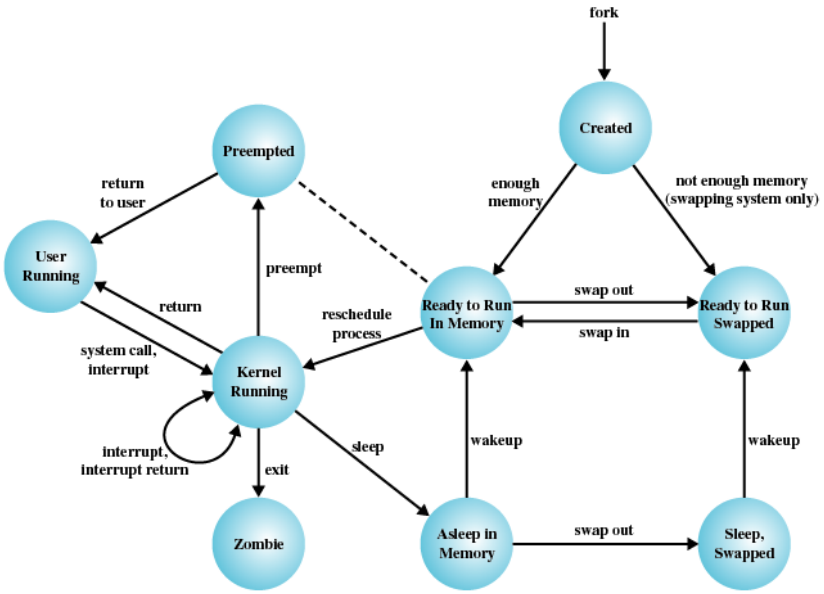
Freie Universität Berlin

TI III: Operating Systems & Computer Networks
Booting, Services, and Security

Prof. Dr.-Ing. Jochen Schiller
Computer Systems & Telematics
Freie Universität Berlin, Germany



Content

1. Introduction and Motivation
2. Subsystems, Interrupts and System Calls
3. Processes
4. Memory
5. Scheduling
6. I/O and File System
7. **Booting, Services, and Security**

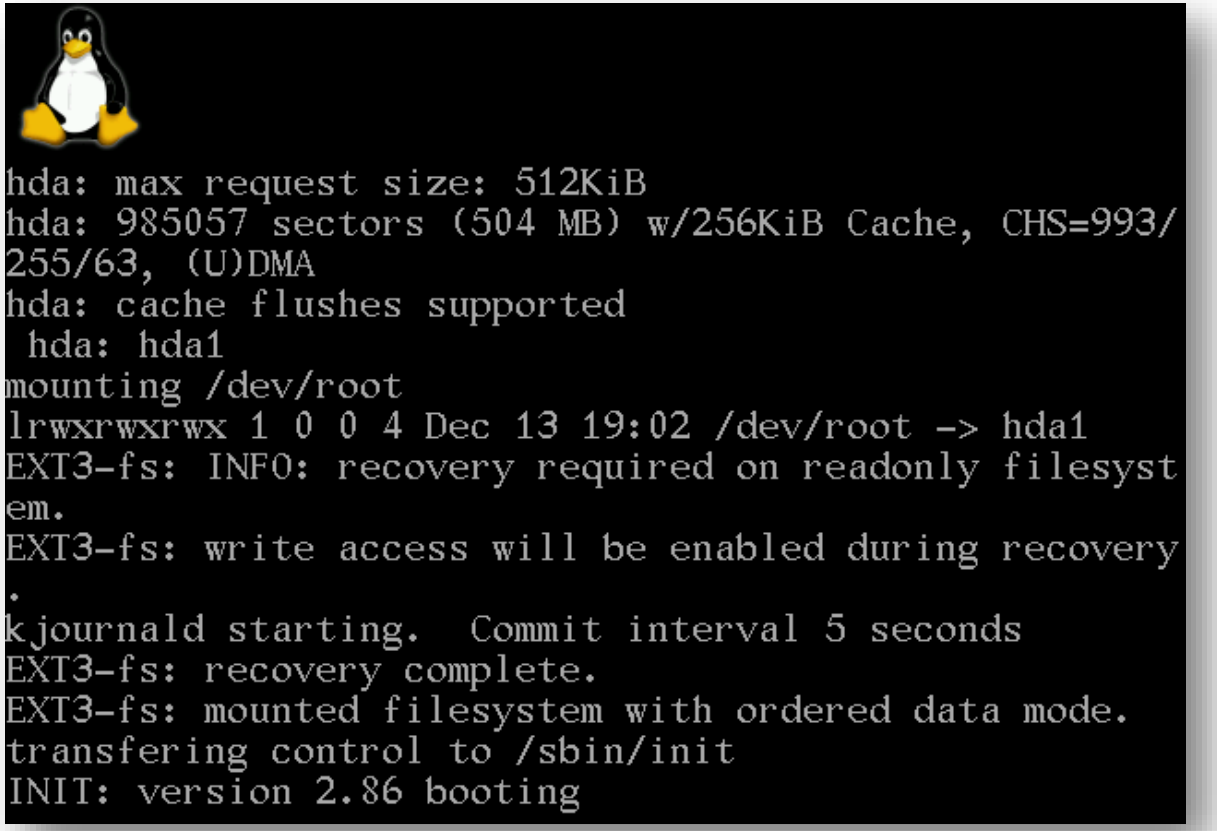
Boot Process Overview

Process of starting an operating system

- “Bootstrapping” – simple system is used to start more complex system

Goals:

- Hardware detection and setup
- Operating system loading and initialization

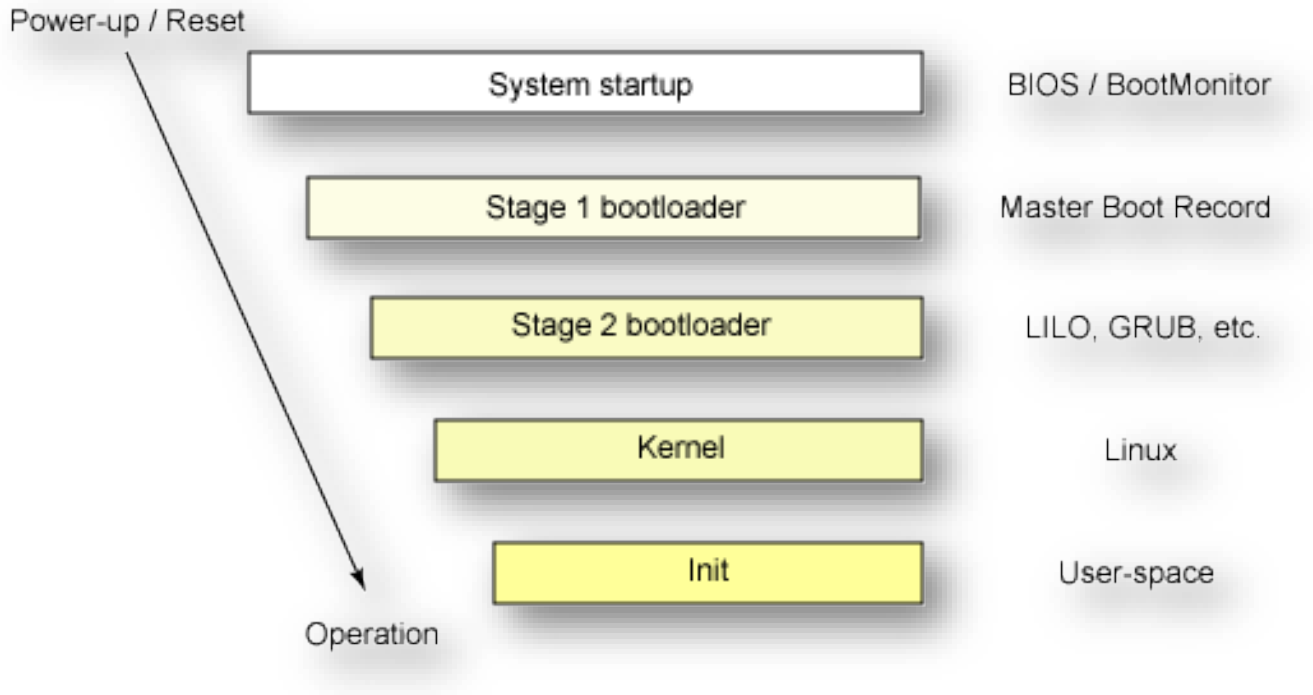


Boot Process Overview

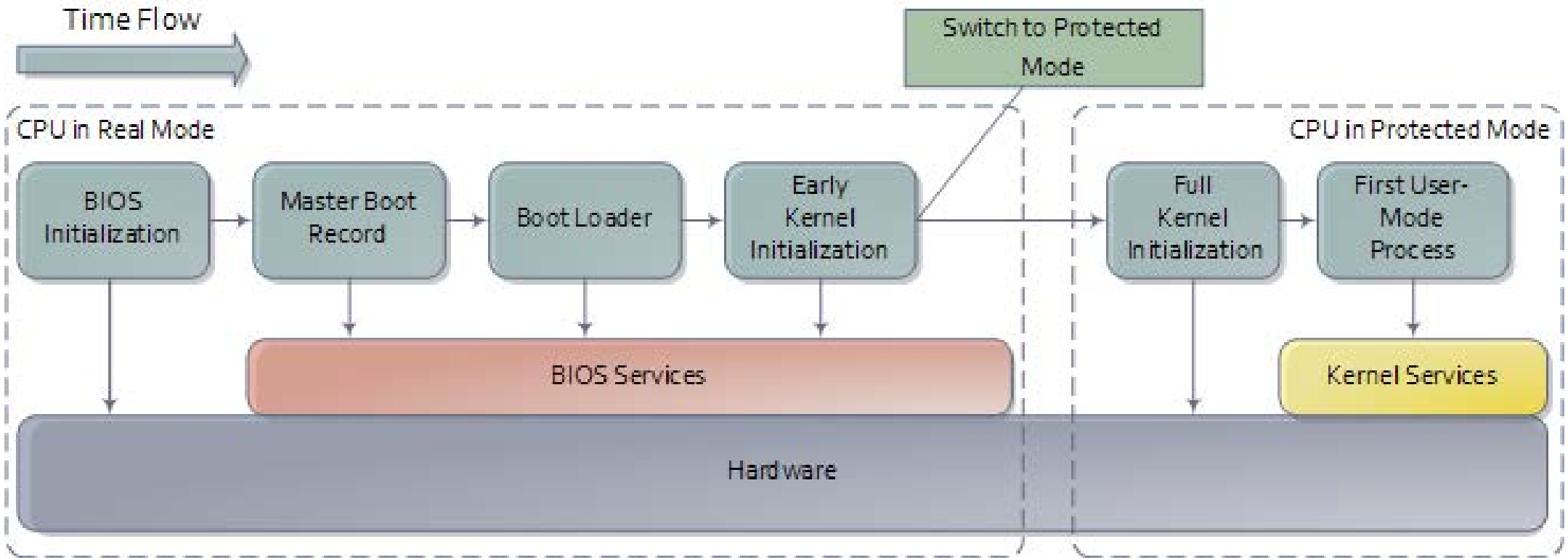
Multi-stage process (early stages without OS involvement)

- System startup
  - Simple hardware initialization performed by BIOS
- Stage 1 bootloader
  - Continue initialization
  - Load state 2 bootloader
- Stage 2 bootloader
  - Continue initialization
  - Load kernel image
- Kernel
  - Continue initialization
  - Start first process “init”
- Init
  - Start system services, including login shell

➢Only limited resources available at each state



Boot Process Overview



http://duartes.org/gustavo/blog/post/how-computers-boot-up

Booting – System Startup (with classical BIOS)

Power supply switched on → Power supply performs self-test
- Sends Power Good signal to CPU if OK

1. CPU Timer chip receives Power Good signal
   - ➢ Stops sending reset signals to CPU
2. CPU executes ROM BIOS code at **`0xFFFF0000`**
3. BIOS performs basic hardware test
   - Beeps on errors
4. BIOS checks for adapters with own ROM BIOS
   - ➢ Initializes video adapter
5. BIOS runs Power On Self Test (POST)
   - No memory check, if “warm-start”
   - Error messages on non-fatal errors
   - Abort and beep code on fatal error
6. BIOS reads system configuration from CMOS
   - Defines in which order to check drives for operating system
7. BIOS examines first sector of bootable medium
   - ➢ Loads Master Boot Record (MBR)

Booting – Stage 1/2 Bootloader

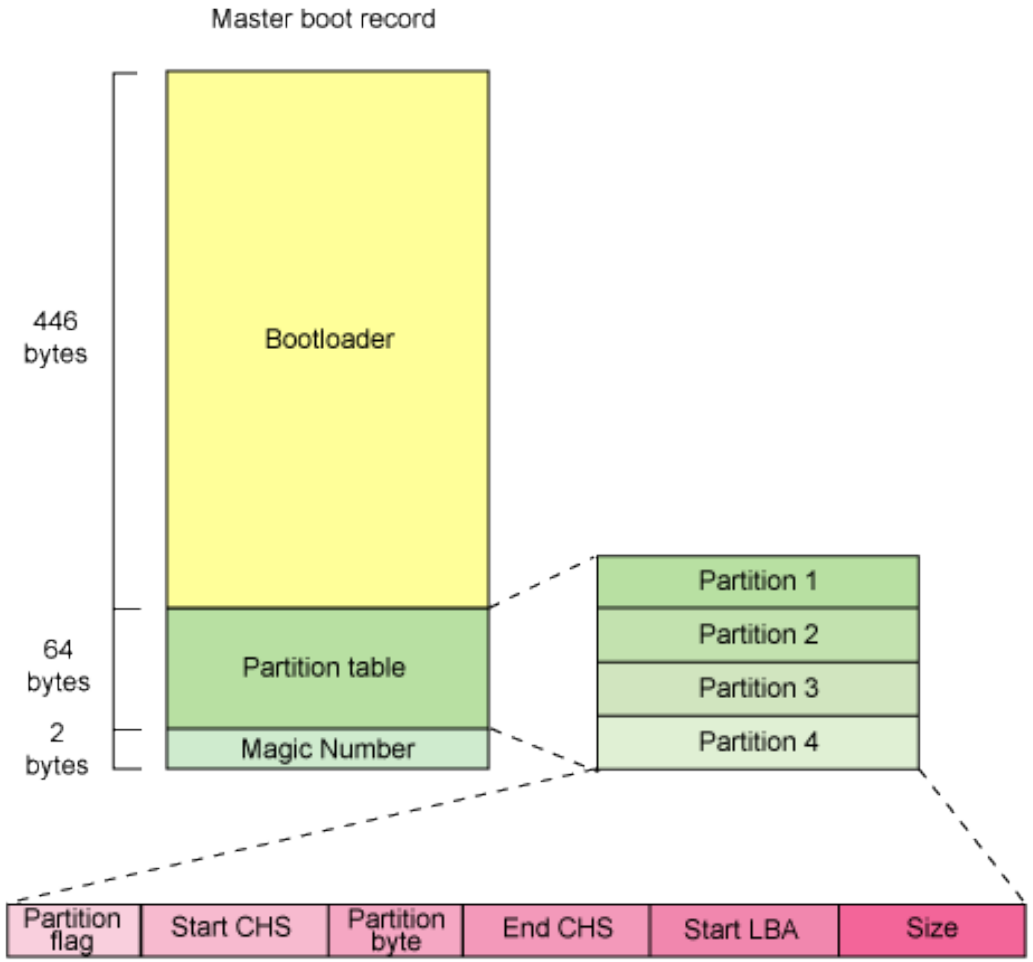
- BIOS transfers control to bootloader
  - Windows: NTLDR
  - Linux: Linux Loader (LILO) or GRand Unified Bootloader (GRUB)

May offer choices for OS or system parameters to user

Boot loader works in stages
- Work around size limitations (only 446 bytes available in MBR)
- Incrementally add support for loading larger binaries from anywhere on disk
- Commonly two, sometimes three stages (“stage 1.5”)

Last stage of bootloader is powerful enough to load kernel image

Chain loading allows bootloader to pass control to other bootloaders, e.g. GRUB to NTLDR



UEFI/EFI

Today, UEFI (Unified Extensible Firmware Interface) and the predecessor EFI (Extensible Firmware Interface) describe the interface between firmware, components of a computer, and the operating system

Includes Secure Boot

- allows booting of signed bootloaders only
- Typically, shim needed for Linux as many mainboard manufacturers only deliver MS signatures

Allows the selection of drivers, exclusion of software components of the OS

Additional capabilities for diagnostics (networking, hardware etc.)

Alternatives: Open Firmware (PowerPC, SPARC), coreboot

(legacy BIOS not supported any longer)

EFI specification

Digital Rights Management

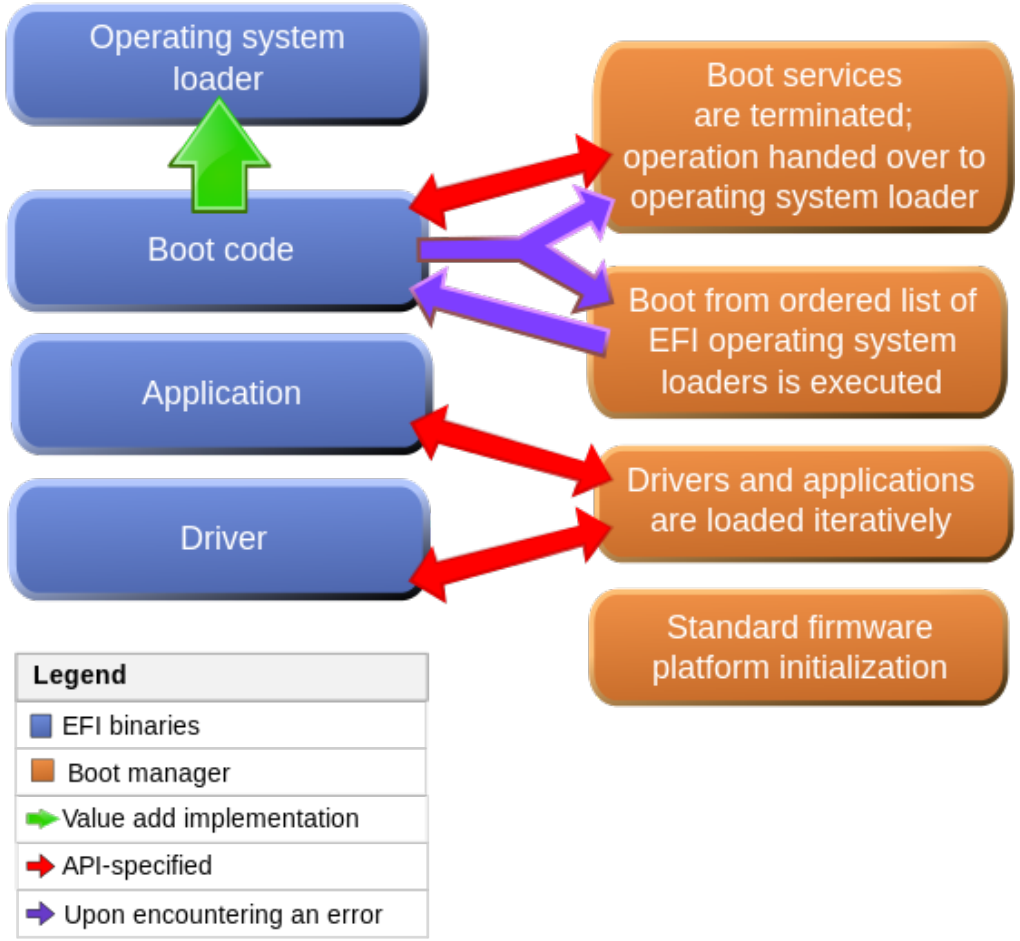
Remote maintenance using own network stack
- security?

Preboot execution environment

Graphics adapter support

Sandbox mode

Support for HD > 2 TB for booting



Booting – Kernel

Bootloader passes control to kernel

Detect/initialize hardware
- CPU, e.g. on x86, switch from real (8086) to protected mode
  - Enables support for memory > 1MB, virtual memory, process isolation
  - Possibly start-up additional CPUs in multi-processor systems
- System buses and memory (“north bridge”)
- Peripheral buses and devices (“south bridge”)
- May require loading of additional device drivers from disk
  - Problem: Ordering of steps in initialization procedure non-trivial
  - Workaround: Append file system containing drivers to kernel image, use bootloader to load most important drivers into RAM

Start kernel subsystems
- Allocate/initialize core data structures, e.g., process queues, page tables, ...

Setup environment for user processes
- Mount root file system
- Start scheduler, enable multi-tasking

Start first user-space process, e.g. /sbin/init in Unix systems

Booting – Init (Unix SysV Init)

SysV Init starts system services according to runlevels:

Typical runlevels:

- Runlevel 0: Halt
- Runlevel 1: Single-user mode
- Runlevel 2: Not used (user-definable)
- Runlevel 3: Full multi-user mode
- Runlevel 4: Not used (user-definable)
- Runlevel 5: Full multi-user mode (GUI login screen)
- Runlevel 6: Reboot

➢ Vendor-specific variations

Typical system services:

- System logging daemon
- Remote Procedure Call (RPC) server
- Print spooler
- Mail Transfer Agent (MTA)
- SNMP daemon
- SSH server
- Login shell
- X Windows System (GUI)

Also performs more complex initialization, e.g. networking

Questions & Tasks

- Do all computers perform booting? Think of different application scenarios.
- Check the booting process of your computer. Depending on the different platforms you might need to press certain keys during the initialization to see all steps.
- Why do we sometimes need several stages of booting?
- Check the story behind UEFI and its implications when it comes to secure boot!

UNIX Daemons

System services on UNIX-derived systems

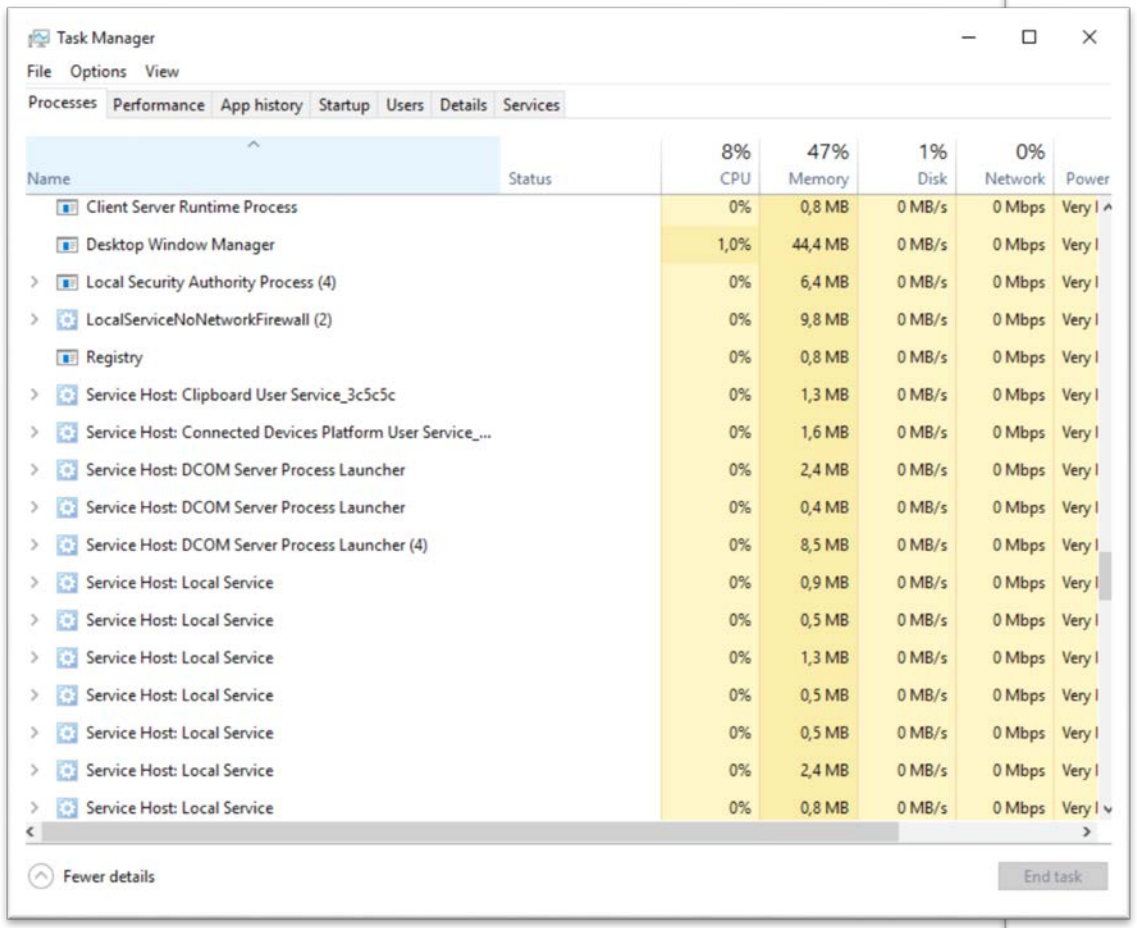
Typically started at boot time

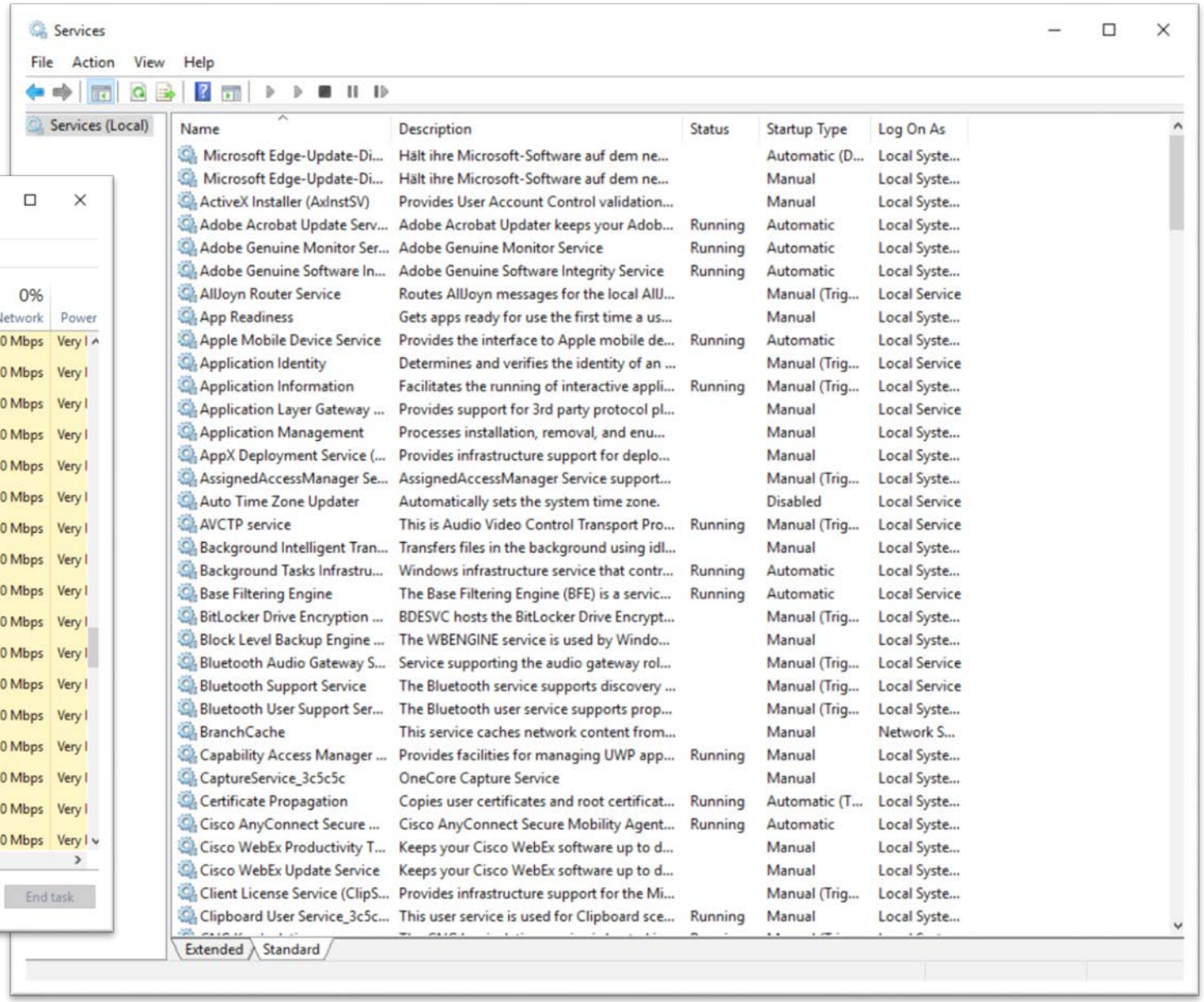
Process that runs in background

- Disassociate from console (if any)
- Fork a child on start-up, then exit
  - ➢ **`init`** adopts the child
- Set working directory to root directory
- Close open file descriptors (if any)
- Set **`stdin`**, **`stdout`**, and **`stderr`** to logfile, console, or **`/dev/null`**

➢ May store PID in known file for future interaction

Windows offers services





System Services Example: `crond`

Periodically runs commands, referred to as “cron jobs”

Commands are configured in **`/etc/crontab`**:

```
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file.
# This file also has a username field, that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

MAILTO=staff-simpel2@inf.fu-berlin.de

# m h dom mon dow user  command
25 6    * * *   root    test -e /usr/sbin/anacron || run-parts --report /etc/cron.daily
47 6    * * 7   root    test -e /usr/sbin/anacron || run-parts --report /etc/cron.weekly
52 6    1 * *   root    test -e /usr/sbin/anacron || run-parts --report /etc/cron.monthly
#

#tos: der cron job sollte nicht root-mail zuspammen..
0 4 * * * root /etc/simpel/delayed_update >/dev/null 2>&1
#
# vielleich hilft das gegen eingefrorene maeuse, vv 17.6.02
#0 7 * * * root /etc/init.d/gpm restart > /dev/null 2>&1
```

System Services Example: `syslogd`

Daemon for local and/or remote message logging

Uses Syslog protocol (RFC 3164) to receive messages via UDP or TCP on port 514

Writes messages to **`/var/log/syslog`**:

```
Nov 28 01:31:59 vaio syslogd 1.4.1#18: restart.
Nov 28 01:31:59 vaio anacron[4603]: Job `cron.daily' terminated (mailing output)
Nov 28 01:32:00 vaio anacron[4603]: Normal exit (1 job run)
Nov 28 01:36:50 vaio kernel: usb 1-1.1: new high speed USB device using ehci_hcd and address 7
Nov 28 01:36:50 vaio kernel: usb 1-1.1: configuration #1 chosen from 1 choice
Nov 28 01:36:50 vaio kernel: usb-storage: device found at 7
Nov 28 01:36:50 vaio kernel: usb-storage: waiting for device to settle before scanning
Nov 28 01:36:55 vaio kernel:   Vendor: WDC WD16  Model: 00BB-22GUC0       Rev: 0811
Nov 28 01:36:55 vaio kernel:   Type:   Direct-Access                      ANSI SCSI revision: 00
Nov 28 01:36:55 vaio kernel: usb-storage: device scan complete
Nov 28 01:36:55 vaio kernel: SCSI device sda: 312581808 512-byte hdwr sectors (160042 MB)
Nov 28 01:36:55 vaio kernel: sda: test WP failed, assume Write Enabled
Nov 28 01:36:55 vaio kernel: sda: assuming drive cache: write through
Nov 28 01:36:55 vaio kernel:  sda: sda1
Nov 28 01:36:55 vaio kernel: sd 0:0:0:0: Attached scsi disk sda
Nov 28 01:37:03 vaio hald: mounted /dev/sda1 on behalf of uid 1000
Nov 28 01:55:40 vaio hald: unmounted /dev/sda1 from '/media/TREKSTOR-1' on behalf of uid 1000
Nov 28 01:56:23 vaio kernel: UDF-fs: No VRS found
Nov 28 01:56:23 vaio kernel: Unable to identify CD-ROM format.
```

Separate daemon **`klogd`** to log kernel messages

System Services Example: `lpd`

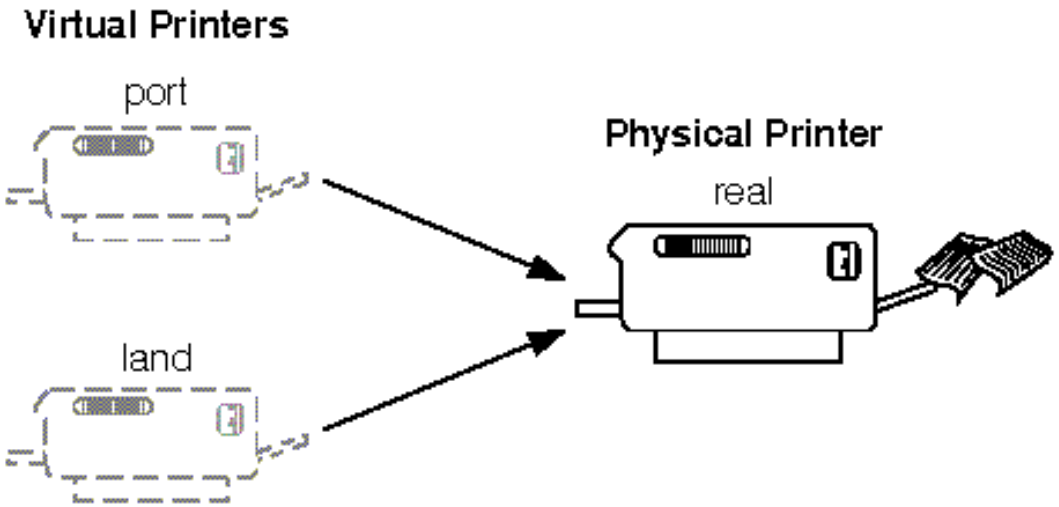
Provides printer spooling

Manages (multiple) queues for each connected printer

Controls/schedules print jobs of (multiple) users

Virtual printers may transparently provide different options to real printer

- ➢Preprocessing, filtering of documents



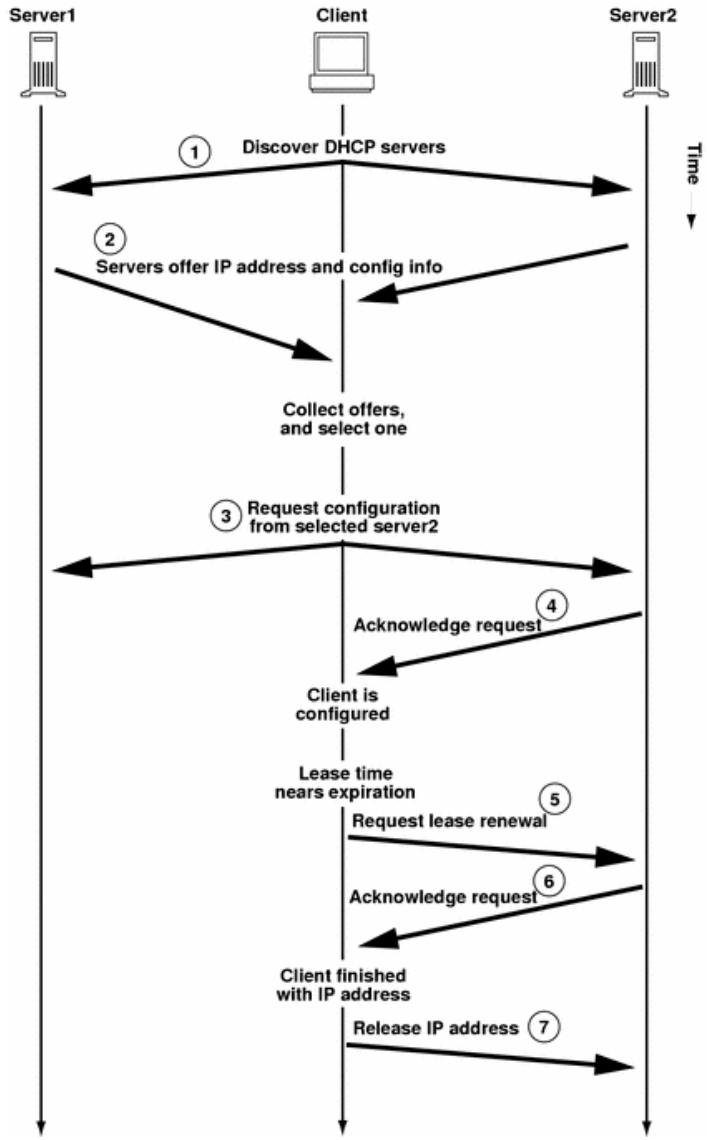
System Services Example: dhcpd

Dynamic Host Configuration Protocol (DHCP, RFC 2131)

Acts as server that manages network information (e.g. IP addresses) for local network

Clients request configuration information at boot time, i.e. upon configuring network interfaces

➢Effectively provides system service for *remote* computers

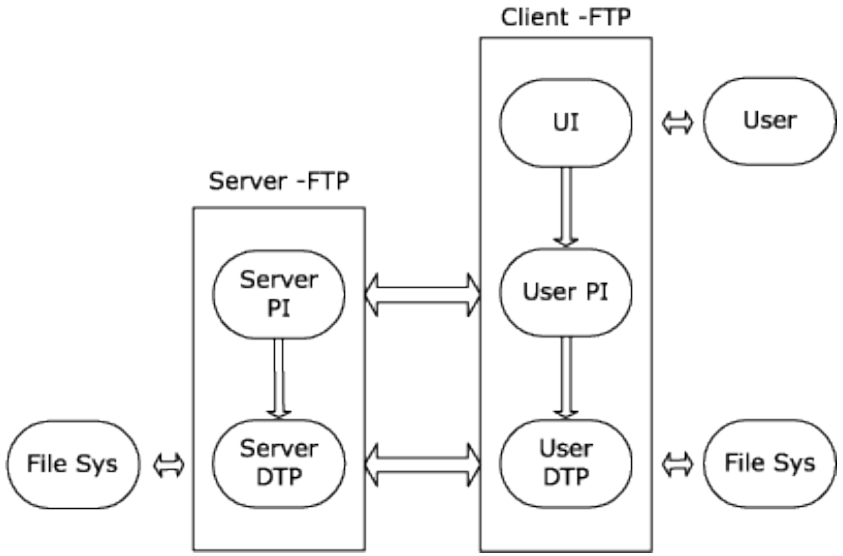


System Services Example: `ftpd`

File Transfer Protocol (FTP, RFC 959)

Allows remote access to data stored on local file system

- Separate connections for commands and data

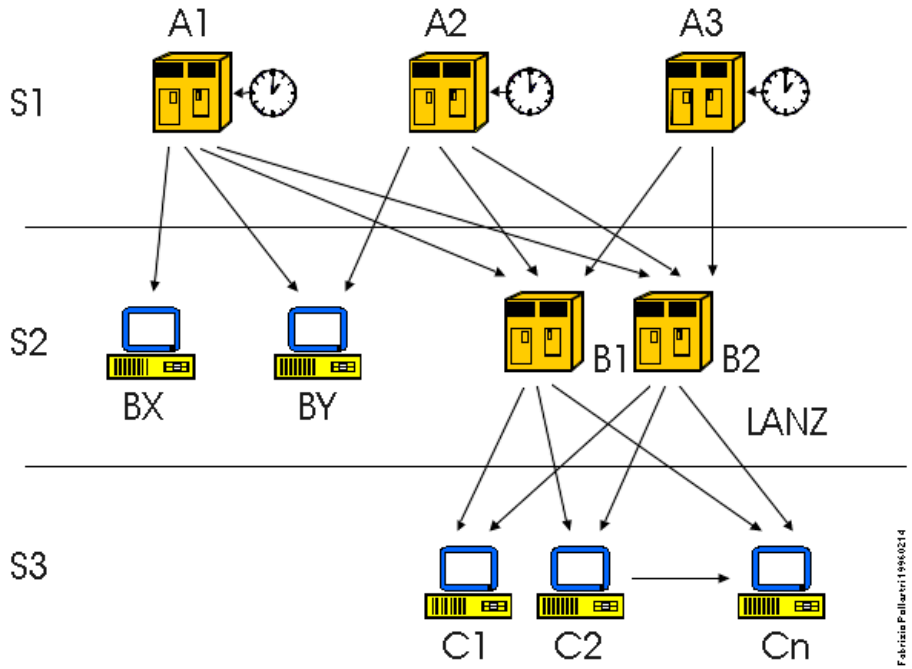


System Services Example: `ntpd`

Network Time Protocol (NTP, RFC 1305)

Synchronizes time of local computer with time servers / other computers

➢Example: **`time.fu-berlin.de`**



System Services Example: `getty`

`getty`:
Opens TTY (terminal typewriter) port and prompts for username (“**`login:”`**)
Upon entry of valid user name, start **`/bin/login`** process

`login`:
Authenticate user by querying for password
- Limited number of retries

Set user and group ID and environment variables as specified in **`/etc/passwd`**
- ➢Effectively drop administrator (“root”) privileges

Start shell, e.g. **`/bin/bash`**

➢Boot process complete, user logged in, *start working!*

Questions & Tasks

- Check the running deamons in your UNIX system – or services if you run Windows. There are quite many …
  The operating systems sometimes offer nice tools to view these deamons/services.

CIA Triad: Security Objectives

Confidentiality

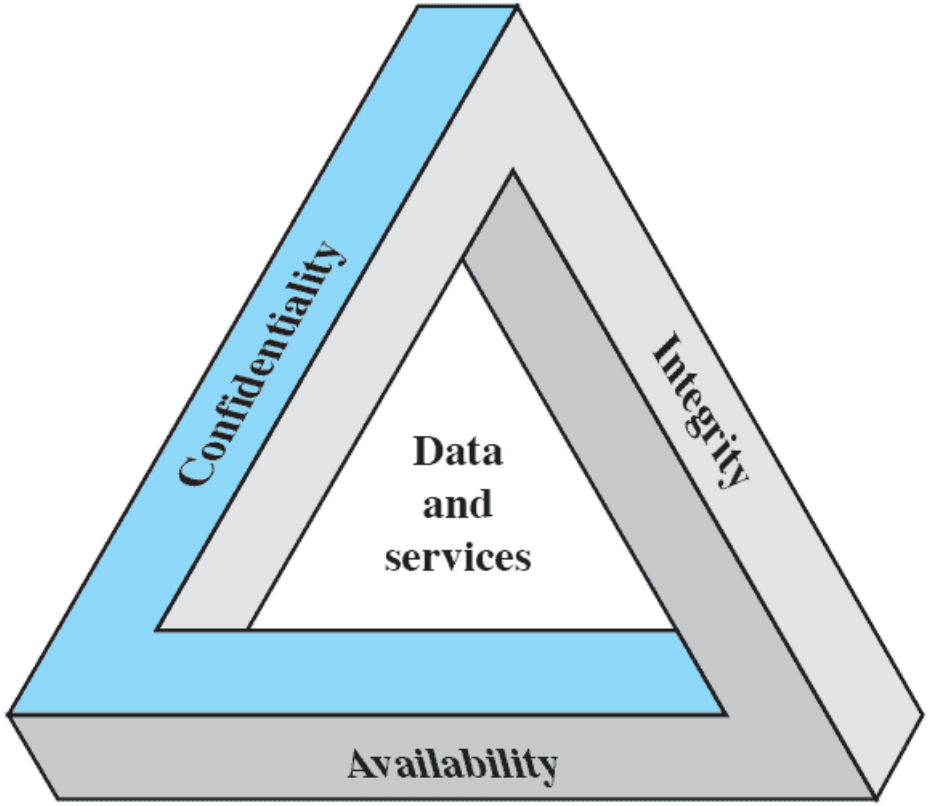
- a loss of confidentiality is the unauthorized disclosure of information

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



The Security Requirements Triad

Additional Concepts

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

Passive Attacks

Learn/use information from system but do not affect
- Eavesdropping and monitoring

Goal: Obtain information

Difficult to detect, since no alteration of data
- Emphasis is on prevention rather than detection
- Encryption for prevention

Types

- release of message contents
- traffic analysis

Active Attacks

Modify data or create false data

Four categories:

Replay
- first capture data / second retransmit

Masquerade
- pretend to be a different entity

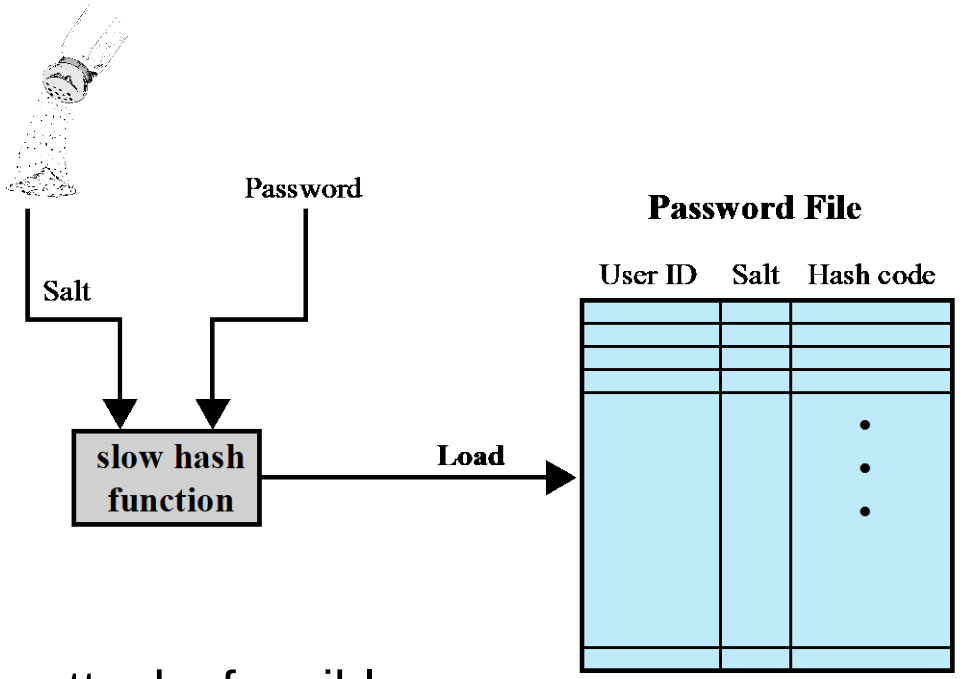
Modification of messages
- alter some portion of a message / delay or reorder message

Denial of service
- prevent normal use of communications facilities
- disrupt entire network by disabling the network or by overloading it

Authentication – Password Creation

Library function **`crypt()`** encrypts user's password by adding some randomness ("salt")

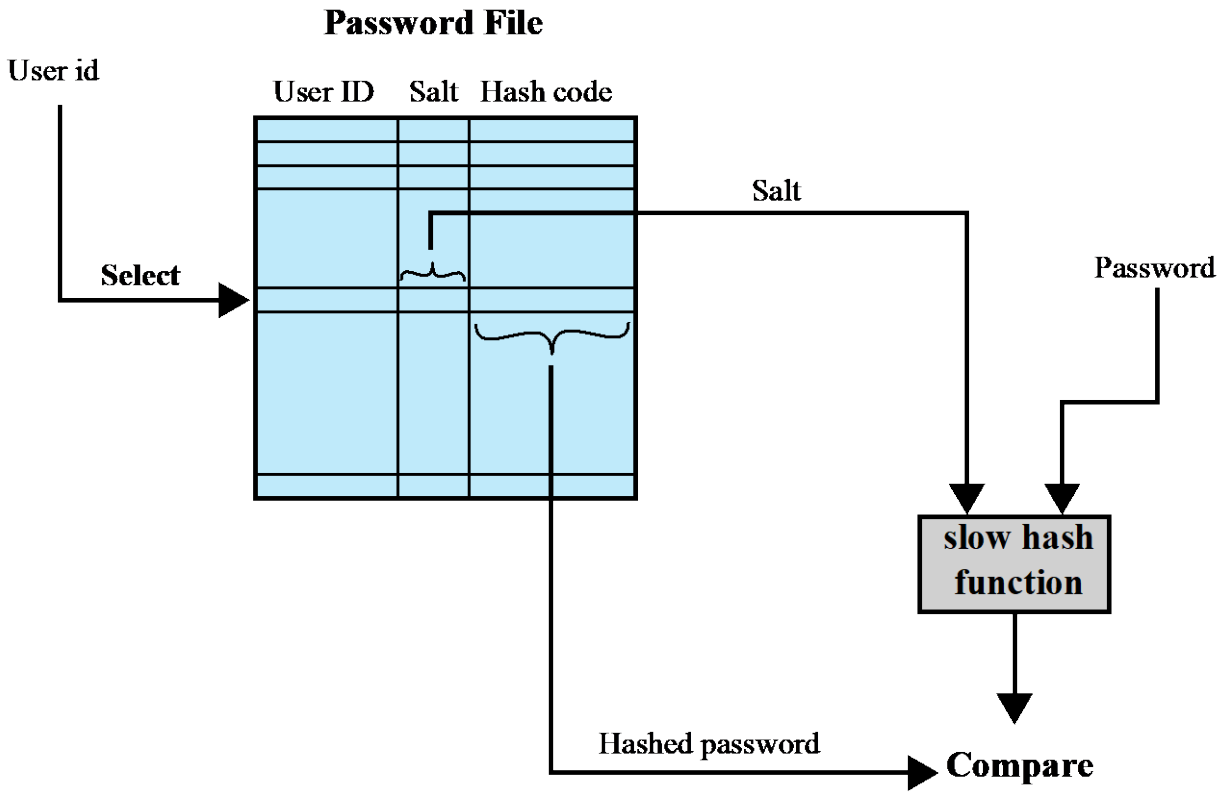
- Cryptographic hash function
- Based on Data Encryption Standard (DES) algorithm



➢Limited key space, brute-force attacks feasible

Authentication – Password Verification

Performed by **`login`** parameterized with user name (ID)



Access Control – Permission Bits

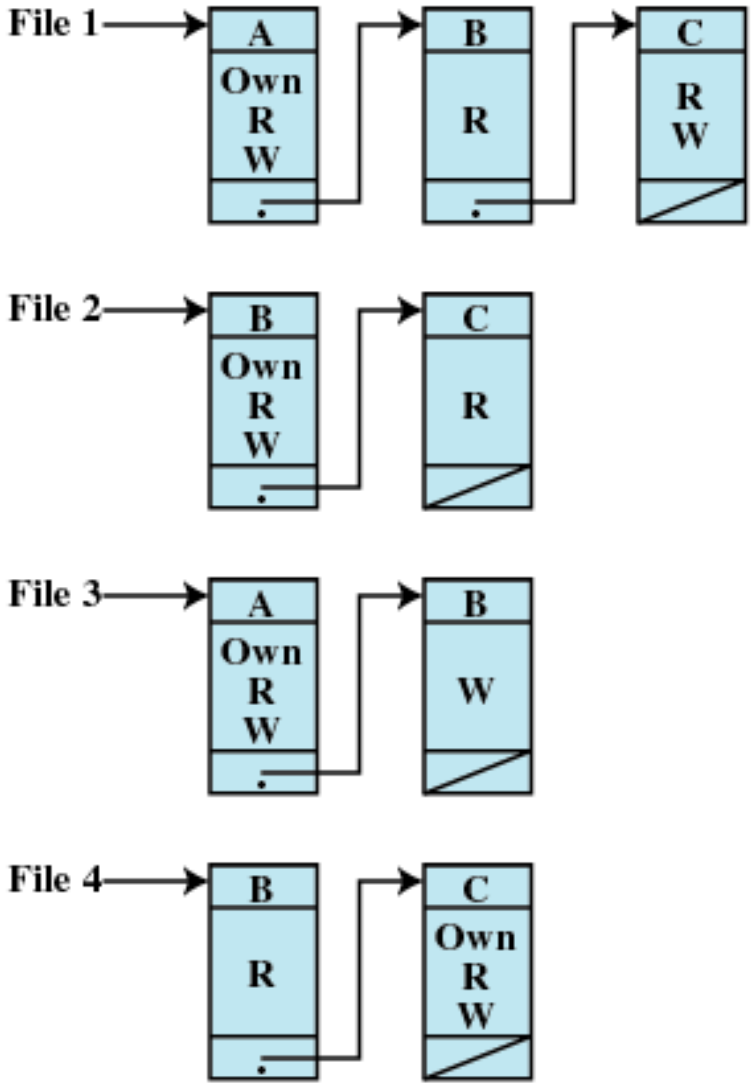
Bit vector per object (e.g. file, directory) encodes:

- Entities (users): Owner, group, other
- Access rights: Read, write, execute (“change into” for directories)
  - ➢Example: **`rwxr-x---`**
    - Owner may read, write and execute
    - Owning group may read and execute
    - Others aren’t allowed to do anything
- Special bits for advanced features, e.g. SUID big – execute as owner
  - ➢Subtle semantics: Changing name of a file requires write access to containing directory

More advanced: Access matrix

	File 1	File 2	File 3	File 4	Account 1	Account 2
User A	Own R W		Own R W		Inquiry Credit	
User B	R	Own R W	W	R	Inquiry Debit	Inquiry Credit
User C	R W	R		Own R W		Inquiry Debit

Access Control – Access Control Lists



- Permissions bits at times not flexible enough
- Access matrix has scalability issues, is sparsely populated

Access Control Lists (ACLs):

Variable number of bit vectors in linked list

- Slight access overhead

One per required entity

Need additional semantics

- What to do if access granted to group but not to specific user?

Access Control – Capabilities

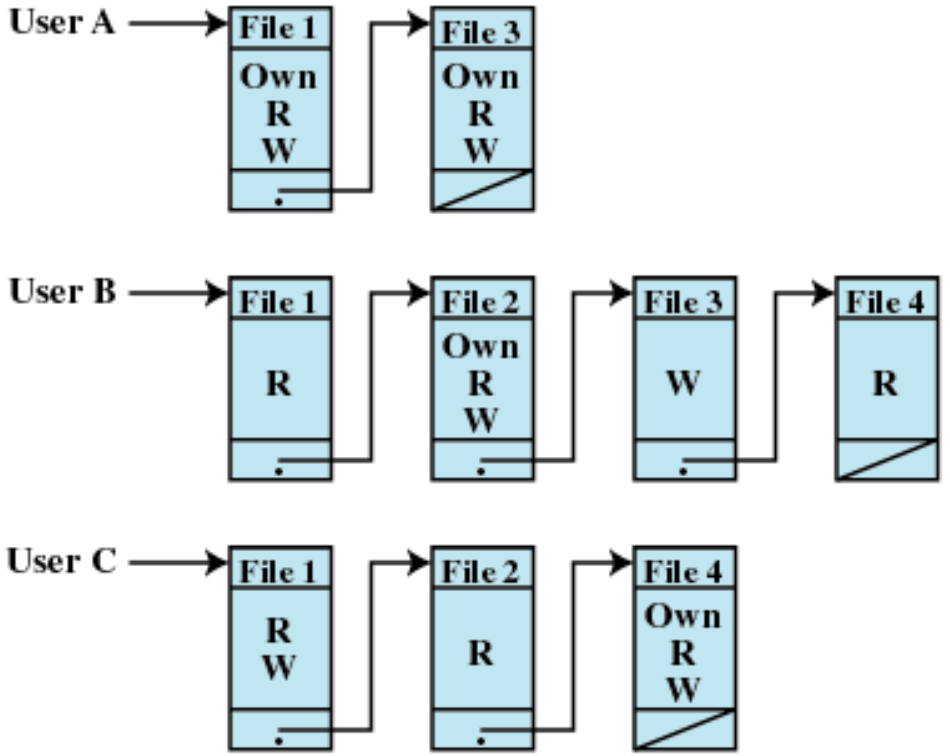
Assign set of capabilities to each entity

➢Example:

- File access (rarely used)

System capabilities:

- Traditional:
  - Administrator (root): Can do everything
  - User: Can do nothing
- With capabilities:
  - Fine grained control
  - E.g. access to specific system calls, kernel subsystems



Attack Vector – Directory Traversal

Trick “weak” program to access protected/internal file

- Exploits insufficient checking/sanitization of user-supplied file name

Example:

- Documents for student homepages are stored in **`/web/page.mi.fu-berlin.de/web-home/user_name/`**
  - E.g. **`http://page.mi.fu-berlin.de/~userA/index.html`** is located at **`/web/page.mi.fu-berlin.de/web-home/userA/index.html`**
- What happens if an attackers queries the web server for **`http://page.mi.fu-berlin.de/../../../../etc/passwd`**?

- Defenses:
  - Scan input for illegal character sequences
    - “**`../`**” in this examples, but this may be masked as “%2E%2E%2F” (ASCII encoded)
  - Canonization, i.e. convert input into a canonical form
    - Depends on type of input, absolute path (without “**`../`**”) for path names
  - Limit access of web server process to file system

Countermeasures – Limited Access

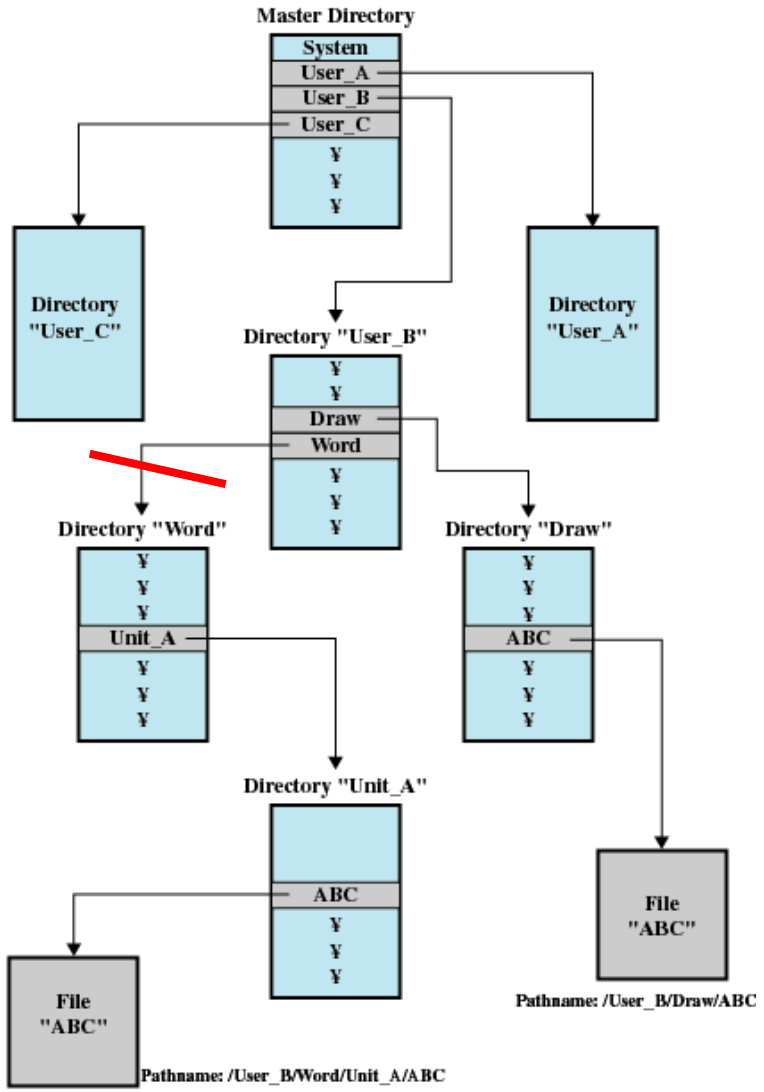
System support of limiting view of entire system for specific process

`chroot` system call changes root of file system as perceived by process

- Process is caught in “chroot jail”, cannot break out

➢ Good precaution for processes that

- are self-contained
- deal with untrusted input



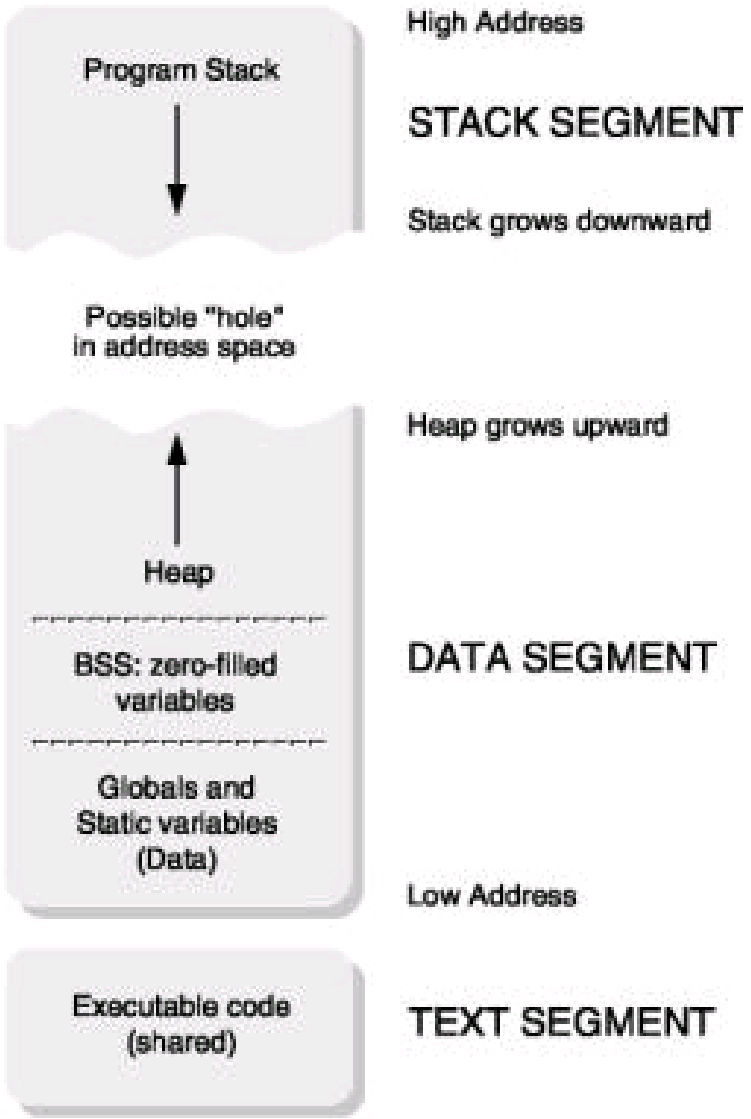
Attack Vector – Buffer Overflow

Overwrite flow control information of a process to gain control of the process

- Works for processes that process external data without proper bounds checking

User-space process address space layout:

- Stack for function calls, parameters, local variables
- Heap for dynamically managed memory, e.g. with **`malloc()`**, ...



Attack Vector – Buffer Overflow

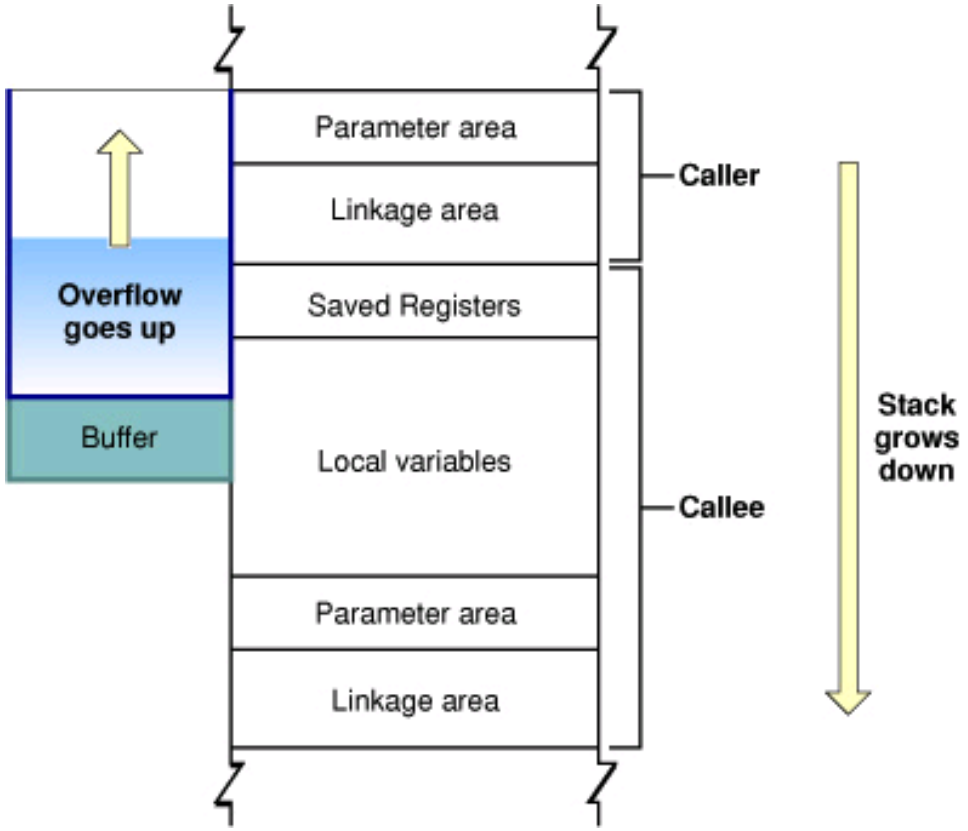
➢Exploit weak bounds checking in buffer operations

Example:

- Allocate fixed size buffer
- Copy untrusted data to buffer using **`strcpy()`** (instead of **`strncpy()`**)
- May overwrite control information in stack frame in order to change program flow

Heap-based variants exist

➢Defense: Use safe versions of library routines, i.e. know how much memory you *have*



Attack Vector – Buffer Overflow

Example:

```
#include <stdio.h>

int main()
{
    char str1[4];
    char str2[4];
    gets(str2);

    printf("str1=%s, str2=%s\n", str1, str2);
}
```

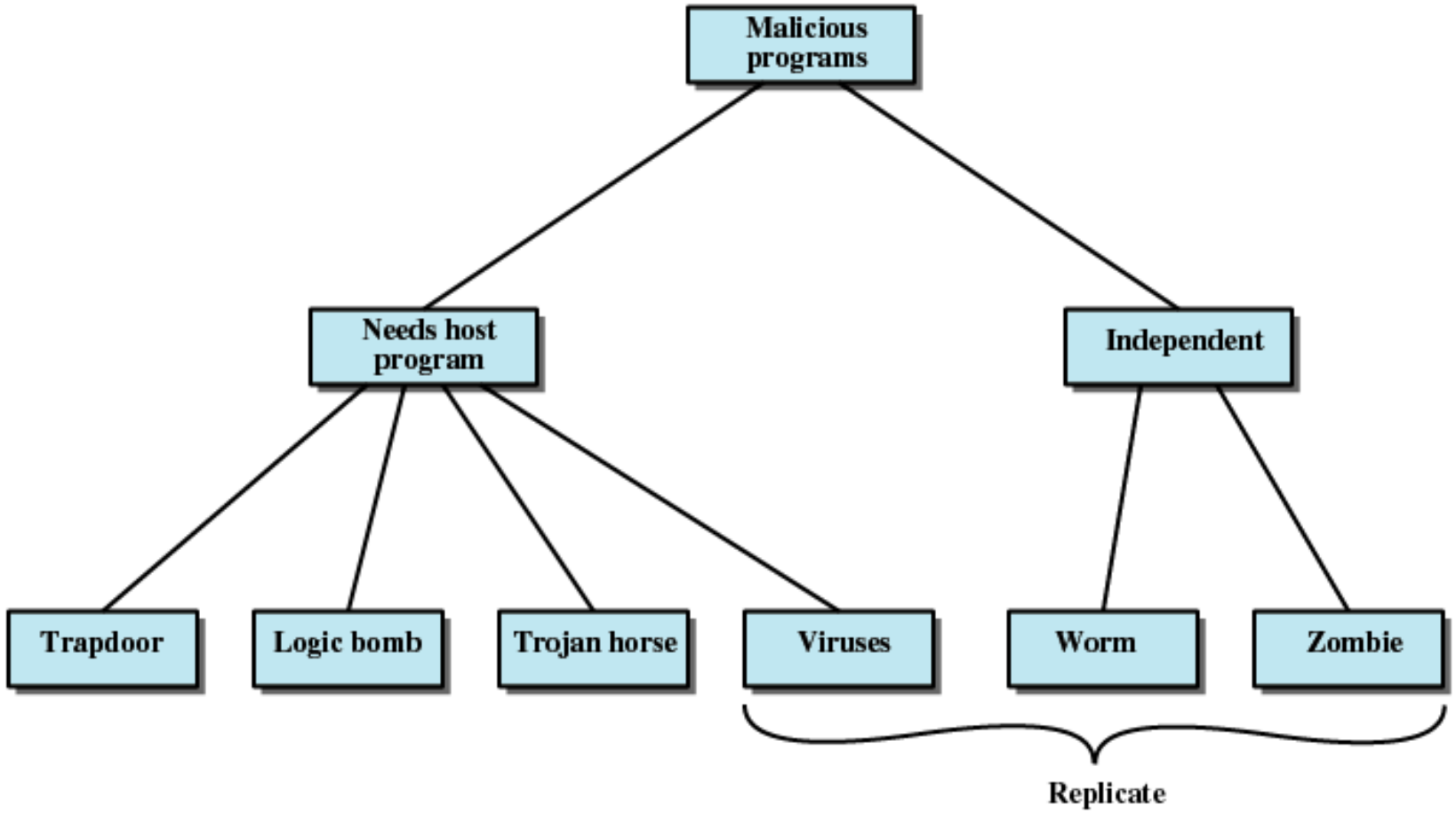
Questions & Tasks

- Check `/etc/passwd` on a Unix system and in this context “shadow passwords” – why are they needed?
- Encryption helps in many ways against active and passive attacks. But what about denial of service?

Classification of Threats

Classification by dependency on environment

- Related to propagation method, and thus potential targets



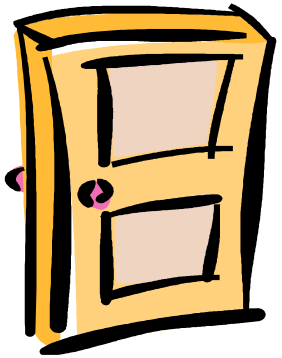
Trapdoor/Backdoor

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

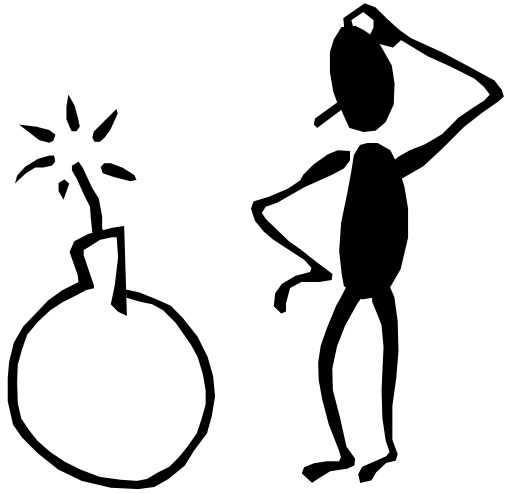


Logic Bomb

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:

- continuing to perform the function of the original program and additionally performing a separate malicious activity
- continuing to perform the function of the original program but modifying the function to perform malicious activity or to disguise other malicious activity
- performing a malicious function that completely replaces the function of the original program

Mobile Code

Programs that can be shipped unchanged to a heterogeneous collection of platforms and execute with identical semantics

Transmitted from a remote system to a local system and then executed on the local system without the user's explicit instruction

Often acts as a mechanism for a virus, worm, or Trojan horse to be transmitted to the user's workstation

Takes advantages of vulnerabilities

Popular vehicles for mobile code include Java applets, ActiveX, JavaScript, and VBScript

Multiple-Threat Malware

Infects in multiple ways

Typically the multipartite virus is capable of infecting multiple types of files

A blended attack uses multiple methods of infection or transmission to maximize the speed of contagion and the severity of the attack

An example of a blended attack is the Nimda attack

Nimda uses four distribution methods

- E-mail
- Windows shares
- Web servers
- Web clients

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 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

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

Macro Viruses

In the mid 1990's macro viruses became by far the most prevalent type of virus

Macro viruses are particularly threatening because:

- they are platform independent; e.g. many macro viruses infect Microsoft Word documents or other Microsoft Office documents
- they infect documents, not executable portions of code
- they are easily spread; a very common method is by electronic mail
- file system access controls are of limited use in preventing their spread
- Thus: new Office version disable macros by default

E-Mail Viruses

The first rapidly spreading e-mail viruses made use of a Microsoft Word macro embedded in an attachment

In 1999 a newer, more powerful version of the e-mail virus appeared

- can be activated merely by opening an e-mail that contains the virus rather than opening an attachment
- the virus uses the Visual Basic scripting language supported by the e-mail package

Worms

A program that can replicate itself and send copies from computer to computer across network connections

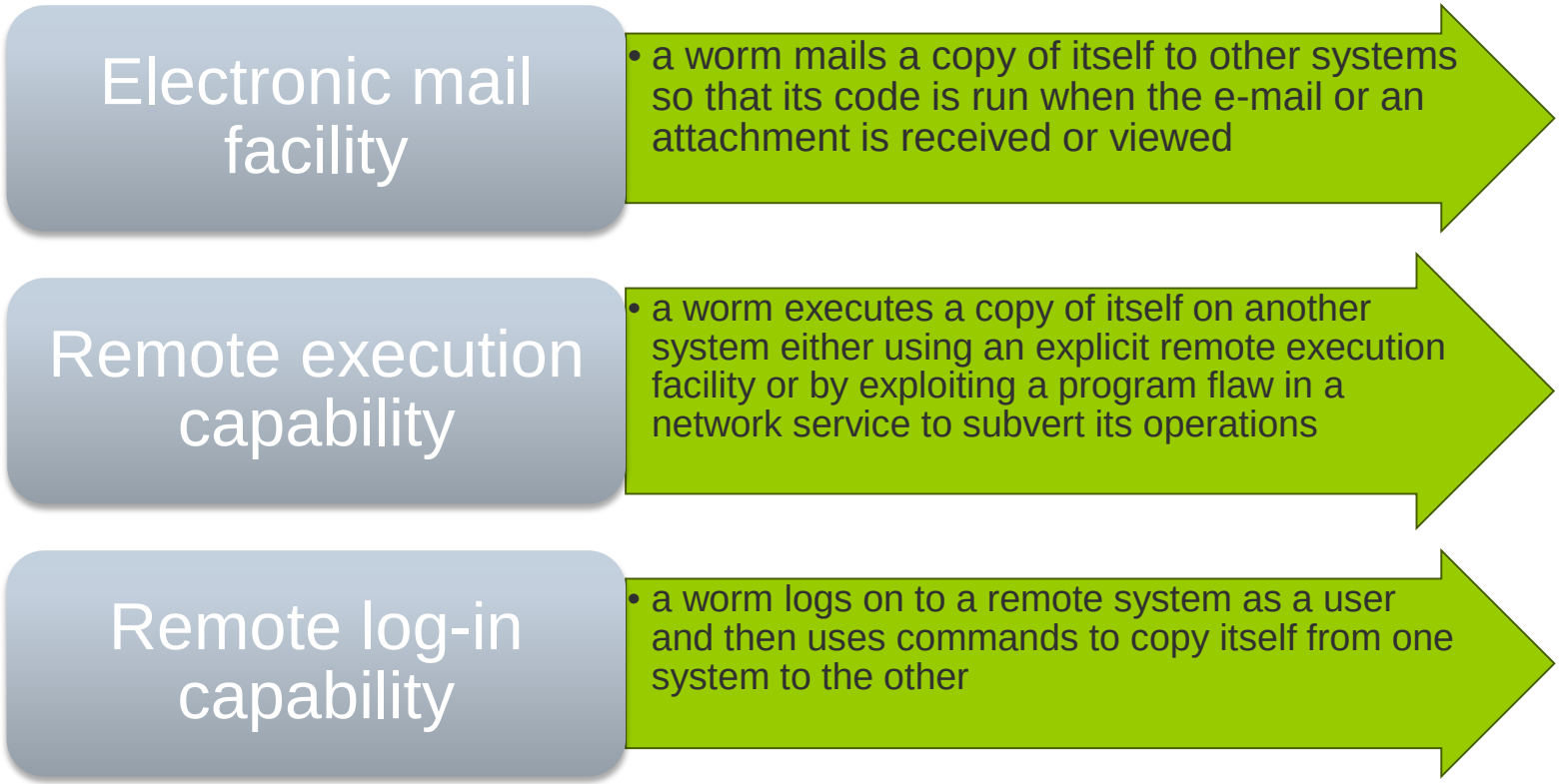
Upon arrival the worm may be activated to replicate and propagate again

In addition to propagation the worm usually performs some unwanted function

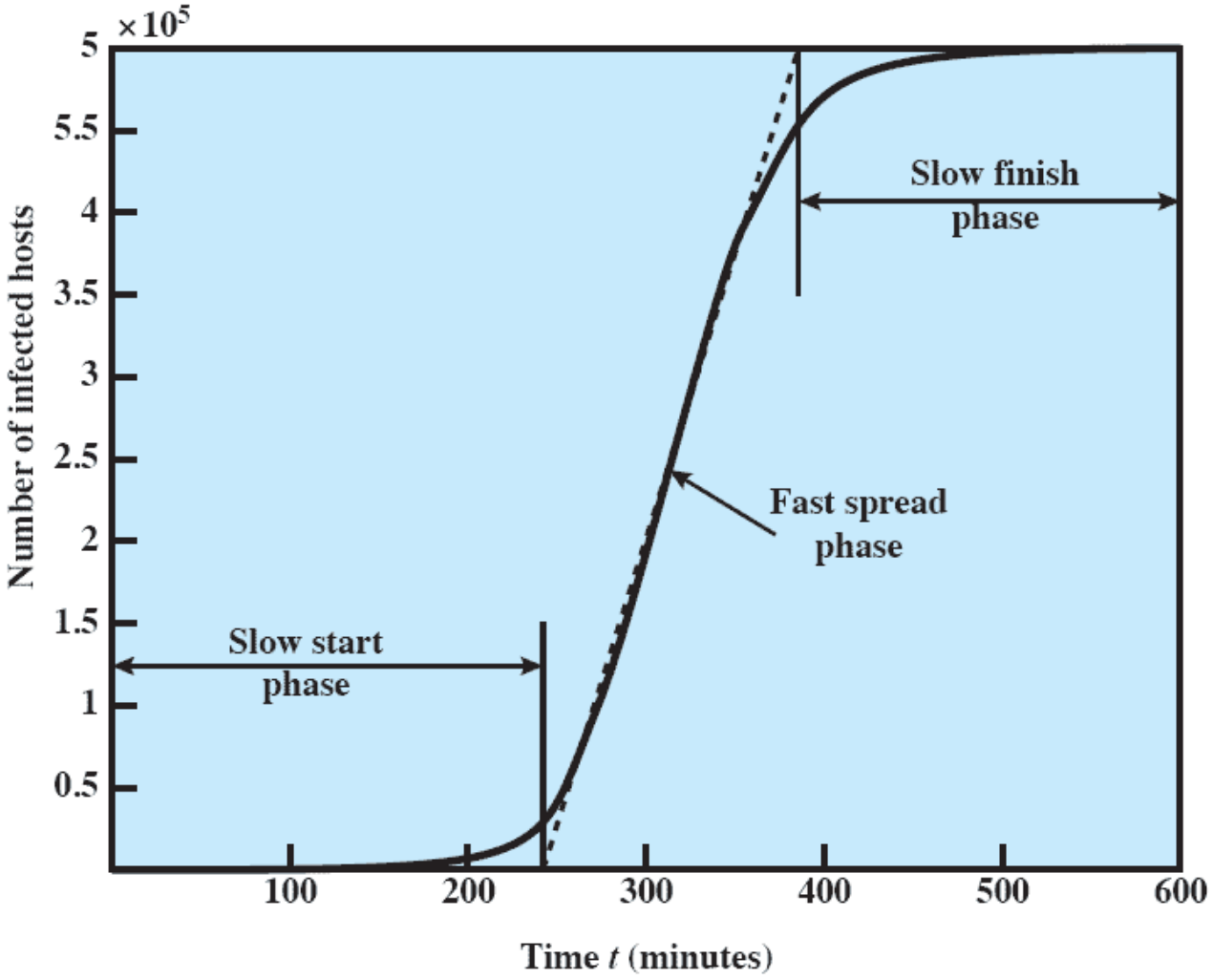
Actively seeks out more machines to infect and each machine that is infected serves as an automate launching pad for attacks on other machines

Worm Propagation

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



Worm Propagation Model



Bots

A program that secretly takes over another Internet-attached computer and then uses that computer to launch attacks that are difficult to trace to the bot's creator

- also known as a Zombie or drone

Typically planted on hundreds or thousands of computers belonging to unsuspecting third parties

- today millions of "things" used in the context of the Internet of Things, attacks > 1 Tbit/s

Collection of bots acting in a coordinated manner is a botnet

A botnet exhibits three characteristics:

- the bot functionality
- a remote control facility
- a spreading mechanism to propagate the bots and construct the botnet

Rootkit

Set of programs installed on a system to maintain administrator (or root) access to that system

Root access provides access to all the functions and services of the operating system

The rootkit alters the host's standard functionality in a malicious and stealthy way

- with root access an attacker has complete control of the system and can add or change programs and files, monitor processes, send and receive network traffic, and get backdoor access on demand
- A rootkit hides by subverting the mechanisms that monitor and report on the processes, files, and registries on a computer

Exploits

Quite often today's computers contain several mechanisms and programs defending against malware/viruses etc.

However, malicious code may still be able to take over control by using exploits

Exploit: code that takes advantage of bugs/vulnerabilities of a computer (SW/HW) to cause unintended behavior

Goals: e.g. privilege escalation to execute malicious code, DoS attack

Zero day exploit: cyber attack that occurs on the same day a weakness is discovered

Questions & Tasks

- Backdoors seem to be a bad idea. Why are they still used?
- Why do we have the problems of viruses, exploits, DoS attacks etc. at all? Think of technical reasons but also motivations!

Related System Calls (Linux)

`uid_t getuid(void)`
`int setuid(uid_t uid)`
- Gets/sets the user ID

`int chroot(const char *path)`
- Changes root directory of process to **`path`**
- Access to files outside “chroot jail” not possible *afterwards*

Library functions:

`char *crypt(const char *key, const char *salt)`
- Encrypts user’s password **`key`** using **`salt`** (2 bytes) to add randomness
- Returns encrypted password as 13-character string
  - First 2 bytes correspond to **`salt`**

Content (1)

1. Introduction and Motivation
   - Tasks
   - Services
   - Virtual Resources
   - Historical Perspective
   - Examples
   - Tools
2. Subsystems, Interrupts and System Calls
   - System Structure
   - Flow of Control
   - System Library
   - POSIX
3. Processes
   - Definition
   - Implementation
   - State Model
4. Memory
   - Paging & Segmentation
   - Virtual Memory
   - Swap Policies
5. Scheduling
   - Types of Scheduling
   - Decision Modes
   - Process Priorities
   - Scheduling Policies
6. I/O and File System
   - Devices
   - Buffering and Caching
   - Files and Directories
7. **Booting, Services, and Security**
   - System Startup
   - System Services
   - Security Issues

Content (2)

8. Networked Computer & Internet
9. Host-to-Network
10. Internetworking
11. Transport Layer
12. Applications
13. Network Security
14. Example