Malicious Software (Malware)

CS4264 Fall 2016
Malicious code has been around a long time

- Fred Cohen’87. First article on “Computer Viruses”
- Ken Thompson. Turing awardee
  - [Link](http://amturing.acm.org/award_winners/thompson_4588371.cfm)
  - Developed UNIX and C
  - Reflection on trusting trust’84
  - “The press must learn that misguided use of a computer is no more amazing than drunk driving of an automobile.”
  - Mentioned “trojan horse”

Thompson & Dennis Ritchie receiving National Medal of Technology for Unix and C
Definitions

- **Virus**: replicates itself, and passes on malcode to other programs by modifying them
- **Worm**: spreads copies of itself through a network
- **Trojan horse**: has a legitimate primary effect and a malicious nonobvious effect
- **Logic bomb**: goes off when a pre-defined condition is met
  - Time bomb
- **Trapdoor/backdoor**: a feature of a program in a secret way
  - Usually enabling remote connection, or escalated privileges
- **Insider threat**:
  - Security breach by insiders (e.g., employees, supposed to be trusted)
  - Security hole in a software system created by its own programmers
- **Social engineering attack**:
  - Trick the victim to give away important info, e.g., passwords
Mariposa botnet 12 million IPs; Stolen data belonging to 800K users; Malware changes every 48 hours; Attacker uses real name in DNS

http://pandalabs.pandasecurity.com/mariposa-botnet/

<table>
<thead>
<tr>
<th></th>
<th>Cities</th>
<th>%</th>
<th>IPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Seoul</td>
<td>5.36%</td>
<td>761,444</td>
</tr>
<tr>
<td>2</td>
<td>Bombay</td>
<td>4.45%</td>
<td>631,927</td>
</tr>
<tr>
<td>3</td>
<td>New Delhi</td>
<td>4.27%</td>
<td>605,518</td>
</tr>
<tr>
<td>4</td>
<td>Mexico</td>
<td>3.89%</td>
<td>551,705</td>
</tr>
<tr>
<td>5</td>
<td>Bogotá</td>
<td>2.68%</td>
<td>380,487</td>
</tr>
</tbody>
</table>
Real-time “Maps” of Internet Attacks

FireEye’s Cyber Threat Map
http://www.fireeye.com/cyber-map/threat-map.html

Norse Corp’s IPViking
http://map.norsecorp.com/

Arbor Networks’ Digital Attack Map (Global DDoS)
http://www.digitalattackmap.com/
#anim=1&color=0&country=ALL&list=0&time=16434&view=map

Kaspersky’s Cyberthreat Real-Time Map
https://cybermap.kaspersky.com/

A good summary article here:
http://krebsonsecurity.com/2015/01/whos-attacking-whom-realtime-attack-trackers/
• Trojan horse is a tale about the Trojan War in Troy
• A huge wooden horse with Greek soldiers hidden in it
Social Engineering Attack

Kevin Mitnick was the most-wanted computer criminal in US

Mitnick states that he compromised computers solely by using passwords and codes that he gained by social engineering

5 years in prison → Consultant 😊

Book: Ghosts in the wire
More on Social Engineering Attacks

- A reporter invited hackers to hack him, this is what happened
- https://www.youtube.com/watch?v=lc7scxvKQOo
COMPUTER VIRUS
How Viruses Attach to a Program

• Virus through email attachments
  – May require activation in order to replicate
  – You received a Word in the attachment, is it safe to open it?
  – Word macro virus (Macro allows one to define commands)

• Appended viruses
  – Inserts itself into programs
Recap: Love Letter Virus

Love letter virus May 2000
http://www.newsflash.org/2000/05/hl/hl012106.htm

A simple subject of "ILOVEYOU" and an attachment "LOVE-LETTER-FOR-YOU.TXT.vbs"

Most machines hide .vbs by default.

sent a copy of itself to everyone in the Windows Address Book and with the user's sender address
Infection Types

• Overwriting
  – Destroys original code
• Pre-pending
  – Keeps original code
  – Possibly compressed
• Infection of libraries
  – Allows virus to be memory resident
  – E.g., kernel32.dll
• Macro viruses
  – Infects MS Office documents
  – Installs in main document template
  – E.g., Melissa virus
Degrees of Complication

- Viruses have various degrees of complication in how they can insert themselves in computer code.
Virus Resident

• **One-time execution**
  – Email attachment being executed when open

• **Boot sector viruses**
  – Automatically execute when machine boots
  – First program to run
  – Takes advantage of chaining in bootstrap
  – Boots sector invisible to users

• **Memory-resident viruses**
  – Frequently used programs
  – Remain in memory – resident
  – Viruses attach to resident code

*Boot sector of a storage device contains machine code to be loaded into RAM (memory) by a computer’s firmware.*
# Virus Effects And Causes

<table>
<thead>
<tr>
<th>Virus effect</th>
<th>How it is caused</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attach to executables</td>
<td>Modify file directory</td>
</tr>
<tr>
<td></td>
<td>Write to executables</td>
</tr>
<tr>
<td>Attach to data/control file</td>
<td>Modify directory, data, append to data</td>
</tr>
<tr>
<td>Remain in memory</td>
<td>Intercept interrupt by modifying interrupt handler address table</td>
</tr>
<tr>
<td>Infect disks</td>
<td>Intercept interrupt, syscalls</td>
</tr>
<tr>
<td></td>
<td>Modify system file</td>
</tr>
<tr>
<td>Conceal self</td>
<td>Intercept syscalls that would reveal self and falsify result, classify self as hidden file</td>
</tr>
<tr>
<td>Spread infection</td>
<td>Infect boot sector, system program</td>
</tr>
<tr>
<td>Prevent deactivation</td>
<td>Store copy to reinfect, activate before deactivation program runs</td>
</tr>
</tbody>
</table>
Encrypted Viruses

- Decrypt routine is the same; different key for each infection
  - If a user launches an infected program, the virus decryptor first gains control of the computer and decrypts the virus body
  - The decryptor then transfers control to the decrypted virus

Virus Decryption Routine

1. Count = #VirusBytes
2. Temp = FetchNextByte
3. Temp = Decrypt(Temp)
4. StoreNextByte(Temp)
5. Decrement Count
6. If Count>0, GOTO 2
7. #$^@^#^#!^!#^!^#^!!
8. !#@%$!@%!@%!@#'
9. $#&!&%!#&#!%^!!

First Decrypted Byte

```
1. Count = #VirusBytes
2. Temp = FetchNextByte
3. Temp = Decrypt(Temp)
4. StoreNextByte(Temp)
5. Decrement Count
6. If Count>0, GOTO 2
7. #$^@^#^#!^!#^!#^!!
8. !#@%$!@%!@%!@#'
9. $#&!&%!#&#!%^!!
```
Polymorphic Viruses

• Why Polymorphic?
  – Virus replicates itself in different forms
  – Virus looks different for each copy so that signature based detection fails

• Polymorphic virus
  – Having an encryption/decryption routine
  – Mutates with each copy, a new decryption routine

• Polymorphic virus contains a mutation engine
• When infecting new program, mutation engine randomly generates a new decryption routine in the new copy of virus
• Virus encrypts body and mutation engine, append new decryption routine

In 1992 Dark Avenger distributed the Mutation Engine, also known as MtE and kit for writing polymorphic viruses
Metamorphic Virus

• Decrypted program body mutates as well (along with the decryption routine)
  – Metamorphics are body-polymorphic.

Different ways to achieve body polymorphic

1. Use different registers
2. Insert NOP instructions
3. Insert dummy instructions
4. Insert JUMPS
5. Reorder/permutation of instructions (that are independent of each other)
6. Code the instruction differently (e.g., sub eax, eax → xor eax, eax)

Polymorphic Virus Detection

First off, read file from disk into VM memory

To load encrypted virus into a virtual machine and let it decrypt itself; Then, scan for known signatures on the decrypted memory region
Load the virus (decryptor, encrypted body) and host program into memory;
Virus decrypts itself. 
We record how the memory changes as the virus decrypts itself.
After the virus is decrypted, we scan the memory for virus signatures.
Virus Signatures (Characteristic Patterns) And Virus Scanner

• What is a virus signature
  – Hexdecimal opcode: 88 16 00 80 88 26 00 0d cd 13 cd 19
  – With many NOPs: 88 16 00 80 90 88 26 00 0d 90 90 cd 13 90 90 90 cd 19

• How to extract virus signature
  – Signature is found in every infected object by the virus, but not otherwise
  – Statistical methods on a large corpus of programs

• How to do fast scan
  – Boyer-Moore fast string search algorithm
  – A demo program here:
    http://userweb.cs.utexas.edu/users/moore/best-ideas/string-searching/fstrpos-example.html

• Create your own signature with Clam Anti-Virus (open-source)

Strother Moore,
NAE member, ACM Fellow
COMPUTER WORMS
Morris Worm ‘88

- Dictionary attack on passwords
  - User passwords were encrypted & stored in a public file
  - Attacker encrypted lists of possible pwds and compared against ones in file
  - If pwd broken, worm attempted to break into remote machines

- Exploits buffer overflow vulnerability in fingerd on VAX
  - finger daemon (fingerd)
  - gets routine (scanf, fscanf, sscanf, sprintf, strcat, strcpy) no boundary checking on inputs

- Exploited DEBUG command of sendmail
  - The worm specified commands instead of email address as email recipient
  - DEBUG option enabled by default for ease of configuration (backdoor)
Code Red Worm (2001)

- Infected Windows Internet Information Server (IIS)
  - 2nd, most popular web server, next to Apache
  - Of the 6 million IIS web servers, 1 out of 8 were infected
- The worm overflowed a buffer in a dynamic link library idq.dll
- It defaced the web server and propagated

- Later version opened a backdoor that allowed to run cmd.exe
  - Backdoor allows someone to access a program via unconventional method usually with special privileges
  - 300-600 threads and tried for 24-48 hours to connect to other machines
  - Changed and kept resetting system registry to make sure its code runs
Targeted malicious code: written for a particular system, application, purpose

- Cyber espionage,
- Trojan hydraq
- Stuxnet (USB-based malware targeting SCADA)

*Trapdoor/backdoor*: undocumented entry point to a module
- For debugging, unit testing, integration testing with stubs and drivers
- Forget to remove them
- Intentionally leave there for testing (e.g., sendmail in Morris Worm)
- Intentionally leave there for attacks
Sony XCP Rootkit

- **Rootkit:**
  - Malicious code attempt to operate as root and hide its presence
  - Interferes with the normal interaction between a user and OS
    - E.g., deletes itself from files, processes listing (ls, ps commands in Linux)
- Detection is usually based on comparison
  - Compare data obtained at different OS levels to find discrepancies

- **Sony XCP rootkit (2005) prevents a user from copying a music CD (for digital rights management)**
  - Infected 22 millions CDs
  - Automatically install itself from the music CD (called autorun.exe)
  - It blocks the display of any program beginning with $sys$

**What Sony did wrong:**
1. distributed code that may open a system to possible infection by other attackers
2. install code without user’s knowledge, much less consent, and hide itself
Other Targeted Malcode

• Privilege escalation
  – Example: Symantec flaw in Live Update that may escalate a user program with elevated privileges
  – A non-privileged user can access and manipulate the Automatic Live Update interface in order to gain privileged access to the host computer.

• Interface illusion, social engineering
  – Possible defense: trusted display

• Keystroke logging

• Timing attack, side channel attack
  – E.g., infer cryptographic key based on observing computation time

• Covert channels

“Covert channels emerged in mystery and departed in confusion. “ Jonathan Millen
Service program is malicious; it colludes with Spy’s program:

- It gives away info about protected data by manipulating the lock of a file;
- The lock is visible by an attacker.

101001000010101 ... ➔ “secret”
**example:** The service program sends out Signal “100” to the attacker via the lock of the file.
Covert Channel Summary

• Challenging to detect/prevent covert channels, if not impossible
• Actual covert channel attacks do not exist (*)
  – But the analysis is sound
• Steganography: the technique for hiding an image within a coverimage
• Kurak-McHugh 1992 method:
  – replace the n least significant bits (LSB) of each pixel in coverimage with the n most significant bits (MSB) from the pixel of the secret image (n ~ 2)

* Arrests of alleged spies draws attention to long obscure field of steganography
  http://www.washingtonpost.com/wp-dyn/content/article/2010/06/30/AR2010063003108.html
Countermeasure: Program Analysis

• Useful to identify new and “zero day” malware

• **Static programming analysis**
  – Based on the instructions, determine whether the program is malicious, i.e., program contains instruction to delete system files

• **Dynamic analysis**
  – Run code in isolated emulation environment
  – Monitor actions that target file takes
    – E.g., visiting websites and executable downloaded automatically
  – If the actions are harmful, mark as virus

• May trigger false alarms
White listing: Tripwire (a tool for file sys integrity)

- Tripwire is a classic file system integrity tool
  - How to make sure your OS files are not tampered with?

- Maintain database of cryptographic hashes for
  - Operating system files (clean version)
  - Popular applications (clean version)

- Re-compute hash value of each current file

- Look up the recomputed hash values into database

- Needs to protect the integrity of the database

Think about pragmatic security issues to make this secure
Computer Virus – Theory and Experiments
by Fred Cohen
J. of Virology 1987

“That is, they (USC) told me everything I needed to know all the way down to the bottom.”
Virus Detection is Undecidable
There is no “perfect” virus scanning algorithm

- Theoretical result by Fred Cohen (1987)
- Virus abstractly modeled as program that eventually executes `infect`
- Code for `infect` may be generated at runtime
- Proof by contradiction

1. Suppose antivirus program `isVirus(P)` determines whether program P is a virus
2. Define new program Q:
   ```
   if (not isVirus(Q))
   infect
   stop
   ```
3. Running `isVirus` on Q achieves a contradiction
Other Undecidable Detection Problems

- Detection of a virus
  - by its appearance
  - by its behavior
- Detection of an evolution of a known virus
- Detection of a triggering mechanism
  - by its appearance
  - by its behavior
- Detection of a virus detector
  - by its appearance
  - by its behavior
How to prevent computer viruses from spreading?

Isolationism

If there is no sharing, then no dissemination of information across information boundaries.

Thus, a virus cannot spread outside a single partition.

- However, isolationism is unacceptable if we wish to benefit from the work of others.
Partition models and flow models

- **Partition model** -- partitions of a system into isolated subsystems
  - Resulting in closed subsets
  - e.g., Biba integrity model (read-up/write-down)
  - Biba: no user at a given integrity level can read an object of lower integrity or write an object of higher integrity.

- **Flow model** – no partition, but limit the extend virus can spread
  - Flow distance’ policy keeps track of the distance (number of sharing) over which data has flowed
  - e.g., user A can only be allowed to access information written by users (B and C) or (B and D), but not information written by B, C, or D alone. Useful to enforce certification of information by B, before C or D can pass it to A.