Operating System Security - Basic Principles

CS239
Advanced Operating Systems

November 10, 1998
Dr. Richard Guy

Lecture Source Materials

- Singhal, Chapter 14: **Resource Security and Protection: Access and Flow Control**
- Stallings, Operating Systems, Ch. 14: **Security**

Outline

- Introduction
- Security threats
- Basic security principles
- Security on a single machine
  - Distributed systems security in Spr99 CS239

Introduction

- Security is an *engineering* problem
  - must tradeoff safety, cost, and inconvenience
  - not much solid theory in the field
- Hard to provide any real guarantees
  - making mistakes is easy
  - nature of the problem implies that mistakes are always exploited

Why Is Security Important Today?

- Electronic commerce
- Increased general reliance on computers
- Ubiquity of networks
- Limited computer knowledge of many users
- More users means more bad guys

Fundamental Constraints of Practical Computer Security

- Security isn't free
  - if it costs too much, it won't be used
- If it isn't easy, it won't be used
- Misuse often nullifies security measures
- Match the stringency of the measures to the threat being countered

Security is only as Strong as the Weakest Link

- Those breaking security will find and attack the weakest point
  - expensive lock on a cheap door doesn't work
- Must consider security problems/solutions as part of an integrated system, not just a single component

Security Threats

- Overall threat characteristics
  - extremely wide range
  - from a wide variety of sources
  - requiring a wide variety of countermeasures
- Countering most threats costs something
  - Often try to counter as few as one can afford

Security Aspects

- Physical
- Social
- Theoretical

Physical Security

- Some threats involve physical access to the equipment itself
  - theft
  - destruction
  - tampering
- Physical threats usually require physical prevention methods

Social Engineering and Security

- Computer security easily subverted by bad human practices
  - e.g., giving password out over the phone to anyone who asks
- Social engineering attacks tend to be cheap, easy, effective
- So all our work may be for naught

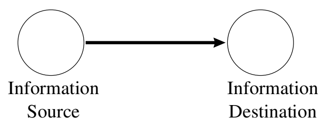
Security Requirements

- Secrecy
- Integrity
- Availability
- Exclusivity

A Classification of Threats

Threats are viewed as types of attacks on normal service

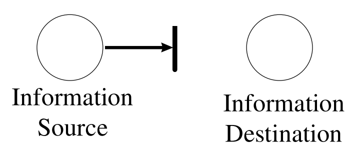
So what is normal service?



Classification of Threat Types

- Interruption
- Interception
- Modification
- Fabrication

Interruption



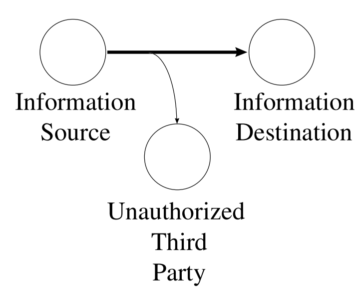
Interruption Threats

- Denial of service
  - prevents source from sending information to receiver
  - or receiver from sending requests to source
- A threat to availability requirement

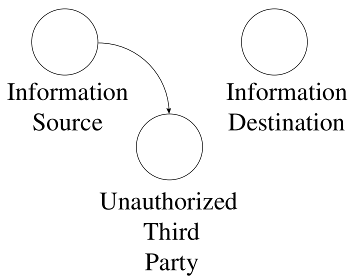
How Does an Interruption Threat Occur?

- Destruction of HW/SW
- Interference with communications channel
- Overloading a shared resource

Interception



Another Type of Interception



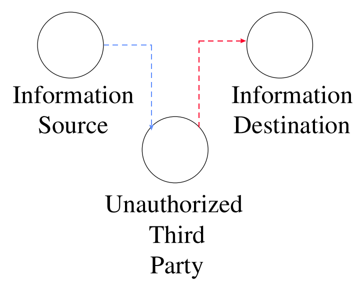
Interception Threats

- Data or services provided to unauthorized party
  - Either in conjunction with, or independent of, authorized access
- A threat to secrecy requirement
- Also a threat to exclusivity

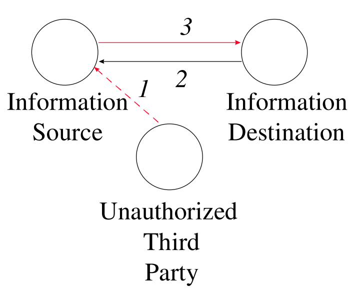
How Do Interception Threats Occur?

- Eavesdropping
- Masquerading
- Break-ins
- Illicit data copying

Modification



Another Type of Modification Threat



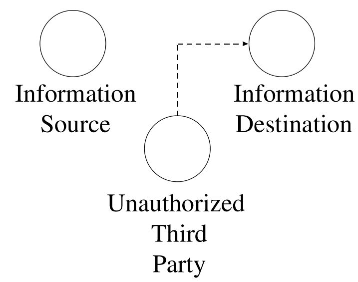
Modification Threats

- Unauthorized parties modify data
  - either on the way to the users
  - or permanently at the servers
- A threat to integrity requirement

How Do Modification Threats Occur?

- Interception of data requests
- Masquerading
- Illicit access to servers/services

Fabrication



Fabrication Threats

- Unauthorized party inserts counterfeit objects into the system
  - causing improper changes in data
  - or improper use of system resources
- A threat to integrity requirement

How Do Fabrication Threats Occur?

- Masquerading
- Bypassing protection measures
- Duplication of legitimate requests

Active Threats vs. Passive Threats

- *Passive threats* are forms of eavesdropping
  - No modifications, injections of requests, occur
- *Active threats* are more aggressive
- Passive threats are mostly to secrecy
- Active threats jeopardize
  - availability
  - integrity
  - exclusivity

What Are We Protecting?

- Hardware
- Software
- Data
- Communications lines and networks
- “Cycles” and economic values

Basic Security Principles

- Definitions
- Design Principles
- Mechanisms

Definitions

- *Security* is a policy
  - "no unauthorized user may access this file"
- *Protection* is a mechanism
  - "the system checks user identity against access permissions"
- Protection mechanisms <u>implement</u> security policies

Design Principles for Secure Systems

- Economy
- Complete mediation
- Open design
- Separation of Privileges
- Least privilege
- Least common mechanism
- Acceptability
- Fail-safe defaults

Economy in Security Design

- Economical to develop
  - and to use
  - and to verify
- Should add little or no overhead
- Should do only what needs to be done
- Generally, try to keep it simple and small

Complete Mediation

- Apply security on <u>every</u> access to an object that a mechanism is meant to protect
  - e.g., each read of a file, not just the open
- Involves checking access on <u>everything</u> that could be attacked

Open Design

- Don't rely on "security through obscurity"
- Assume all potential intruders know everything about the design
  - and completely understand it

Separation of Privileges

- Provide mechanisms that separate the privileges used for one purpose from those used for another
- Allows flexibility in the security system
- E.g., separate access control on each file

Least Privilege

- Give bare minimum access rights required to complete a task
- Require another request to perform another type of access
- E.g., don't give write permission if program only asked for read

Least Common Mechanism

- Avoid sharing parts of the security mechanism among different users
- Coupling users leads to possibilities for them to breach the system

Acceptability

- Mechanism must be simple to use
  - simple enough that people will use it automatically
- Must rarely or never prevent permissible accesses

Fail-Safe Designs

- Default to lack of access
- So if something goes wrong/is forgotten/isn't done, no security is lost
- If important mistakes are made, you'll find out about them
  - without loss of security
  - but if it happens too often . . .

Sharing Security Spectrum

- No protection
- Isolation
- Share all or nothing
- Share with access limitations
- Share with dynamic control

Important Security Mechanisms

- Access control mechanisms
- Authentication
- Encryption
- Passwords
- Other authentication mechanisms
- Auditing features

Data Access Control Mechanisms

- Methods of specifying who can access what in which ways when
- Based on assumption that the system has authenticated the user

Access Matrix

- Describes permissible accesses for the system
- *Subjects* access *objects* with particular *access rights*
- A theoretical concept, never kept in practice

Access Matrix Example

	File 1	File 2	Server Z	Segment 57
Subject A	Read Write	None	Query	Read
Subject B	Read	Write	Update	None
Subject C	None	Read	Start/Stop	None
Subject D	None	None	Query	None

Methods for Implementing Access Matrix

- Access control lists
  - Decomposition by columns
- Capabilities
  - Decomposition by rows

Access Control Lists

- Each object controls who can access it
  - using an *access control list*
- Add subjects by adding entries
- Remove subjects by removing entries

\+ Easy to determine who can access an object
\+ Easy to change who can access object
– Hard to tell what someone can access

Access Control List Example

File 1's ACL	Segment 57's ACL
Subject A: Read, Write Subject B: Read	Subject A: Read

Note that group and negative access rights are possible

Capabilities

- Each subject keeps track of what it can access
  - typically by keeping a *capability* for each object
- Capabilities are like admission tickets

\+ Easy to tell what a subject can access
– Hard to tell who can access an object
– Hard to revoke/control access

Capability Example

Subject A's Capabilities

File 1:	Read Write
Server Z:	Query
Segment 57:	Read

Subject B's Capabilities

File 1:	Read
File 2:	Write
Server Z:	Update

Controlling Capabilities

- If a subject has a capability, can he give it to another subject?
  - If the capability is just a pointer, hard to stop
- But the OS can control capabilities
  - so it can also control their distribution
- Must be inaccessible to users

Some Example Systems

- Capability-based systems
  - Amoeba
  - Hydra
  - Parts of Windows NT
- Access control list-based systems
  - Most Unix systems
  - Andrew
  - Parts of Windows NT

Other Models of Access Control

- Military model
- Information flow models
- Lattice model of information flow

Single System Security

- Protecting "the system"
- Dealing with intruders
- Malicious programs

Protecting a Single Machine

- Only worrying about the security of a single machine (possibly a multiprocessor)
- One operating system is in control
- Threats come from multiple users
  - or from external access

What Needs to Be Protected?

- Memory
- Files
- Processes and IPC
- I/O and devices

Basic Methods of Protection

- All based on *separation*
  - Physical separation
  - Temporal separation
  - Logical separation
  - Cryptographic separation
- Often in combination

Protecting Memory

- Virtual memory offers strong protection tools
  - model prevents naming another user's memory
- What about shared memory?
  - use access control mechanisms
  - backed up by hardware protection on pages

Protecting Files

- Unlike memory, files are in a <u>shared</u> namespace
- Requires more use of access controls
- Typically, access checked (only) on open
  - system assumes user has right to continue using open file
  - why is it done this way?

File Access Control in Unix

- Every file has an owning user and group
- Access permissions settable for read, write, and execute
- For owning user, owning group, everyone else
- Processes belong to one user
  - and possibly multiple groups
- Files opened for particular kinds of access

Protecting Processes

- Most of a process' state not addressable externally
- But IPC channels allow information to flow
- So security must be applied at IPC points

Protecting IPC

- Typically, IPC requires cooperation from both ends
- So a major question is authentication
  - "Does this channel connect to where I think it does?"
- OS guarantees identity, ownership of other process

Limiting IPC Access

- Each party to IPC has control over what is done on his side
- Some IPC mechanisms allow differing modes of access for different users
- So access control required for such cases

Protecting Devices

- Generally treated similarly to files
- But special care is necessary
  - in some cases, a mistake allows an intruder unlimited access
  - e.g., if you let him write any block on a disk drive

Beware of Back Doors

- Many systems provide low-level ways to access various resources
  - /dev/kmem
  - raw devices
  - pipes stored in the file system
- The lock on the back door must be as strong as the lock on the front door

Intruders

- Modern systems usually allow remote access
  - from terminals
  - from modems
  - from the network
- Intruders can use all of these to break in

How Intruders Get in

- Often by masquerading as a legitimate user
- Less frequently by inserting commands through insecure entry points
  - finger daemons
  - holes in electronic mail
  - making use of interpreters that access data remotely

Detecting Intruders

- The sooner detected, the better
- Systems that detect and eject intruders quickly are less attractive targets
- Information gained from detecting intruders can be used to prevent further intrusions
- Detection presumes you can differentiate the behavior of authorized users and intruders

Some Approaches to Detecting Intruders

- Statistical anomaly detection; based on
  - Overall system activity
  - Individual user profiles
- Rule-based detection
  - Rules that detect anomalies
  - Penetration expert systems

Audit Records

- Keep track of everything done on system
- Powerful tool for detecting intruders
- Used to build detection mechanisms
- Can use either general accounting info or specially gathered data
- Also invaluable if you decide to prosecute
- Must be carefully protected to be valuable

Malicious Programs

- Clever programmers can get software to do their dirty work for them
- Programs have several advantages for these purposes
  - Speed
  - Mutability
  - Anonymity

Kinds of Malicious Programs

- Trojan horses
- Trapdoors
- Logic bombs
- Worms
- Viruses

Trojan Horses

- Seemingly useful program that contains code that does harmful things
- Unsuspecting users run the Trojan horse to get the advertised benefit
  – at which time the Greeks spring out and slaughter your system
- Particularly dangerous in compilers

Trapdoors

- A secret entry point into an otherwise legitimate program
- Typically inserted by the writer of the program
- Most often found in login programs or programs that use the network
- But also found in system utilities

Logic Bombs

- Like trapdoors, typically in a legitimate program
- A piece of code that, under certain conditions, “explodes”
- Also like trapdoors, typically inserted by program authors

Worms

- Programs that seek to move from system to system
  – making use of various vulnerabilities
- Other malicious behavior can be built-in
- The Internet worm is the most famous example
- Can spread very, very rapidly

Viruses

- A program that can infect other programs
- Infected programs in turn infect others
- Along with mere infection, Trojan horses, trapdoors, or logic bombs can be included
- Like worms, viruses can spread very rapidly

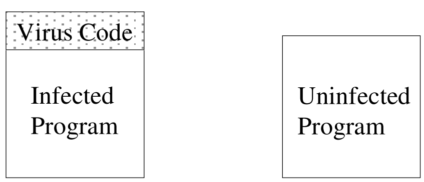
How Do Viruses Work?

- When a program is run, it typically has the full privileges of its running user
- Including write privileges for some other programs
- A virus can use those privileges to replace those programs with infected versions

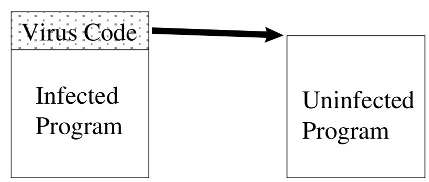
Typical Virus Actions

1) Find uninfected writable programs
2. Modify those programs
3) Perform normal actions of infected program
4) Do whatever other damage is desired

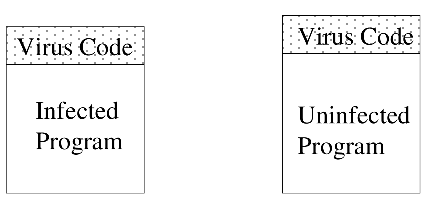
Before the Infected Program Runs



The Infected Program Runs



Infecting the Other Program



How Do Viruses Fit Into Programs?

- Prepended
- Postpended
- Copy program and replace
- Cleverly fit into the cracks

Some viruses take other measures to hide modifications

Where Can Viruses Occur?

Anywhere that imported code gets executed

- Imported programs
- Some inclusions in mail messages
- Boot sectors and other executable portions of media
- Macros attached to some data files

Dealing With Viruses

- Prevention of infection
- Detection and eradication
- Containment

Preventing the Spread of Viruses

- Don't import untrusted programs
  - but who can you trust?
- Viruses have been found in commercial shrink-wrap software
- Trusting someone means not just trusting their honesty, but also their caution

Other Prevention Measures

- Scan incoming programs for viruses
  - some viruses are designed to hide
- Limit the targets viruses can reach
- Monitor updates to executables carefully
  - requires a broad definition of "executable"

Virus Detection

- Many viruses have detectable signatures
  - but some work hard to hide them
- Smart scanners can examine programs for virus-like behavior
- Checksums attached to programs can detect modification
  - if virus smart enough to generate checksum itself, digitally sign it

Virus Eradication

- Tedious, because you must be thorough
- Restore clean versions of everything
- Take great care with future restoration of backups

Containment

- Run suspect programs in an encapsulated environment
  - limiting their forms of access to prevent virus spread
- Requires versatile security model and strong protection guarantees

Summary

- Basic security requirements include…
- Basic threats include…
- Security design principles include…
- Security mechanisms include…
- Single system security protects… by ...
- Malicious programs include…
- Viruses work by ...