Reversing Malware

[based on material from the textbook]

What is Malware?

- Malware (malicious software) is any program that works against the interest of the system's user or owner.
- **Question:** Is a program that spies on the web browsing habits of the employees of a company considered malware?
- What if the CEO authorized the installation of the spying program?

Reversing Malware

- Revering is the strongest weapon we have against the creators of malware.
- Antivirus researchers engage in reversing in order to:
  - analyze the latest malware,
  - determine how dangerous the malware is,
  - learn the weaknesses of malware so that effective antivirus programs can be developed.

Uses of Malware

- Why do people develop and deploy malware?
  - Financial gain
  - Psychological urges and childish desires to "beat the system".
  - Access private data
  - …

Typical Purposes of Malware

- Backdoor access:
  - Attacker gains unlimited access to the machine.
- Denial-of-service (DoS) attacks:
  - Infect a huge number of machines to try simultaneously to connect to a target server in hope of overwhelming it and making it crash.
- Vandalism:
  - E.g., defacing a web site.
- Resource Theft:
  - E.g., stealing other user's computing and network resources, such as using your neighbors' Wireless Network.
- Information Theft:
  - E.g., stealing other user's credit card numbers.

Types of Malware

- Viruses
- Worms
- Trojan Horses
- Backdoors
- Mobile code
- Adware
- Sticky software

Viruses

- Viruses are self-replicating programs that usually have a malicious intent.
- Old fashioned type of malware that has become less popular since the widespread use of the Internet.
- The unique aspect of computer viruses is their ability to self-replicate.
- However, someone (e.g., user) must execute them in order for them to propagate.

Viruses (Cont'd)

- Some viruses are harmful (e.g.,):
  - delete valuable information from a computer's disk,
  - freeze the computer.
- Other viruses are harmless (e.g.,):
  - display annoying messages to attract user attention,
  - just replicate themselves.

Viruses: Operation

- Viruses typically attach themselves to executable program files
  - e.g., `.exe` files in MS Windows
- Then the virus slowly duplicates itself into many executable files on the infected system.
- Viruses require human intervention to replicate.

Origin of the term Computer Virus

- The term computer *virus* was first used in an **academic** publication by Fred Cohen in his 1984 paper *Experiments with Computer Viruses*.
- However, a mid-1970s science fiction novel by David Gerrold, *When H.A.R.L.I.E. was One*, includes a description of a fictional computer program called VIRUS.
- John Brunner's 1975 novel The *Shockwave Rider* describes programs known as *tapeworms* which spread through a network for deleting data.
- The term computer virus also appears in the comic book *Uncanny X-Men* in 1982.

The first Computer Viruses

- A program called *Elk Cloner* is credited with being the first computer virus to appear "in the wild". Written in 1982 by Rich Skrenta, it attached itself to the Apple DOS 3.3 operating system and spread by floppy disk.
- The first PC virus was a boot sector virus called (c)Brain, created in 1986 by two brothers, Basit and Amjad Farooq Alvi, operating out of Lahore, Pakistan.

Worms

- Worms are malicious programs that use the Internet to spread.
- Similar to a virus, a worm self-replicates.
- Unlike a virus, a worm does not need human intervention to replicate.
- Worms have the ability to spread uncontrollably in a very brief period of time.
  - Almost every computer system in the world is attached to the same network.

Worms: Operation

- A worm may spread because of a software vulnerability exploit:
  - Takes advantage of the OS or an application program with program vulnerabilities that allow it to hide in a seemingly innocent data packet.
- A worm may also spread via e-mail.
  - Mass mailing worms scan the user's contact list and mail themselves to every contact on such a list.
  - In most cases the user must open an attachment to trigger the spreading of the worm (more like a virus).

Trojan Horses

- A Trojan Horse is a seemingly innocent application that contains malicious code that is hidden somewhere inside it.
- Trojans are often useful programs that have unnoticeable, yet harmful, side effects.

Trojan Horses: Operation (1)

- Embed a malicious element inside an otherwise benign program.
- The victim:
  1. receives the infected program,
  2. launches it,
  3. remains oblivious of the fact that the system has been infected.
- The application continues to operate normally to eliminate any suspicion.

Trojan Horses: Operation (2)

- Fool users into believing that a file containing a malicious program is really an innocent file such as a video clip or an image.
- This is easy to do on MS Windows because file types are determined by their extension as opposed to examining the file headers.
- E.g.,
  - "`A Great Picture.jpg .exe`"
  - The `.exe` might not be visible in the browser.
  - The Trojan author can create a picture icon that is the default icon of MS Windows for `.jpg` files.

Backdoors

- A backdoor is malware that creates a covert access channel that the attacker can use for:
  - connecting,
  - controlling,
  - spying,
  - or otherwise interacting with the victim's system.

Backdoors: Operation

- Backdoors can be embedded in actual programs that, when executed, enable the attacker to connect to and to use the system remotely.
- Backdoors may be planted into the source code by rogue software developers before the product is released.
  - This is more difficult to get away with if the program is open source.

Mobile Code

- Mobile code is a class of benign programs that are:
  - meant to be mobile,
  - meant to be executed on a large number of systems,
  - not meant to be installed explicitly by end users.
- Most mobile code is designed to create a more active web browsing experience.
  - E.g., Java applets, ActiveX controls.

Mobile Code (Cont'd)

- Java scripts are distributed in source code form making them easy to analyze.
- ActiveX components are conventional executables that contain native IA-32 machine code.
- Java applets are in bytecode form, which makes them easy to decompile.

Mobile Code: Operation

- Web sites quickly download and launch a program on the end user's system.
- User might see a message that warns about a program that is about to be installed and launched.
  - Most users click OK to allow the program to run.
  - They may not consider the possibility that malicious code is about to be downloaded and executed on their system.

Adware

- Adware is a program that forces unsolicited advertising on end users.
- Adware is a new category of malicious programs that has become very popular.
- Adware is usually bundled with free software that is funded by the advertisements displayed by the Adware program.

Adware: Operation (1)

- The program gathers statistics about the end user's browsing and shopping habits.
  - The data might be transferred to a remote server.
- Then the Adware uses the information to display targeted advertisements to the end user.

Adware: Operation (2)

- Adware can be buggy and can limit the performance of the infected machine.
  - E.g., MS IE can freeze for a long time because an Adware DLL is poorly implemented and does not use multithreading properly.
- Ironically, buggy Adware defeats the purpose of the Adware itself.

Sticky Software

- Sticky software implements methods that prevent or deter users from uninstalling it manually.
- One simple solution is not to offer an uninstall program.
- Another solution in Windows involves:
  - installing registry keys that instruct Windows to always launch the malware as soon as the system is booted.
  - The malware monitors changes to the registry and replace the keys of they are deleted by the user.
  - The malware uses two mutually monitoring processes to ensure that the user does not terminate the malware before deleting the keys.

Future Malware

- Today's malware is just the tip of the iceberg.
- The next generation of malware may take control of the low levels of the computer system (e.g., BIOS, Firmware).
  - The antidote software will be in the control of the malware …
- Also the theft of valuable information can result in holding it for ransom.

Information-stealing Worms

- Present-day malware does not take advantage of cryptography much.
- Asymmetric encryption creates new possibilities for the creation of information-stealing worms.
- A worm encrypts valuable data on the infected system using an asymmetric cipher and hold the data as ransom.

Information-stealing Worms:Operation

1. The Kleptographic worm embeds a public encryption key in its body.
2. It starts encrypting every bit of valuable data on the host using the public key.
3. Decryption of the data is impossible without the private key.
4. Attacker blackmails the victim demanding ransom.
5. Attacker exchanges the private key for the ransom while maintaining anonymity.
   - Theoretically possible using zero-knowledge proofs
   - Attacker proves that he has the private key without exposing it.

BIOS/Firmware Malware

- Antivirus programs assume that there is always some trusted layer of the system.
- Naïve antivirus programs scan the hard drive for infected files using the high-level file-system service.
- A clever virus can intercept file system calls and present to the virus with fake versions (original/uninfected) of the files on disk.
- Sophisticated antivirus programs reside at a low enough level (in OS kernel) so that malware cannot distort their view of the system.

BIOS/Firmware Malware: Operations (1)

- What is the malware altered an extremely low level layer of the system?
- Most CPUs/hardware devices run very low-level code that implements each assembly language instruction using low level instructions (micro-ops).
- The micro-ops code that runs inside the processor is called firmware.
- Firmware can be updated using a firmware-updating program.

BIOS/Firmware Malware: Operations (2)

- Malicious firmware can (in theory) be included in malware that defeats antivirus programs.
- The hardware will be compromised by the malicious firmware.
- Not easy to do in practice because firmware update files are encrypted (private key inside the processor).

Reversing Malware

- Malware is vulnerable to reversing.
- Even encryption-based protection can be reversed.
  - E.g., examine the unencrypted version of the code being executed in memory.
- One approach is to hide the malware from the user by embedding it into benign code.
  - E.g., file name changes, embedding code in OS code.
- Another approach is using anti-reversing:
  - Anti-reversing techniques attempt to scramble or complicate the code to prolong the reversing process.

Reversing Malware (3): Static Analysis of Malware

- **BinText:**
  - Extracts strings from executables, revealing registry keys used, and various commands stored in string format.
- **IDA Pro:**
  - Disassembler (executable to assembly code).
- **UPX:**
  - UPX compression and decompression, the most common executable packer used by virus and malware writers.
- **Proc Dump:**
  - Dumps code from memory.
- **OllyDbg:**
  - A debugger that enables the user to attach to a process and insert breakpoints.

Reversing Malware (3): Dynamic Analysis of Malware

- **Process Explorer:**
  - Tells what processes are currently running.
- **FileMon:**
  - Monitors files for operations.
- **RegMon:**
  - Monitors registry for operations.
- **RegShot:**
  - Takes a snapshot of the registry and associated files .
- **TCPView:**
  - Displays all TCP and UDP open connections and the process that opened and is using the port.
- **TDIMon:**
  - Logs network connectivity, but does not log packet contents.
- **Ethereal:**
  - Packet Scanner that captures packets and supports the viewing of contents/payload.

Antivirus Programs

- Antivirus programs identify malware by looking for unique signatures in the code of each program (i.e., potential virus) on a computer.
  - A signature is a unique sequence of code found in a part of the malicious program.
- The antivirus program maintains a frequently updated database of virus signatures.
  - The goal is for the database to contain a signature for every known malware program.
- Well known antivirus software includes:
  - Symantec (http://www.symantec.com)
  - McAfee (http://www.mcafee.com)

Polymorphic Viruses

- Polymorphism is a technique that thwarts signature-based identification programs.
- Polymorphic viruses randomly encode or encrypt the program code in a semantics-preserving way.
- The idea is to encrypt the code with a random key and decrypt it at runtime.
  - Each copy of the code is different because of the use of a random key.

Polymorphic Viruses: Decryption technique

- A decryption technique that polymorphic viruses employ involves "XORing" each byte with a randomized key that was saved by the parent virus.
- The use of XOR-operations has the additional advantage that the encryption and decryption routine are the same:
  - a xor b = c
  - c xor b = a

Polymorphic Viruses: Weaknesses

- Many antivirus programs scan for virus signatures in memory.
  - I.e., after the polymorphic virus has been decrypted.
- If the virus code that does the decryption is static, then the decryption code can be used as a signature.
- This limitation can be addressed (somewhat) if the decryption code is scrambled (superficially):
  - randomize the use of registers,
  - add no-ops in the code, …

Metamorphic Viruses

- Instead of encrypting the program's body and making slight alterations in the decryption engine, alter the entire program each time it is replicated.
- This makes it extremely difficult for antivirus writers to use signature-matching techniques to identify malware.
- Metamorphism requires a powerful code analysis engine that needs to be embedded into the malware.

Metamorphic Viruses: Operation

- Metamorphic engine scans the code and generates a different version of it every time the program is duplicated.
- The metamorphic engine performs a wide variety of transformations on the malware and on the engine itself.
  - Instruction and register randomization.
  - Instruction ordering
  - Reversing (negating) conditions
  - Insertion of "garbage" instructions
  - Reordering of the storage location of functions

Case Study: Backdoor.Hackarmy.D

- In Chapter 8 of the book there is an interesting case study on reversing malware.
- It involves reversing the Backdoor.Hackarmy.D malware.
- The next few slides outline some of the salient results from the analysis.
- You are encouraged to read through the analysis in detail and, perhaps, try to re-create the analysis yourselves.

Backdoor.Hackarmy.D: Overview

- Backdoor.Hackarmy.D is a Trojan that lacks any automated self-replication mechanisms.
- It is distributed as an innocent picture file and has a `.scr` (screensaver) extension.
- The Trojans temps the unsuspecting user to open the picture and, thus, activate the backdoor.

Backdoor.Hackarmy.D: Unpacking the Executable

- An executable packer is a program that compresses or encrypts an executable program.
- The program is automatically restored to original state in memory once the program is launched.
- Some packers are designed as anti-reversing tools that encrypt the program and try to fend off debuggers and disassemblers.
- Some packers simply compress the program to decrease its size.
- Backdoor.Hackarmy.D uses the UPX packer to simply decrease its size.

Backdoor.Hackarmy.D: Initialization

- When the backdoor is launched, nothing happens from the user's perspective.
- If the backdoor was more clever, it would launch an application and display a picture.
- However, if you check the processes on the Task Manager you will see a process called `ZoneLockup.exe`.
- The name is supposed to fool the use into thinking that the process is a security component.

Backdoor.Hackarmy.D: A Chat Program

- The assembly code reveals that port number 6667 is being used.
- This port number is in the range 6665-6669, which is usually reserved for Internet Relay Chat (IRC) services.
- Looks like the Trojan is looking to chat with someone … the attacker most likely.
- The USER string is embedded in the assembly:
  - `NICK vsorpy USER vsorpy "X.COM" "X":X`
- This registers a new user called `vsorpy` onto the IRC server.

Backdoor.Hackarmy.D: Communicating

- The attacker communicates with the backdoor through the use of **private-message packets (PRIVMSG)**.

1. Find the code for parsing the backdoor commands by searching for the part of the code that processes the PRIVMSG commands sent from the server.
2. Reverse the command strings (this is easy).
3. Reverse the commands by analyzing the code that follows the parsing of the command strings.

Backdoor.Hackarmy.D: Summary of Commands (1)

- `!?dontuseme`:
  - Self destruct the program by removing its `Autorun` registry entry and deleting the executable.
- `!socks4`:
  - Turns the infected system into a proxy servers.
- `!threads`:
  - Lists currently active server threads.
- `!info`:
  - Lists general information about the infected host (e.g., name, IP address, CPU model).
- `!?quit`:
  - Closes the backdoor process without uninstalling the program.
- `!?disconnect`:
  - Causes the program to disconnect from the IRC server, wait, and then reconnect.

Backdoor.Hackarmy.D: Summary of Commands (2)

- `!execute`:
  - Executes a local binary on the host.
- `!delete`:
  - Deletes a file from the infected host.
- `!webfind64`:
  - Instructs the infected host to download a file from a remote server using http or ftp.
- `!killprocesses !listprocesses` :
  - Unreachable code, perhaps a future feature.
  - Names suggest what these features will do …

Backdoor.Hackarmy.D: More on `!?dontuseme`

- The `!?dontuseme` command uninstalls the program from the registry and deletes the executable.
- This is difficult because an executable program file cannot be deleted while the program is running.
- A self-destruct batch file is generated, which deletes the executable after the program exists.
- The code for the batch file explains how this is done …

Backdoor.Hackarmy.D: More on `!?dontuseme (rm.bat)`

```
@echo off
:start
if not exist "c:\WINNT\SYSTEM32\ZoneLockup.exe"
   goto done
del "c:\WINNT\SYSTEM32\ZoneLockup.exe"
goto  start
:done
del rm.bat
```

Backdoor.Hackarmy.D: More on `!socks4`

- The Backdoor.Hackarmy.D socks4 command establishes a thread that waits for connections that use the SOCKS4 protocol.
- SOCKS4 is a proxy communications protocol that can be used for indirectly accessing a network.
- Using SOCKS4 one can route all traffic through a single server.
- Allows attackers to connect "anonymously" (i.e., with the userid of the victim on the host) to servers on the Internet.
- Difficult to trace back to the system from which traffic is originating.

Timeline of famous malware (1982-1988) [wikipedia]

- 1982
  - **Elk Cloner**, written for Apple II systems, is credited with being the first computer virus.
- 1987
  - **(c)Brain**, the first virus written for PCs.
  - **SCA**, a boot sector virus for Amiga appears, immediately creating a pandemic virus-writer storm. A short time later, SCA releases another, considerably more destructive virus, the Byte Bandit.
- 1988
  - **Morris** worm infects DEC VAX machines connected to the Internet, and becomes the first worm to spread extensively.

Timeline of famous malware (1998-2000) [wikipedia]

- 1998
  - **CIH** virus version 1.
- 1999
  - **Melissa** worm is released, targeting Microsoft Word and Outlook-based systems, and creating considerable network traffic.
- 2000
  - The **VBS/Loveletter** worm, also known as the "I love you" virus appeared. As of 2004, this was the most costly virus to business, causing upwards of 10 billion dollars in damage.

Timeline of famous malware (2001) [wikipedia]

- **Klez** worm.
- **Nimda** worm.
- **Code Red II** worm (spreads in China, attacks Microsoft's Internet Information Services.
- **Sircam** worm (spreads through e-mails and unprotected network shares).
- **Sadmind** worm (spreads by exploiting holes in both Sun Microsystem's Solaris and MS IIS).
- **Raman** worm (similar to the Morris worm infected only Red Hat Linux machines running version 6.2 and 7.0, using three vulnerabilities in `wu-ftpd`, `rpc-statd` and `lpd`.

Timeline of famous malware (2003) [wikipedia]

- **Sober** worm is first seen and maintains its presence until 2005 with many new variants.
- **Sobig** worm (technically the Sobig.F worm) spread rapidly via mail and network shares.
- **Blaster** worm also know as the Lovesan worm spread rapidly by exploiting MS computers.
- **SQL slammer** worm also known as the Sapphire worm, attacked vulnerabilities in Microsoft SQL Server and MSDE, causes widespread problems on the Internet.

Timeline of famous malware (2004) [wikipedia]

- **Sasser** worm emerges by exploiting a vulnerability in LSASS, causes problems in networks.
- **Witty** worm is a record breaking worm in many regards.
  - It exploited holes in several Internet Security Systems (ISS) products.
  - it was the first internet worm to carry a destructive payload and it spread rapidly using a pre-populated list of ground-zero hosts.
- **MyDoom** emerges, and currently holds the record for the fastest-spreading mass mailer worm.

Timeline of famous malware (2005) [wikipedia]

- **Zotob** worm, the effect was overblown because several United States media outlets were infected.

Conclusions

- Educating users on how malware works is important, but not enough.
- Software developers must exercise caution to make their programs more secure from vulnerabilities such as buffer overflow attacks.
- You learned about:
  - the different types of malware
  - how malware works
  - how malware hides from antivirus scanners
- Hands on experience on reversing the BeagleJ virus is a good introduction to reversing and understanding malware.