Operating System Security, Con’t

CS 239, Spring 1995

April 27, 1995

Operating System Security,Con’t

- Single system security
  - Memory, files, processes, and devices
  - Dealing with intruders
  - Malicious programs
- Distributed system security
  - Using encryption
  - Secure distributed applications

Single System Security

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

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

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 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 of the communications

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

Controlling IPC Access in Windows NT

- General model related to file access control
- Processes try to access objects
  - Objects include IPC entities
- On first access, request desired access rights
  - Set of granted access rights returned
- System checks granted access rights on each attempted access

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

- 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

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

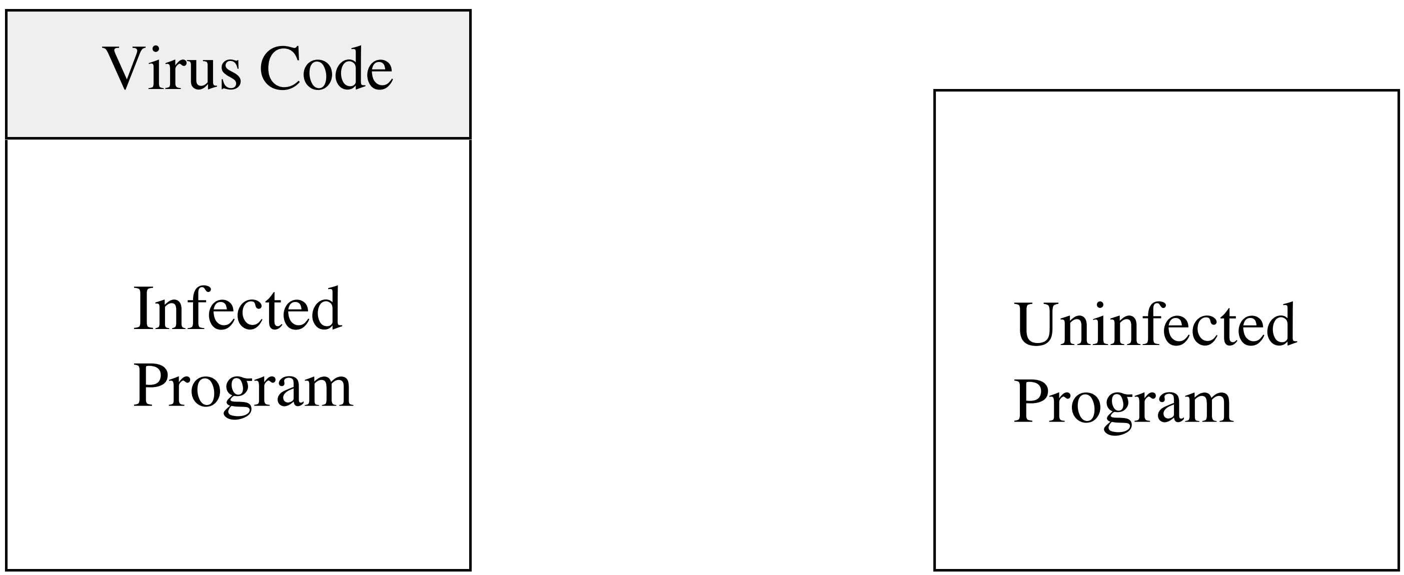
Including write privileges for some other programs

A virus can use those privileges to replace those programs with infected versions

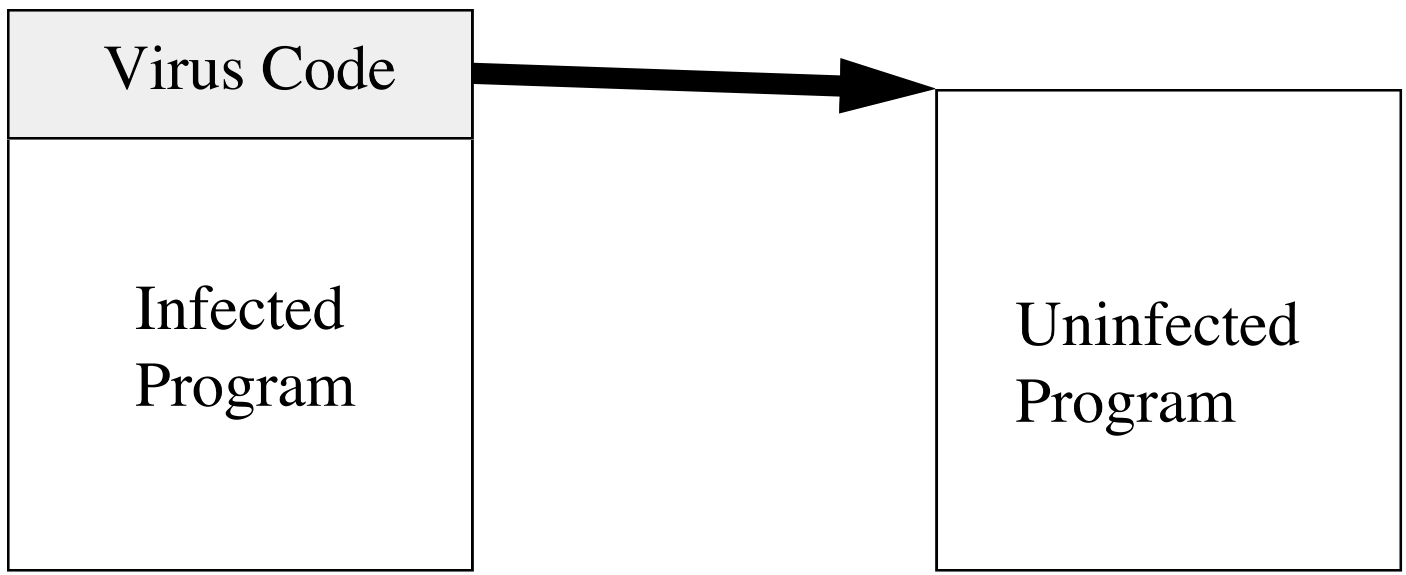
Typical Virus Actions

- Find uninfected writable programs
- Modify those programs
- Perform normal actions of infected program
- Do whatever other damage is desired

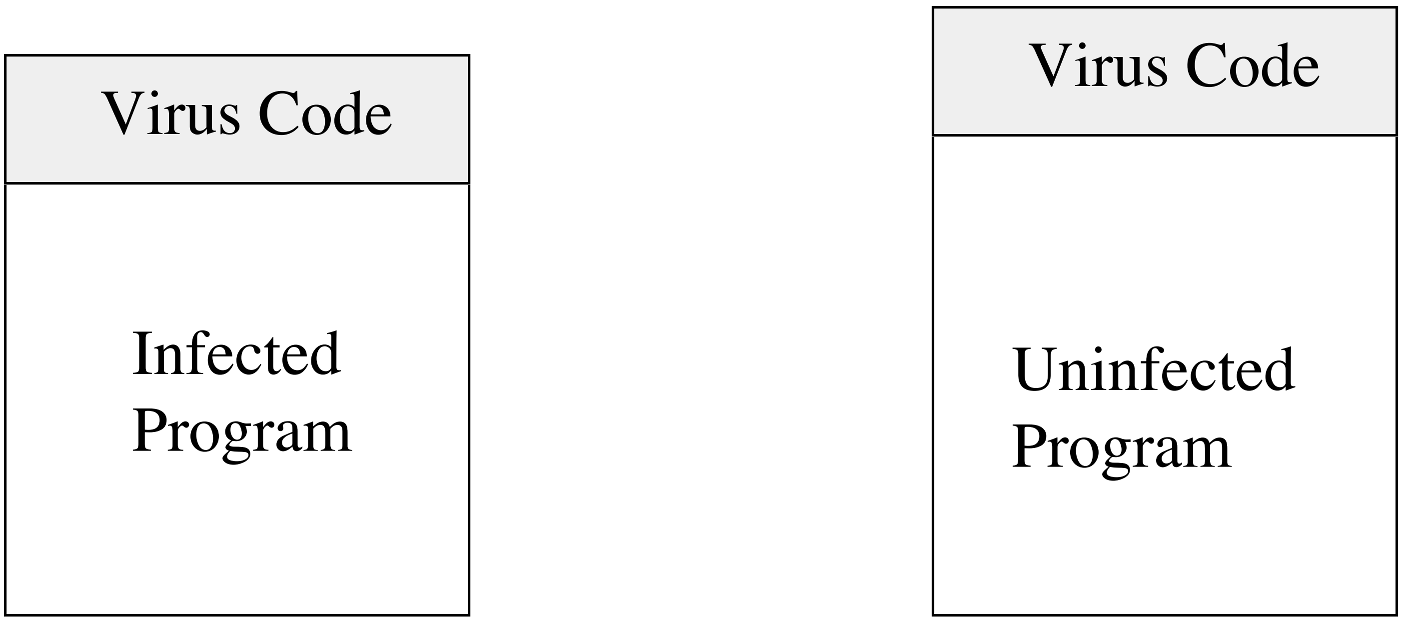
Before the Infected Program Runs



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

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 shrinkwrap 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

Security in Distributed Systems

A substantially harder problem

Many single-system mechanisms are based on trusting a central operating system

Single-system mechanisms often assume secure communications channels

Single-system mechanisms can (in principle) have access to all relevant data

Security Mechanisms for Distributed Systems

- Encryption
- Authentication
- Firewalls

Encryption for Distributed Systems

Can protect secrecy of data while on insecure links

Can also prevent modification and many forms of fabrication attacks

But keys are a tricky issue

Encryption Keys and Distributed System Security

To gain benefit from encryption, communicating entities must share a key

Each separate set of entities need a different key

How do you securely distribute keys?

Key Distribution Schemes

- Manual distribution by one party
- Manual distribution by third party
- Use existing key to send new key
- Key servers

Key Servers

- Trusted third party that can provide good keys on demand
- Typically on a separate machine
- Tremendous care must be taken to ensure secure communications with the key server

Authentication For Distributed Systems

When a message comes in over the net, how do you tell who sent it?

Generally with some form of *digital signature*

- Must be unique to signing user
- And also unique to the message

Often using *public key cryptography*

Authentication Servers

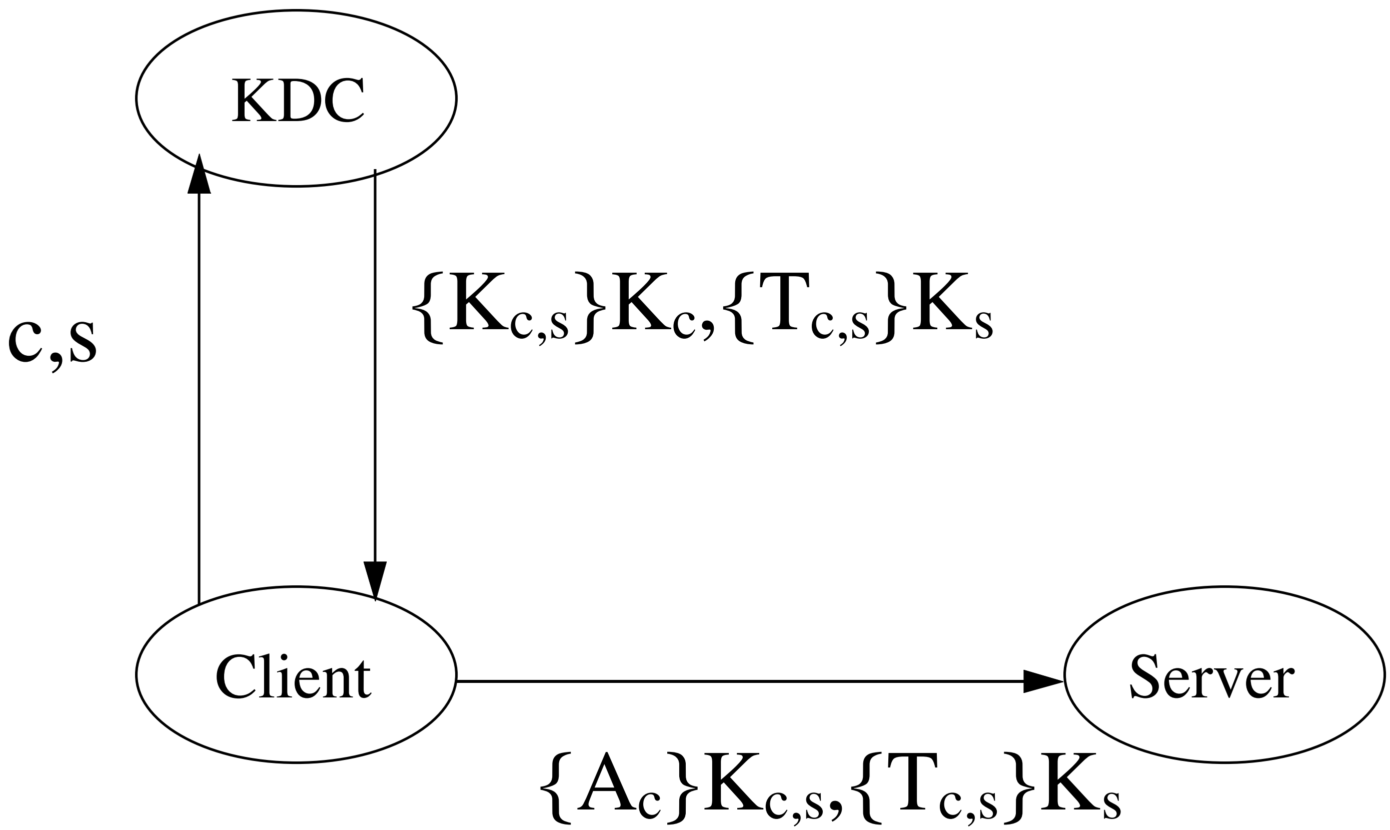
Like key servers, trusted third parties

An authentication server can produce a ticket that guarantees the identity of a user

Generally tickets expire

Kerberos is the most popular authentication server

Kerberos in Action



Firewalls

- A program to allow selective access to the network
  - In both directions
- Typically, firewalls protect entire networks
- They must examine everything that tries to pass into the protected domain
- Only authorized transmissions permitted

Firewall Example