Virus Concepts and Terminology

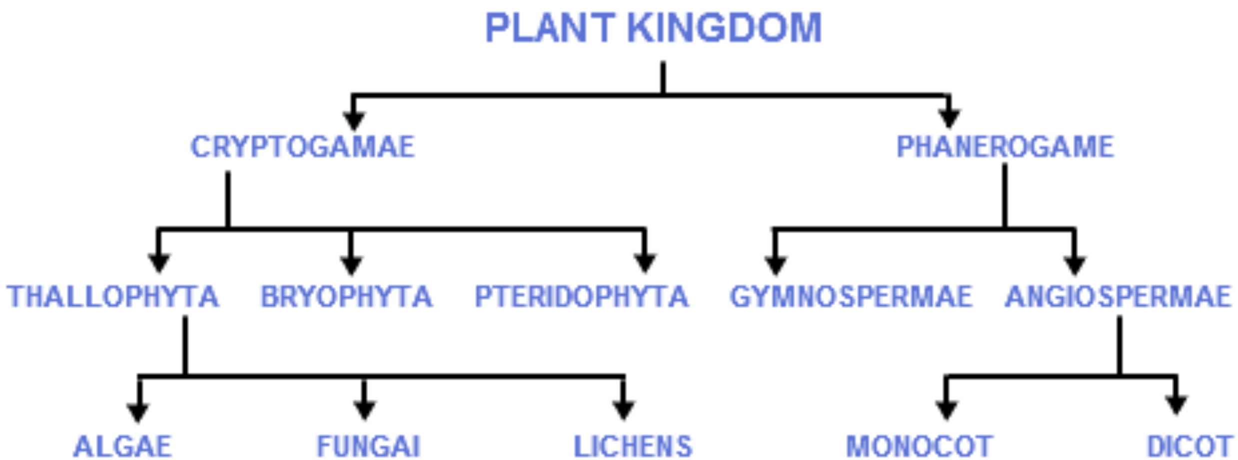
CS 4440/7440 Malware Analysis & Defense

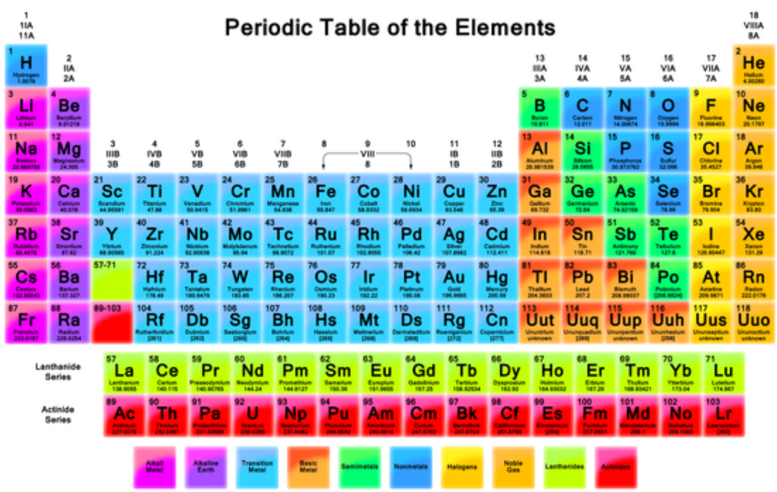
Today

- Take a look at this:
  - https://www.corelan.be/index.php/articles/
  - Check out the "Exploit Writing Tutorials"

- Starting Szor, Chapter 2.
- Check out the Virus Bulletin: https://www.virusbulletin.com

Taxonomy & Controlled Vocabulary





CARO Ontology

- Computer Antivirus Researchers Organization
- Standard Taxonomy for Malware
  - From 1991
  - A bit long in the tooth.
- <malware_type> ://<platform>/ <family_name>.<group_name>.<infective_length>. …
- <family_name>: key component in classification
- <malware_type>:
  - Virus
  - Trojan
  - …

Concepts and Terminology

- First we will learn to classify attacks, then learn the definitions of malicious code types
- One key term first:
  - "A computer virus is code that recursively replicates a [possibly evolved] copy of itself." (Szor, section 2.3.1)
  - A "worm" is just a virus that spreads over networks
  - More details on viruses, worms, etc. later

Classifying Malicious Attacks

- We understand malicious attacks by asking the right questions:
  1. How was the attack created?
  2. How was malicious code transported?
  3. What vulnerabilities were exploited?
  4. What damage did the attack cause?

Classifying Malicious Attacks

1. How was the attack created?
2. How was malicious code transported?
3. What vulnerabilities were exploited?
4. What damage did the attack cause?

1. How Was the Attack Created?

- Assembly language code
  - Very common
  - Security professionals must be expert in assembly language to analyze attacks
- High level language or scripts
- Virus generator kits
  - Attackers distribute kits to generate most of the code of common viruses, ready for alteration and enhancement

Creation in Assembly Language

- Easier to use assembly language to create the typical virus code that hides inside a user application
  - Space available can be tight
  - Must analyze existing object code, deposit virus object code inside it
  - Virus must perform its own assembler/linker work, e.g. relocations
- Easier to obfuscate assembly code

Creation in HLL or Script

- Most useful for standalone attack code
  - Root kits (exploit OS weakness to run commands as *root*, or *admin*)
  - DOS (denial of service) attacks that flood a website
  - Program attached to email, opened by unsuspecting user
  - Macros in Word, Excel, etc. files
- Increasing due to spread of scripting and macro languages
- Many applications have extensible API
  - Flexibility (good) & potential for exploitation (bad)

Script Attacks

- Script and macro languages are popular because they are high-level
- Scripts are useful because they can call basic operating system functions
  - This is what makes them dangerous!
- OS designers must carefully decide what functions can be called by user-level scripts
  - Permission errors are common, allowing attacks to succeed
- LoveLetter mass mailer virus is an example of succeeding because the script was granted high permissions.
  - An email attachment should not have been granted as much permission as Outlook gave it.

Virus Construction Kits

- First was VCS (Virus Construction Set) in Germany in 1990
- Dozens have followed, creating assembly and HLL code, 16-bit and 32-bit DOS and Windows viruses, malicious scripts of many kinds, worms, etc.
- Usually create standalone programs, but these can embed viruses in applications when they are first executed
- Metasploit

Virus Construction Kits cont'd.

- VCL (Virus Creation Laboratory) in 1992 produced the first viruses to become widespread
  - Produced assembly language code
  - User could select among different payloads, infection strategies, and encryption techniques
- Very hard for antivirus software to detect all possible combinations
- Graphical IDE made it possible for "script kiddies" to create viruses
- VCL is discussed in Chapter 7 of Szor.

Classifying Malicious Attacks

- How was the attack created?
- How was malicious code transported?
- What vulnerabilities were exploited?
- What damage did the attack cause?

How was Malware Transported?

- Early viruses were on floppy disks shared among users
- Email attachments are common
  - Self-remailing viruses have been among the most costly
- Worms send themselves over network
- Also: chat/IM transport; free software downloads from web or FTP sites

Floppy Diskette Transport

- Pre-internet viruses were on floppy disks shared among users
  - Virus lived on hard disk or in memory
  - Sometimes infected the OS utilities that are called whenever a diskette is formatted or written
  - Infected system then created infected diskettes
- Flash drives are being infected in an analogous way today
  - Not as common, because email programs and internet access provide a greater opportunity for wider and faster malicious code transport

Email Transport

- Viruses can use an email program and associated address books to re-mail to many users
- Usually starts by opening an attachment that is executable
- Virus creators try to disguise the file type so it does not look executable
- Even spreadsheet and document files can contain macros that are executable viruses

Email Transport cont'd.

- Why would anyone open an email attachment that is obviously an executable?
- The virus creator can make it look like the file is NOT executable
- Example: The "I Love You" mass mailer virus came in an attachment called LOVE-LETTER-FOR-YOU.TXT.vbs
- "User-friendly" Windows OS suppresses file extensions for known file types unless you prevent it, so it removed the ".vbs" extension
- Attachment now looks like a *.txt file

Internet Transport

- Internet provides great opportunities for malicious code transport
  - Virus can access OS networking commands, e.g. sendmail and rlogin
  - Networking utilities allow virus to probe the Internet for the next victim machine
  - Broadband access means many machines are always on and always connected
  - FTP sites and public web sites are, by nature, accessible to outsiders to some degree

Internet Transport cont’d.

- Internet provides great opportunities for malicious code transport (cont’d.)
  - Browsers have hidden background tasks, cookies, spyware and other information-gathering software
  - Data packets over the internet can be “snooped” by attackers, and most such packets are unencrypted
  - Sensitive information is stored all over the internet on e-commerce servers, government servers, etc
  - Network file systems permit remote access to files

Downloaded Software Transport

- Free software has become widely available
- Contributors can post infected files, knowingly or not, for others to download
- How do you know you can trust what you are downloading?
  - Trust in downloaded software comes from data authentication along with antivirus scanning on the server side.

Classifying Malicious Attacks

- How was the attack created?
- How was malicious code transported?
- What vulnerabilities were exploited?
- What damage did the attack cause?

What Vulnerabilities Were Exploited?

- "Vulnerability" often refers only to vulnerable code in an OS or applications
  - E.g. Unguarded buffer overflow in OS command allows attacker to run arbitrary command, gain root access, etc.
  - Failure to validate user input
  - Allowing ActiveX controls to be run from scripts
- More generally, a vulnerability is whatever weakness in an overall system that makes it open to attack
  - System administration and configuration flaws
  - Dangerous user behavior

Code Vulnerabilities

- Buffer overflow is the most common
  - Array bounds not usually checked at run time (Why not?)
- What comes after the buffer being overflowed determines what can be attacked
  - Return address can be changed to malicious code
  - Function pointer can point to malicious code
  - Output file name for a program can be overwritten with file name desired by attacker
- Buffer overflows are simple to guard against, yet they remain the most common code vulnerability
  - W `or` X stack disciplines

Buffer Overflow Example

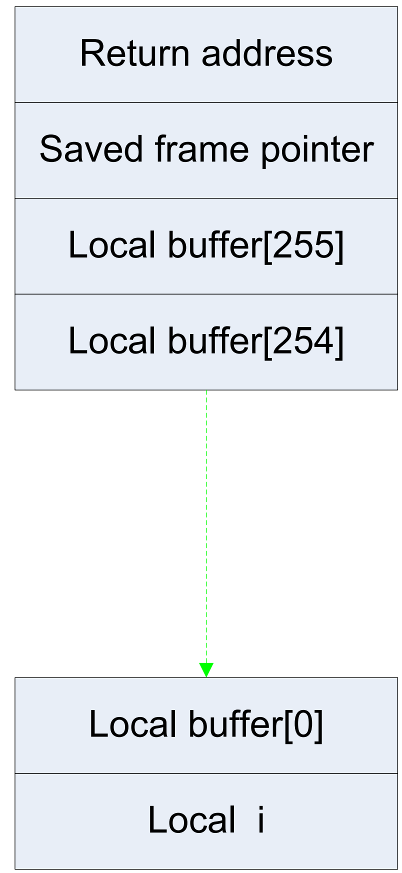
```
void bogus(void) {
    int i;
    char buffer[256];                // Return address follows!

    printf(“Enter your data as a string.\n”);
    scanf(“%s”, buffer);          // No bounds check!

    process_data(buffer);
    return;
    // Returns to the return address that follows buffer[]
    // on the stack frame
}
```

Buffer Overflow cont'd.

- In the stack frame for bogus(), buffer[257] would fall on top of the return address:



Buffer Overflow cont'd.

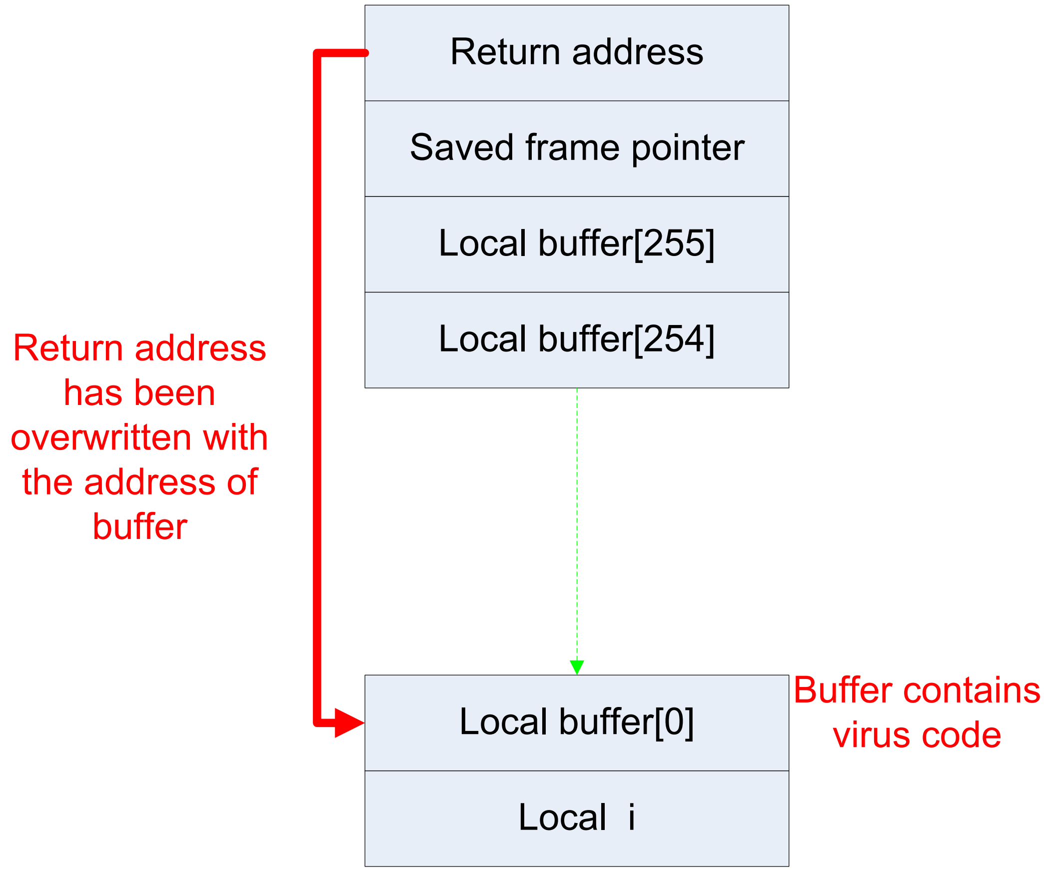
- Notice that the program does not check to make sure that the user inputs 255 characters or less
- Source code is available for many operating systems and applications; OR they can be disassembled and analyzed by the attacker
- Attacker can see that it is possible to overflow the buffer
- Buffer is last data item on the stack frame; the return address from this function will be at a defined distance after it

Buffer Overflow cont’d.

- Attacker can enter a character string representation of his malicious object code, long enough to fill the buffer
- At the end of the malicious code, the attacker passes the address of variable “buffer” so that it overwrites the return address of function bogus() on the stack frame
- When bogus() returns, it will cause a return to the buffer address, executing the malicious code in it

Buffer Overflow cont'd.

- **`bogus()`** is now "returning" to **`buffer[0]`**



User Behavior Vulnerabilities

- Poor password selection
  - Too short; all alphabetic; common words
- 1988 Morris worm used a list of only 432 common passwords, and succeeded in cracking many user accounts all over the internet
  - This was the main reason the worm spread more than the creator thought it would;
  - Morris did not realize that password selection was that bad!

User Behavior Vulnerabilities cont'd.

- Opening executable email attachments
  - "This email is from my friend; it must be safe." But, the friend's PC has a virus!
  - Knowing the sender is not enough to make it safe to open
  - Virus creator can disguise the attachment to look like it is not executable
  - Remember the "Love Letter" virus!

Classifying Malicious Attacks

- How was the attack created?
- How was malicious code transported?
- What vulnerabilities were exploited?
- What damage did the attack cause?

Types of Damage

- Loss of data
- Loss of computer resources
- Lost time
- Loss of privacy
- Loss of confidentiality
- Monetary loss

Classification by Payload

- Szor, Chapter 8, has another set of criteria for classifying viruses by *payload*
- The payload is the malicious code that is delivered into the system by the virus
- Rather than categorizing by privacy, time loss, data loss, etc., the severity of the damage is the primary classifier

Classification by Payload cont'd.

- No payload
- Accidentally destructive payload
- Nondestructive payload
- Somewhat destructive payload
- Highly destructive payload

Classification by Payload cont'd.

- No payload
  - Virus just replicates
  - Creator might just be testing a concept: Can I infect systems in a certain way?
  - Creator often is playing a game with antivirus researchers; leaves a message in the body of the virus
  - More viruses in this category than in any other
  - Still wastes some resources

Classification by Payload cont’d.

- Accidentally destructive payload
  - “Stoned” virus was an example
  - Tried to save the disk boot sector, infect and replicate, then restore the boot sector
  - Accidentally copied the boot sector, on certain systems only, on top of useful data when saving it
  - Some users lost their file system entirely as a result

Classification by Payload cont'd.

- Nondestructive payload
  - Payload displays a message on the screen for a few seconds
  - No other action is taken
  - About half of all viruses are either "no payload" or "nondestructive payload"

Classification by Payload cont'd.

- Somewhat destructive payload
  - Some viruses try to disable a particular antivirus program, but attack nothing else
- HPS was a Windows 95 virus that only activated if you booted up on a Saturday; then it did a horizontal reversal of Windows bitmap files (see Szor, p. 300)

- Wazzu virus of 1996 randomly scrambled 3 words in documents, and inserted the word *wazzu* into sentences

Classification by Payload cont'd.

- Highly destructive payload
  - Includes the examples we already classified, such as loss of data, loss of privacy, DOS (denial of service), etc.
  - Also: *data diddlers*, which slowly change data on disk, eluding detection until damaged data has probably infected backup tapes
  - *Hardware destroyers*: e.g. 1998 Taiwanese virus, CIH, overwrote the flash BIOS of more than 10,000 PCs

Payload Question

- Why would a somewhat destructive payload sometimes be more damaging than a highly destructive payload?

Security Terminology

1. Virus: self-replicating code
2. Worm: a virus that replicates over a network
3. Time bomb: malicious code that awakens itself on a certain date and/or time
4. Logic bomb: malicious code that becomes active when certain conditions are met

Security Terminology cont'd.

5. Trojan Horse: code that seems to be benign and useful (e.g. a screen saver) that performs replication and/or malicious operations in the background
6. Mailer: a worm that emails itself to another user
7. Mass Mailer: a worm that emails itself to multiple recipients
8. Backdoor: hidden access method in software, known only to an attacker

Security Terminology cont’d.

9. Exploit: an attack that takes advantage of a specific vulnerability
10. Kit: a virus generator program
11. Flooder: program that generates a large amount of network traffic to a certain server
12. DOS (denial of service): an attack that bogs down a server with a generated workload, generally a network packet load from a flooder

Security Terminology cont’d.

13. Keylogger: a malicious program that captures keystrokes on an infected system, usually to steal passwords, credit card numbers, ATM PINs (personal identification numbers), etc.
14. Spyware: a background program that collects data on a computer user’s browsing and computing habits, often installed without explicit permission
15. Malware: any form of malicious software
16. Firewall: hardware and/or software used to enforce a network access policy by filtering out some packets before they get routed by the network router

Security Terminology cont'd.

17. Payload: the malicious code that performs operations other than replication, e.g. deleting files, modifying files, stealing passwords

- Notice that most of our key terms (e.g. virus, worm, mailer, mass mailer) only refer to the means of transport and replication, not the actual payload!
- Malware can lack any payload at all and still cause damage due to resource usage during replication, e.g. the Morris worm