Last time

- Program security
- Flaws, faults, and failures
- Types of security flaws
- Unintentional flaws
  - Buffer overflows
  - Incomplete mediation
  - TOCTTOU errors

This time

- Finish TOCTTOU
- Malicious code: Malware
  - Viruses
  - Trojan horses
  - Logic bombs
  - Worms
- Other malicious code: web bugs

TOCTTOU errors

- TOCTTOU (“TOCK-too”) errors
  - Time-Of-Check To Time-Of-Use
  - Also known as “race condition” errors
- These errors occur when the following happens:
  - User requests the system to perform an action
  - The system verifies the user is allowed to perform the action
  - The system performs the action

Example

- A particular Unix terminal program is setuid (runs with superuser privileges) so that it can allocate terminals to users (a privileged operation)
- It supports a command to write the contents of the terminal to a log file
- It first checks if the user has permissions to write to the requested file; if so, it opens the file for writing
- The attacker makes a symbolic link:
  `logfile -> file_she_owns`
- Between the “check” and the “open”, she changes it:
  `logfile -> /etc/passwd`

The problem

- The state of the system changed between the check for permission and the execution of the operation

- The file whose permissions were checked for writeability by the user (`file_she_owns`) wasn't the same file that was later written to (`/etc/passwd`)
  - Even though they had the same name (`logfile`) at different points in time

- Q: Can the attacker really “win this race”?
- A: Yes.

Defences against TOCTTOU errors

- When performing a privileged action on behalf of another party, make sure all information relevant to the access control decision is constant between the time of the check and the time of the action (“the race”)
  - Keep a private copy of the request itself so that the request can't be altered during the race
  - Where possible, act on the object itself, and not on some level of indirection
    - e.g. Make access control decisions based on filehandles, not filenames
  - If that's not possible, use locks to ensure the object is not changed during the race

Malware

- We've looked at some security flaws that can arise due to programmers and designers not being careful enough
- This week, we will look at active and malicious attacks that can cause security failures
- We will start with malware: various forms of software written with malicious intent
  - Viruses
  - Trojan horses
  - Logic bombs
  - Worms

Malware

- A common characteristic of all types of malware is that it needs to be executed in order to cause harm
- How might malware get executed?
  - User action
    - Downloading and running malicious software
    - Viewing a web page containing a malicious ActiveX control
    - Opening an executable email attachment
    - Inserting a CD
  - Exploiting an existing flaw in a system
    - Buffer overflows in network daemons
    - Buffer overflows in email clients or web browsers

Viruses

- A virus is a particular kind of malware that infects other files
  - Traditionally, a virus could only infect executable programs
  - Nowadays, many data document formats can contain executable code (such as macros)
    - Many different types of files can be infected with viruses now
- Typically, when the file is executed (or sometimes just opened), the virus activates, and tries to infect other files with copies of itself
- In this way, the virus can spread between files, or between computers

Infection

- What does it mean to “infect” a file?
- The virus wants to modify an existing (non-malicious) program or document (the host) in such a way that executing or opening it will transfer control to the virus
  - The virus can do its “dirty work” and then transfer control back to the host
- For executable programs:
  - Typically, the virus will modify other programs and copy itself to the beginning of the targets' program code
- For documents with macros:
  - The virus will edit other documents to add itself as a macro which starts automatically when the file is opened

Infection

- In addition to infecting other files, a virus will often try to infect the computer itself
  - This way, every time the computer is booted, the virus is automatically activated
- It might put itself in the boot sector of the hard disk
- It might add itself to the list of programs the OS runs at boot time
- It might infect one of more of the programs the OS runs at boot time
- It might try many of these strategies
  - But it's still trying to evade detection!

Spreading

- How do viruses spread between computers?
- Usually, when the user sends infected files (hopefully not knowing they're infected!) to his friends
  - Or puts them on a p2p network
- A virus usually requires some kind of user action in order to spread to another machine
  - If it can spread on its own (via email, for example), it's more likely to be a worm than a virus

Payload

- In addition to trying to spread, what else might a virus try to do?
- Some viruses try to evade detection by disabling any active virus scanning software
- Most viruses have some sort of payload
- At some point, the payload of an infected machine will activate, and something (usually bad) will happen
  - Erase your hard drive
  - Subtly corrupt some of your spreadsheets
  - Install a keystroke logger to capture your online banking password
  - Start attacking a particular target website

Spotting viruses

- When should we look for viruses?
  - As files are added to our computer
    - Via portable media
    - Via a network
  - From time to time, scan the entire state of the computer
    - To catch anything we might have missed on its way in
    - But of course, any damage the virus might have done may not be reversable

- How do we look for viruses?
  - Signature-based protection
  - Behaviour-based protection

Signature-based protection

- Keep a list of all known viruses
- For each virus in the list, store some characteristic feature (the signature)
  - Most signature-based systems use features of the virus code itself
    - The infection code
    - The payload code
  - Can also try to identify other patterns characteristic of a particular virus
    - Where on the system it tries to hide itself
    - How it propagates from one place to another

Polymorphism

- To try to evade signature-based virus scanners, some viruses are polymorphic
  - This means that instead of making perfect copies of itself every time it infects a new file or host, it makes a modified copy instead
  - This is often done by having most of the virus code encrypted
    - The virus starts with a decryption routine which decrypts the rest of the virus, which is then executed
    - When the virus spreads, it encrypts the new copy with a newly-chosen random key
- How would you scan for polymorphic viruses?

Behaviour-based protection

- Signature-based protection systems have a major limitation
  - You can only scan for viruses that are in the list!
  - But there are several brand-new viruses identified every day
    - Currently 70-75 thousand
  - What can we do?

- Behaviour-based systems look for suspicious patterns of behaviour, rather than for specific code fragments
  - Of course, this is only useful post-infection

False negatives and positives

- Any kind of test or scanner can have two types of errors:
  - False negatives: fail to identify a threat that is present
  - False positives: claim a threat is present when it is not
- Which is worse?
- How do you think signature-based and behaviour-based systems compare?

Trojan horses

- Trojan horses are programs which claim to do something innocuous (and usually do), but which also hide malicious behaviour

You're surfing the Web and you see a button on the Web site saying, "Click here to see the dancing pigs." And you click on the Web site and then this window comes up saying, "Warning: this is an untrusted Java applet. It might damage your system. Do you want to continue? Yes/No." Well, the average computer user is going to pick dancing pigs over security any day. And we can't expect them not to.
-- Bruce Schneier

Trojan horses

- Trojan horses:
  - Gain control by getting the user to run code of the attacker's choice, usually by also providing some code the user *wants* to run
    - “PUP” (potentially unwanted programs) are an example
  - The payload can be anything; sometimes the payload of a Trojan horse is itself a virus, for example
  - Trojan horses usually do not themselves spread between computers; they rely on multiple users executing the “trojaned” software
    - Better: users share the trojaned software on p2p networks

Logic bombs

- A logic bomb is malicious code hiding in the software already on your computer, waiting for a certain trigger to "go off" (execute its payload)
- Logic bombs are usually written by "insiders", and are meant to be triggered sometime in the future
  - After the insider leaves the company
- The payload of a logic bomb is usually pretty dire
  - Erase your data
  - Corrupt your data
  - Encrypt your data, and ask you to send money to some offshore bank account in order to get the decryption key!

Logic bombs

- What is the trigger?
- Usually something the insider can affect once he is no longer an insider
  - Trigger when this particular account gets three deposits of equal value in one day
  - Trigger when a special sequence of numbers is entered on the keypad of an ATM
  - Just trigger at a certain time in the future (called a “time bomb”)

Spotting Trojan horses and logic bombs

- Spotting Trojan horses and logic bombs is extremely tricky. Why?

- The user is intentionally running the code!
  - Trojan horses: the user clicked “yes, I want to see the dancing pigs”
  - Logic bombs: the code is just (a hidden) part of the software already installed on the computer

- Don't run code from untrusted sources?
- Better: prevent the payload from doing bad things
  - More on this later

Worms

- A worm is a self-contained piece of code that can replicate with little or no user involvement
- Worms often use security flaws in widely deployed software as a path to infection
- Typically:
  - A worm exploits a security flaw in some software on your computer, infecting it
  - The worm immediately starts searching for other computers (on your local network, or on the Internet generally) to infect
  - There may or may not be a payload that activates at a certain time, or by another trigger

The Morris worm

- The first Internet worm, launched by a graduate student at Cornell in 1988
- Once infected, a machine would try to infect other machines in three ways:
  - Exploit a buffer overflow in the “finger” daemon
  - Use a back door left in the “sendmail” mail daemon
  - Try a “dictionary attack” against local users' passwords. If successful, log in as them, and spread to other machines they can access without requiring a password
- All three of these attacks were well known!
- Thousands of systems were offline for several days

The Code Red worm

- Launched in 2001
- Exploited a buffer overflow in Microsoft's IIS web server (for which a patch had been available for a month)
- An infected machine would:
  - Deface its home page
  - Launch attacks on other web servers (IIS or not)
  - Launch a denial-of-service attack on a handful of web sites, including www.whitehouse.gov
  - Installed a back door and a Trojan horse to try to prevent disinfection
- Infected 250,000 systems in nine hours

The Slammer worm

- Launched in 2003
- First example of a “Warhol worm”
  - A worm which can infect nearly all vulnerable machines in just 15 minutes
- Exploited a buffer overflow in Microsoft's SQL Server (also having a patch available)
- A vulnerable machine could be infected with a single UDP packet!
  - This enabled the worm to spread extremely quickly
  - Exponential growth, doubling every 8.5 seconds
  - 90% of vulnerable hosts infected in 10 minutes

Other malicious code: web bugs

- A web bug is an object (usually a 1x1 pixel image) embedded in a web page, which is fetched from a different server than the one that served the web page itself.

- Information about you can be sent to third parties (often advertisers) without your knowledge or consent
  - IP address
  - Contents of cookies (to link cookies across web sites)
  - Any personal info the site has about you

Web bug example

- On the quicken.com home page:
  - `<IMG WIDTH="1" HEIGHT="1" src="http://app.insightgrit.com/1/nat?id=79152388778&ref=http://www.eff.org/Privacy/Marketing/web_bug.html&z=668951&purl=http://quicken.intuit.com/?src=www.quicken.com">`

- What information can you see being sent to insightgrit.com?

“Malicious code”?

- Why do we consider web bugs “malicious code”?
- This is an issue of privacy more than of security
- The web bug instructs your browser to behave in a way contrary to the principle of informational self-determination
  - Much in the same way that a buffer overflow attack would instruct your browser to behave in a way contrary to the security policy

Recap

- Malicious code: Malware
  - Viruses
  - Trojan horses
  - Login bombs
  - Worms
- Other malicious code: web bugs

Next time

- Other malicious code
  - Back doors
  - Salami attacks
  - Rootkits
  - Interface illusions
  - Keystroke logging
  - Man-in-the-middle attacks
- Nonmalicious flaws
  - Covert channels
  - Side channels