sites at once

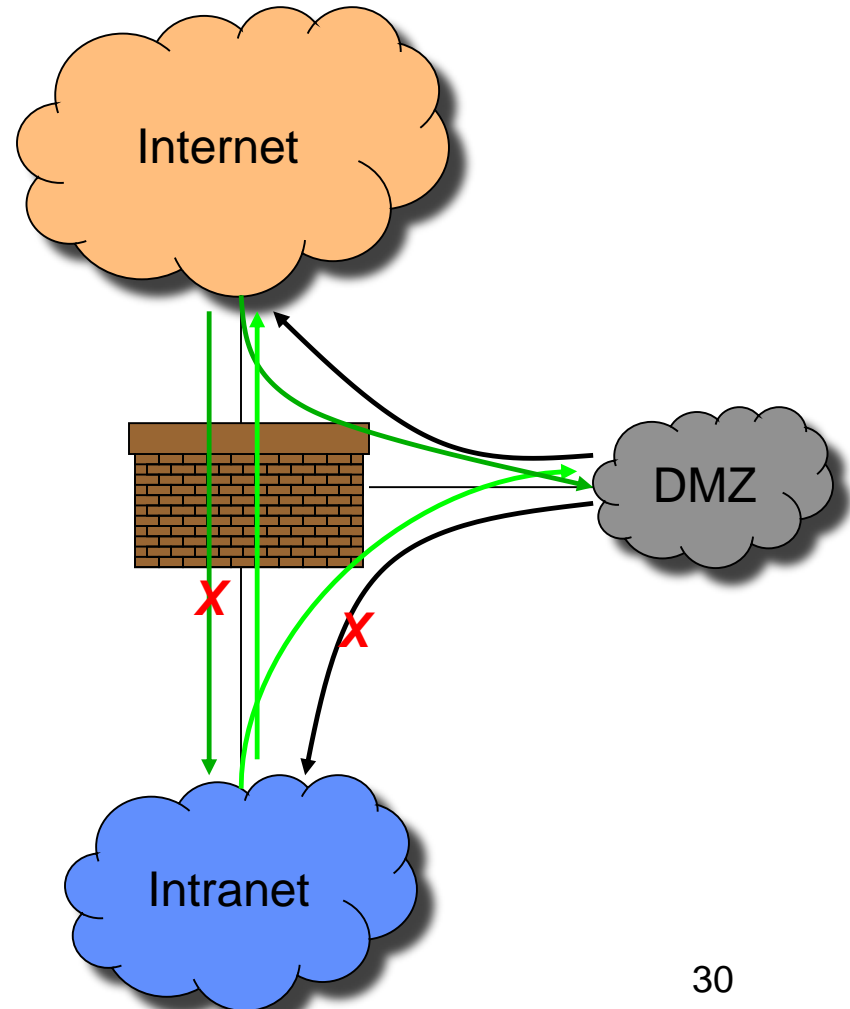
Firewalls

- ❏ Firewall inspects traffic through it
- ❏ Allows traffic specified in the policy
- ❏ Drops everything else



Typical Firewall Configuration

- Internal hosts can access (demilitarized zone) DMZ and Internet
- External hosts can access DMZ only, not Intranet
- DMZ hosts can access Internet only
- Advantages?
 - If a service gets compromised in DMZ it cannot affect internal hosts

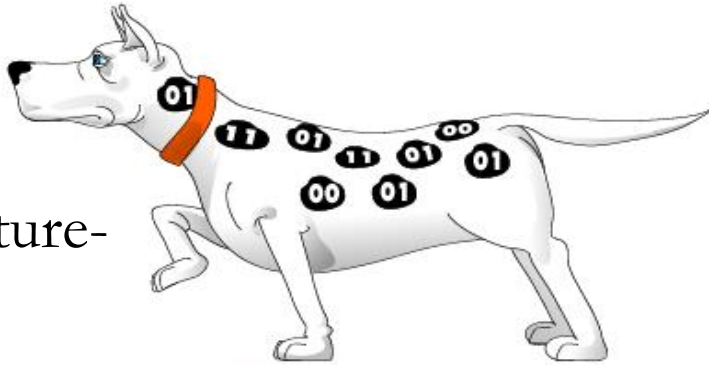


Intrusion Detection Systems

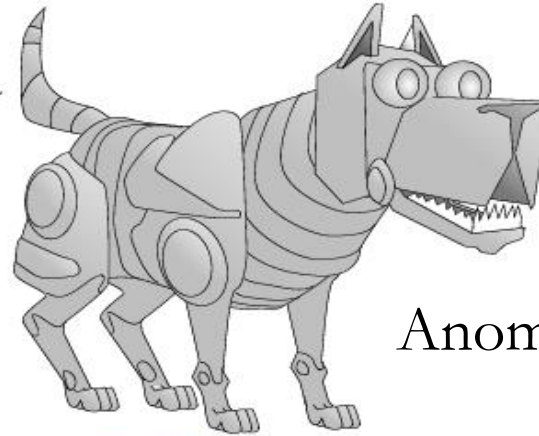
- ❏ Firewalls allow traffic only to/from legitimate hosts and services
- ❏ Traffic to/from the legitimate hosts/services can contain attacks
 - ❏ E.g., Internet worms
- ❏ Solution?
 - ❏ Intrusion Detection Systems
 - ❏ Monitor data and behavior
 - ❏ Report when identify attacks

Types of IDS

Signature-based



Anomaly-based



Host-based














Network-based

Virtual Memory Management

- ❏ How does the demand paging system work?
 - ❏ What is page fault? what should be changed to page table? what is back store?
 - ❏ How is page fault handled?
- ❏ Effective memory access time (EAT) with VM in place
 - ❏ How to compute it?
- ❏ Page replacement algorithms
 - ❏ FIFO; Optimal
 - ❏ LRU (least recently used page to be replaced)
 - ❏ How does it work exactly? Clock-based and stack-based solutions
 - ❏ Approximate solution: second-chance scheme
- ❏ What's the appropriate level of concurrency?
 - ❏ Concept: thrashing

File System Interface

-  Disk structure
 -  Disk; partition; volume; directory
 -  Structures of directory
-  File system mounting
-  File protection
-  Other basic concepts:
 -  File attributes; types
 -  File structures
 -  File basic operations
 -  Synchronization (locking)
 -  Access methods

File System Implementation

- ❏ Storage space allocation
 - ❏ Contiguous (including extent-based FS)
 - ❏ Linked (including FAT)
 - ❏ Indexed (including i-node for UNIX/Linux File systems)
- ❏ Free-space management
 - ❏ Bit vector; Linked list; Grouping; Counting
- ❏ Layered-structure of File management system
- ❏ Basic data structures:
 - ❏ on-disk: boot control block; volume block; directories; FCBs
 - ❏ in-memory: mount table; open-file tables; buffers; caches
- ❏ Basic operations: create; open
- ❏ Other basic concepts: VFS; NFS

I/O Management

- ❏ Interface between host (CPU+Memory) and I/O devices
 - ❏ 4 registers of I/O device controller: in/out/status/control
 - ❏ 3 ways to access the registers: special instruction; load/store; hybrid
 - ❏ 3 protocols for interaction: polling; interrupt; DMA
 - ❏ Blocking/non-blocking/asynchronous I/O operations
- ❏ Basic functions provided by I/O subsystems
 - ❏ Scheduling
 - ❏ Buffering (what are the reasons for buffering?)
 - ❏ Caching; spooling; reservation
- ❏ Other concepts
 - ❏ Kernel data structures
 - ❏ I/O Protection

Protection

- ❏ Concept: Domain of protection
 - ❏ Principle of least privilege
 - ❏ What is DoP?
- ❏ Access matrix
- ❏ Implementations of access matrix
 - ❏ Capability; access control list
 - ❏ How do the two work? Their pros and cons

Security

- ❏ Common security violations
- ❏ How to protect information confidentiality?
 - ❏ Basic concepts of [two encryption/decryption systems](#)
 - ❏ How to use them for confidentiality protection?
 - ❏ How to use encryption/decryption systems for authentication?
- ❏ Program threats
 - ❏ [Buffer overflow attack](#)
 - ❏ Other examples of program threats
- ❏ Network and system threats and protection
 - ❏ Know the terms: worms; port scan; firewall; intrusion detection