Operating System Security - Basic Principles

CS239
Advanced Operating Systems

November 19, 1997
Dr. Richard Guy

Lecture Source Materials

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

Outline

- Introduction
- Security threats
- Basic security principles
- Security on a single machine

CS239, Winter98 will add...

- Encapsulated environments
- Encryption and authentication
- Distributed systems security

Introduction

Security is an engineering problem
Always a tradeoff between safety, cost, and inconvenience
Not much solid theory in the field
Hard to provide any real guarantees
Because making mistakes is easy
And the nature of the problem implies that mistakes are always exploited

History of the Security Problem

In the beginning, there was no security problem

Later, there was a problem, but nobody cared

Now, there are increasing problems, and people are beginning to care

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 costs
  - If it costs too much, it won't be used
- If it isn't easy, it won't be used
- Misuse often makes security measures useless
- Fit the stringency of the measures to the threat being countered

Security is as Strong as the Weakest Link

Those breaking security will attack the weakest point

Putting an expensive lock on a cheap door doesn't help much

Must look on security problems as part of an integrated system, not just a single component

Security Threats

Extremely wide range of threats

From a wide variety of sources

Requiring a wide variety of countermeasures

Generally, countering any threat costs something

So people frequently try to counter as few as they can afford

Physical Security

Some threats involve access to the equipment itself

Such as

- 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

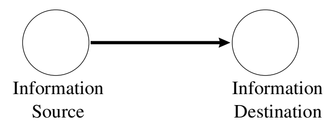
- Secrecy
- Integrity
- Availability
- Exclusivity

A Classification of Threats

Threats are viewed as types of attacks on normal service

So what is normal service?

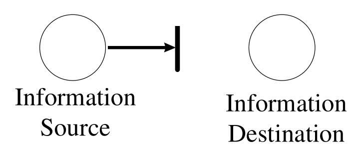
•



Classification of Threat Types

- Interruption
- Interception
- Modification
- Fabrication

Interruption



Interruption Threats

Denial of service

Prevents source from sending information to receiver

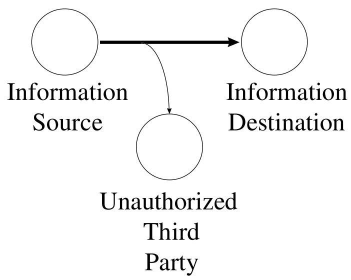
Or receiver from sending requests to source

A threat to availability

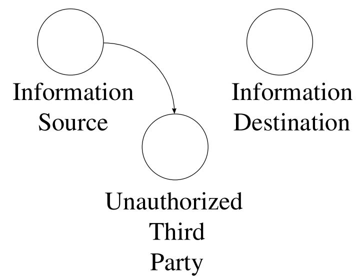
How Does an Interruption Threat Occur?

- 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

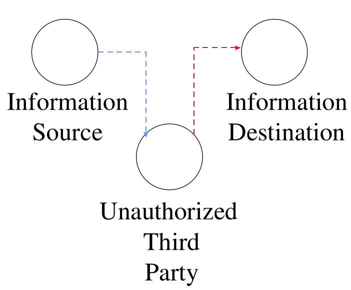
A threat to secrecy

Also a threat to exclusivity

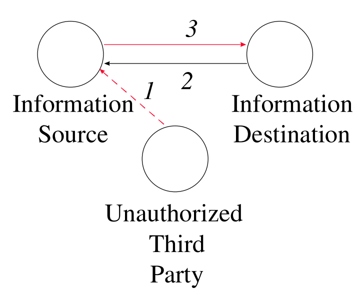
How Do Interception Threats Occur?

- 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

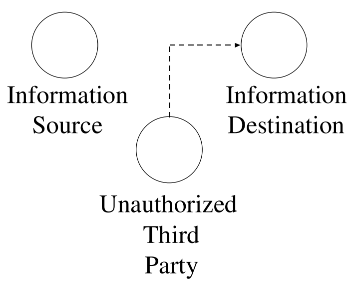
Or permanently at the servers

A threat to integrity

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

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, etc. occur

Active threats are more aggressive

Passive threats are mostly to secrecy

Active threats are to availability, integrity, exclusivity

What Are We Protecting?

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

Basic Security Principles

- Terms and concepts
- Mechanisms

Security and Protection

Security is a policy

E.g., “no unauthorized user may access this file”

Protection is a mechanism

E.g., “the system checks user identity against access permissions”

Protection mechanisms implement security policies

Design Principles for Secure Systems

- Economy
- Complete mediation
- Open design
- 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 every access to an object that a mechanism is meant to protect

E.g., each read of a file, not just the open

Also involves checking access on everything 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

To allow 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 him

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 memory, files, processes, and devices
- 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 shared namespace

Requires more use of access controls

Typically, access checked 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

Usually 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 either -

    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 also 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

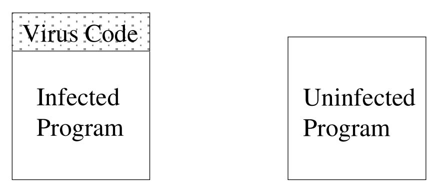
1). Find uninfected writable programs

2). Modify those programs

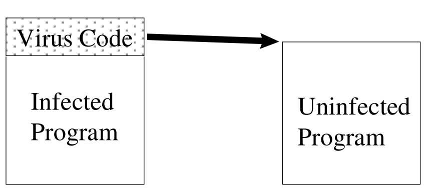
3). Perform normal actions of infected program

4). Do whatever other damage is desired

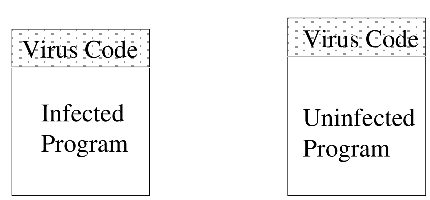
Before the Infected Program Runs



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