Malicious Code
CS 239
Computer Software
March 5, 2003

Outline

- Introduction
- Viruses
- Trojan horses
- Trap doors
- Logic bombs
- Worms
- Examples

Introduction

Clever programmers can get software to do their dirty work for them

Programs have several advantages for these purposes
- Speed
- Mutability
- Anonymity

Where Does Malicious Code Come From?

- Often, it's willingly (but unwittingly) imported into the system
  - Electronic mail (common today)
  - Floppy disks
  - Downloaded executables
  - Sometimes shrinkwrapped software
- Sometimes it breaks in
- Sometimes an insider intentionally introduces it

Is Malicious Code Really a Problem?

- Considering viruses only, by 1994 there were over 1,000,000 annual infections
- Recently, a company discovered that 1 in 325 incoming email messages carried a malicious attachment
- 2002 FBI report shows ~75% of survey respondents had financial losses due to malicious code

More Alarming Statistics

- In 1992, there were around 2000 unique viruses known
- Today, McAfee's databases of viruses includes 60,000+ entries[1]
- The numbers continue to grow

[1] http://vil.mcafee.com

But Don't Get Too Alarmed

- Most viruses are never found "in the wild"
- Most viruses die out quickly
- The Wild List[1] shows 205 active viruses worldwide
  - With another 530 or so with only a single incident reported

[1] www.wildlist.org

How Much Do Viruses Cost?

- The Code Red virus estimated to have cost companies $1 billion
  - Counting clean up, productivity losses, etc.
- Even if it's two orders of magnitude off, that's serious money
- Computer Economics estimates economic impact of viruses in 2001 ~ $13.2 billion
  - But many folks think CE is full of it

But Do **I** Really Have to Worry About Viruses?

- "After all, I run Linux/Mac OS/Solaris/BSD"
- "Aren't all viruses for Windows?"
- Mostly true in practice
  - Definitely not true in theory
- Anyone, at any time, can write and release a virus that can clobber your machine, regardless of what OS you run

Viruses

- "Self-replicating programs containing code that explicitly copies itself and that can 'infect' other programs by modifying them or their environment"
- Typically attached to some other program
  - When that program runs, the virus becomes active and infects others
- Not all malicious codes are viruses

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

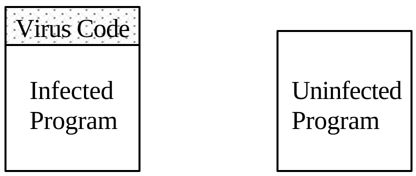
1). Find uninfected writable programs

2). Modify those programs

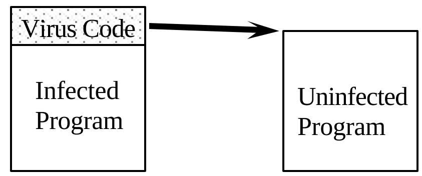
3). Perform normal actions of infected program

4). Do whatever other damage is desired

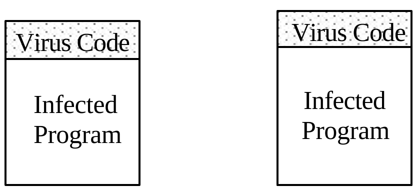
Before the Infected Program Runs



The Infected Program Runs



Infecting the Other Program



A Taxonomy of Viruses

- File infectors
  - Direct-action
  - Resident
- System or boot infectors
- Macro and attachment viruses

File Infectors

- File infector viruses attach themselves to ordinary files
  - Most typically executables
  - But source code viruses have been discovered

Direct-Action File Infectors

- The most basic kind of virus
- When an infected program runs, it infects a previously uninfected target
  - Or possibly several
- Activated only when an infected program is executed
- E.g., the Vienna virus

Resident File Infectors

- The first time a resident file infector is run, it leaves a daemon running
  - Or some other persistent and useful RAM entity
- Whenever any other program is run, the daemon infects it
- E.g., the Jerusalem virus

System Infectors

- Also known as boot section infectors
- Rather than living in programs, they live in the boot sectors of disks
- Typically memory resident
- Typically infect any disk exposed to them
- E.g., the Michelangelo virus
- Multipartite viruses can live in both programs and boot sectors

Macro and Attachment Viruses

- At first, viruses only attacked executables
- But sophisticated modern data files often contain executable components
  - Macros
  - Email attachments
  - Ability to run arbitrary executables from many applications, embedded in data
- Easily the most popular form of new viruses
  - Requires less sophistication to get right

Virus Toolkits

- Helpful hackers have written toolkits that make it easy to create viruses
- A typical smart high school student can easily create a virus given a toolkit
- Generally easy to detect viruses generated by toolkits
  - But we may see “smarter” toolkits

How To Find Viruses

- Basic precautions
- Looking for changes in file sizes
- Scan for signatures of viruses
- TSR monitoring
- Multi-level generic detection

Precautions to Avoid Viruses

- Don’t import untrusted programs
  - But who can you trust?
- Viruses have been found in commercial shrink-wrap software
- The hackers who released Back Orifice were embarrassed to find a virus on their CD release
- Trusting someone means not just trusting their honesty, but also their caution

Other Precautionary 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"

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

Viruses and File Sizes

- Typically, a virus tries to hide
- So it doesn't disable the infected program
- Instead, extra code is added
- But if it's added naively, the size of the file grows
- Virus detectors can look for this growth

Problems With Size Checking for Virus Detection

- Requires keeping carefully protected records of valid file sizes
- Won't work for files whose sizes typically change
  - E.g., Word files with possibly infected macros
- Clever viruses find ways around it
  - E.g., cavity viruses that fit themselves into "holes" in programs

Signature Scanning

- If a virus lives in code, it must leave some traces
- In early and unsophisticated viruses, these traces were essentially characteristic code patterns
- Find the virus by looking for the signature

How To Scan For Signatures

- Create a database of known virus signatures
- Read every file in the system and look for matches in its contents
- Also check every newly imported file
- Also scan boot sectors and other interesting places

Weaknesses of Scanning for Signatures

- What if the virus changes its signature?
- What if the virus takes active measures to prevent you from finding the signature?
- You can only scan for known virus signatures

Polymorphic Viruses

- A polymorphic virus produces varying but operational copies of itself
- Essentially avoiding having a signature
- Sometimes only a few possibilities
  – E.g., Whale virus has 32 forms
- But sometimes a lot

Stealth Viruses

- A virus that tries actively to hide all signs of its presence
- Typically a resident virus
- For example, it traps calls to read infected files
  – And disinfects them before returning the bytes
  – E.g., the Brain virus

Combating Stealth Viruses

- Stealth viruses can hide what’s in the files
- But may be unable to hide that they’re in memory
- Also, if you reboot carefully from a clean source, the stealth virus can’t get a foothold

TSR Monitoring

- TSR - Terminate-and-Stay-Resident
  – Essentially a daemon process
- A virus detector that runs in the background
- Automatically scans (and possibly takes other actions) continuously

Other TSR Monitor Actions

- Signature scanning can’t find new viruses
- Watching system activity for suspicious actions possibly can
- A TSR monitor can run intrusion detection systems or other code to catch new viruses

Multi-Level Generic Detection

- Virus detection software that is specialized to handle both known and new viruses
- Using a combination of methods
- Both continuously and on command

Generic Detection Tools

- Checksum comparison
- Intelligent checksum analysis
  - For files that might legitimately change
- Intrusion detection methods
  - More sophisticated than intelligent checksum analysis
  - Possibly very high overhead

Preventing Virus Infections

- Run a virus detection program
- Keep its signature database up to date
- Disable program features that run executables without users asking
- Make sure users are very careful about what they run

How To Deal With Virus Infections

- Reboot from a clean, write-protected floppy or from a clean CD ROM
  - Important to ensure that the medium really is clean
  - Necessary, but not sufficient
- If backups are available and clean, replace infected files with clean backup copies
  - Another good reason to keep backups

Disinfecting Programs

- Some virus utilities try to disinfect infected programs
  - Allowing you to avoid going to backup
- Potentially hazardous, since they may get it wrong
  - Some viruses destroy information needed to restore programs properly

Trojan Horses

- Seemi... contai... ings
- When Greeks slaught...

Basic Trojan Horses

- A program you pick up somewhere that is supposed to do something useful
- And perhaps it does
  – But it also does something less benign
- Games are common locations for Trojan Horses
- Downloaded applets are increasingly popular locations
- Recently popping up in email attachments

Trojan Horse Login Programs

- Probably the original Trojan horse
- Spoof the login or authentication screen of a machine or service
- Capture attempts to access that service
- Then read the user IDs and the passwords

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
- Often used by disgruntled employees to get revenge

Worms

- Programs that seek to move from system to system
  – Making use of various vulnerabilities
- Other performs other malicious behavior
- The Internet worm used to be the most famous example
  – Sapphire/Slammer is also a worm
- Can spread very, very rapidly

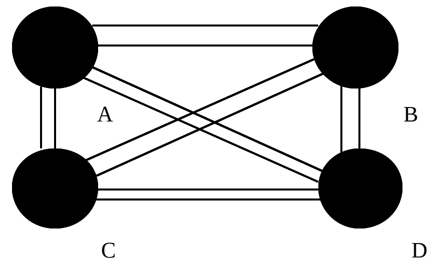
The Internet Worm

- Created by a graduate student at Cornell in 1988
- Released (perhaps accidentally) on the Internet Nov. 2, 1988
- Spread rapidly throughout the network
  – 6000 machines infected

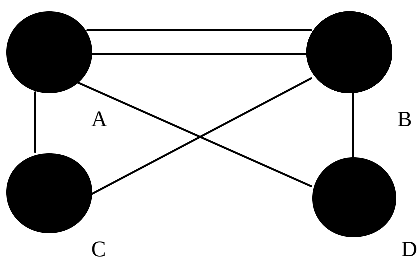
The Effects of the Worm

- Essentially, affected systems ended up with large and increasing numbers of processes devoted to the worm
- Eventually all processes in the process table used up
- Rebooting didn't help, since other infected sites would immediately re-infect the rebooted machine

A Visual Picture of the Infection



And What If Someone Reboots?



How Did the Internet Worm Work?

- The worm attacked network security vulnerabilities in one class of OS
  - Unix 4 BSD variants
- These vulnerabilities allowed improper execution of remote processes
- Which allowed the worm to get a foothold on a system

The Worm's Actions on Infecting a System

- Find an uninfected system and infect that one
- Using the same vulnerabilities
- Here's where it ran into trouble:
  - It re-infected already infected systems
  - Each infection was a new process

The Worm's Breaking Methods

- rsh - if the remote host is on the trusted hosts lists, simply rsh'ing could work
- fingerd - exploit a bug in the fingerd program to overwrite a buffer in a useful way
- sendmail - invoke a debugging option in sendmail and issue commands

What Didn't the Worm Do?

- It didn't attempt to intentionally damage a system
- It didn't attempt to divulge sensitive information (e.g., passwords)
- It didn't try hard to become root
  - And didn't exploit root access if it got superuser access

Stopping the Worm

- In essence, required rebooting all infected systems
  - And not bringing them back on the network until the worm was cleared out
  - Though some sites stayed connected
- Also, the flaws it exploited had to be patched

Effects of the Worm

- Around 6000 machines were infected and required substantial disinfecting activities
- Many, many more machines were brought down or pulled off the net
  - Due to uncertainty about scope and effects of the worm

How Much Did the Worm Cost?

- Hard to quantify
  - Typical for costs of computer attacks
- Estimates as high as $98 million
  - Probably overstated, but certainly millions in down time, sysadmin and security expert time, and costs of disconnections

What Did the Worm Teach Us?

- The existence of some particular vulnerabilities
- The costs of interconnection
- The dangers of being trusting
- Denial of service is easy
- Security of hosts is key
- Logging is important
- We obviously didn't learn enough

The Love Bug

- Virus, worm, or Trojan Horse?
- Some of each, really
- Very wide spread
  - Proportionally smaller than Internet worm, but bigger total numbers
- Arrived in email posing as a love letter

How the Love Bug Worked

- Passed primarily through email
- Contained an attachment with a VSBasic script (Trojan Horse)
- When attachment was opened, script infected various local files (virus)
- Also tried to spread via email (worm)
- Other than destroying files while infecting them, didn't seek to harm system
- But did try to steal passwords

The Love Bug and Microsoft Outlook

- The Love Bug was built to exploit systems using Outlook
- Used poor security default in Outlook
  - In some versions, attachments opened without user request
- Used knowledge of Outlook address book to find new victims

Why Did the Love Bug "Succeed"?

- Not especially sophisticated
- Didn't introduce any new methods
- Didn't exploit any new vulnerabilities
- Apparently more successful than its competitors due to social engineering
  - Lots of people were likely to open a supposed love letter
- And aggression
  - Sent to everyone in address book, not just some

Code Red

- A malicious worm that attacked Windows machines
- Basically used vulnerability in Microsoft IIS servers
- Became very widely spread and caused a lot of trouble

How Code Red Worked

- Attempted to connect to TCP port 80 (a web server port) on randomly chosen host
- If successful, sent HTTP GET request designed to cause a buffer overflow
- If successful, defaced all web pages requested from web server

More Code Red Actions

- Periodically, infected hosts tried to find other machines to compromise
- Triggered a DDoS attack on a fixed IP address at a particular time
- Actions repeated monthly
- Possible for Code Red to infect a machine multiple times simultaneously

Code Red Stupidity

- Bad method used to choose another random host
  - Same random number generator seed to create list of hosts to probe
- DDoS attack on a particular fixed IP address
  - Merely changing the target's IP address made the attack ineffective

Code Red II

- Used smarter random selection of targets
- Didn't try to reinfect infected machines
- Adds a Trojan Horse version of Internet Explorer to machine
  - Unless other patches in place, will reinfect machine after reboot on login
- Also, left a backdoor on some machines
- Doesn't deface web pages or launch DDoS

A Major Difference

- Code Red periodically turns on and tries to infect again
- Code Red II worked intensively for 24-48 hours after infection
  - Then stopped
- Eventually, Code Red II infected all infectable machines
  - Some are still infected, but they've stopped trying to spread it

Impact of Code Red and Code Red II

- Code Red infected over 250,000 machines
- In combination, estimated infections of over 750,000 machines
- Code Red II is essentially dead
- But Code Red is still out there

A Bad Secondary Effect of Code Red

- Generates lots of network traffic
- U. of Michigan study found 40 billion attempts to infect 8 fake "machines" per month
  - Each attempt was a packet
  - So that's ~1 billion packets per day just for those eight addresses
- "The new Internet locust[1]"

[1] Farnham Jahanian, talk at DARPA FTN meeting, Jan 18, 2002

Virus Hoaxes

- Virus hoaxes are at least as common as real viruses
- Generally arrive in email
- Usually demand instant action, on pain of something really terrible
- It's wise to check with a reliable source before taking action on such email messages
  - Or forwarding them