6.857 Computer and Network Security November 06, 2001

Lecture Notes 17 : Viruses and Worms

Lecturer: Ron Rivest *Scribe: Hydari/Krishnamurthy/Yip/Yuditskaya*

1 Topics

- History
- Properties
- Internet worm & "I love you" Virus
- Scanners
- Polymorphic Viruses

2 History

- '81 - First virus "Elk Cloner," a virus for the Apple II computer. It transmitted itself by floppy disk.
- '83 - Term "virus" was coined by Len Adleman - Fred Cohen Thesis
- '88 - Internet worm
- '90s - Anti-virus software made its debut. The number of viruses exploded from 10s to hundreds, to thousands. Symantec emerged as one of the big anti-virus software companies.
- Today - The wars continue: Melissa, Code Red, I love you, Nimda, etc. Developing anti-virus software is a major industry, and viruses remain a major security problem. As yet, no great solutions have been found.

3 Properties

3.1 Viruses vs. Worms

1. Both replicate themselves
    - Gene Spafford - viruses and worms are "forms of artificial life."
    - Rod Brooks disagrees.

2. Worms - standalone programs. They execute themselves and run on their own, port themselves to other machines.
3. Viruses - infect other programs. They infect some other program, which they need in order to run.

3.2 Features That Viruses and Worms May Have

Boot Sector Viruses Disk has boot sector, starting location for boot sequence.

- If virus is in the boot sector, it runs right when the computer boots.
- "serious trouble"
- The Windows OS and virus scanners pay close attention to the boot sector.
- Maybe this is less important because of email viruses now.

TSR Terminate & stay resident viruses infect the OS in memory.

- The virus becomes part of the working operating system.
- The user can no longer trust the OS.
- The virus can pretend to do trusted jobs that the OS used to do.
- Worse yet, the virus can catch keyboard interrupts, thus retrieving passwords, for example.

Self Recognition Viruses don't infect the already sick host.

- This helps the virus stay below the radar in the system.
- This property can be used by virus scanners to detect it, however.

Stealth Viruses hide their own existence.

- For example, a boot sector virus could remain undetected by a disk read by storing what the original boot sector looked like and showing that previous copy during a disk read. I.e., disk reads only show preinfected data - can trick virus scanning programs.
- Can be more dangerous when devices become more intelligent.
- Firmware in hard drive, network card.

Obfuscation Makes the virus program complicated, to avoid being reverse engineered.

- People might try to decompile it to figure out what it's doing, for countermeasures.
- Lengthens the time to develop countermeasures.
- Obfuscation is often used for honorable purposes as well, especially by companies who want to protect the secrecy of their implementations.

Damage / Side Effects Example: A particularly nasty virus, "One-Half," slowly encrypts the hard drive as it's running, in a way that you don't know that it's happening because it's in control of the disk reads and writes: when it does a disk write, it writes the data in encrypted form, and then decrypts it on a disk read. So as far as the application is concerned, the disk is running fine, but when the virus is removed, the encryption key is lost, leaving a hard disk full of securely encrypted, and therefore irretrievable, data.

- Time based trigger: side effects conditional on time. For example, voting viruses would be triggered to change votes only on election day. Or it could be a part of the virus to escape detection. For example, a virus in slot machines could produce time-conditional side effects to stay "below the radar."
- DOS (Denial of Service) attacks: one of the side effects that we have seen a lot of recently is using viruses and other exploits to take over many machines, and then those become slaves for carrying out a denial of service attack.

Network aware, email aware Early viruses propagated by floppy; these days, viruses know they live in a context where machines can attach to networks and where certain email programs are being used, e.g. Microsoft Outlook. They know how to get from machine A to machine B using existing mechanisms of connectivity among hosts.

Polymorphic - a virus that doesn't look the same on program B as on program A. Any two instances of the virus may look different. Polymorphism in viruses makes it particularly hard for the anti-virus scanners to identify them because there's no fixed pattern of bytes you can look for.

4 Internet Worm (Nov. 2, 1988)

Today, there are around 60 million hosts connected to the Internet.

The Internet was very different back in 1988 – there were only about 60,000 hosts connected to the Internet. About 10% were infected by the Internet Worm. That was a big deal. A large part of the Internet got shut down, totally clogged up; this virus had a somewhat defective self-recognition feature. It was really the first major catastrophe over the Internet.

4.1 What techniques did Internet Worm use?

- fingerd buffer overflow: Primarily Unix machines were being taken over; fingerd is a daemon program which gives information about a given user. The input routine is stupidly written so that after reading input, it allocates a fixed size buffer for the input that could see it, and if you send it more input than that, then you would be overwriting parts of memory that are beyond the buffer. Thus, sending particular packets could get malicious code to run locally.
- sendmail: the program that handles email. It was a big, hairy, complicated program, and therefore had many bugs. One of these bugs was an option that allowed leaving debug mode on by default. Thus debug mode was enabled on most installed versions, which caused problems because default mode gave free access to the shell, a security breach. Thus, sending particular mail could get malicious code to run locally.
- passwords: the worm would read in the password file, which contained the hashes of all the passwords. Then, it would do a dictionary attack for each entry until it found the password. It could then access other user accounts.
  - Offline dictionary attack.

  - Look for other machines that have the same username/password.
  - Trusted hosts. Workgroup machines trust other machines, so users don't need to sign on more than once. /rhosts /etc/hosts.equiv

4.2 Lessons Learned

- Least Privilege.
- "We have met the enemy and he is us" - People who attack systems (through viruses, etc) are people who know the systems well, and are therefore usually insiders.
- Diversity is good.
- Backups are important.
- Defenses should be at the host level, not just at the network level - Firewalls don't work perfectly. (Crunchy outside, but soft inside.)
- Logged info is very important.

In 1999, there was a 6.857 assignment to exploit buffer overflows. The basic idea was to find a fixed sized buffer and give it an input which will result in an overflow. The adversary cleverly chooses the input such that it overwrites the return address to a code block of his choosing.

5 "I Love You" virus (May 2000)

- $100 million to $1 billion in damage
- An estimated 50%+ of US companies were infected
- Affected Win 98/NT
- Propagated over email, as an attachment with filename LOVELETTER.TXT.vbs. The *.vbs ending means "Visual Basic script executable." A double click executes the virus.
- Demonstrated that the Internet is akin to an urban situation vs. rural situation.

Remark: Virus writing is not very difficult.
Question: What did the "I love you" virus do?
Answer: It had no destructive payload. However, it did clog the hosts and networks. Professor Rivest's prognosis is that we will see more malicious and destructive viruses.

6 Countermeasures

(Virus may keep file size same, keep size of directory same...)

- Tripwire - designed to help aid in the detection of file modification. It does so by computing an MD5 hash of all files. Later, it compares hashes to files and see if files changed.
- Backups & Logs.
- Updating the OS to fix bugs – usually in the form of patches to be installed by the user. But how do you know you're getting the right update? How do you know that you're downloading "good" code? Need to digitally sign that piece of code with the manufacturer's SK. (also signed by CA) P, $\sigma_m(P), \sigma_{CA}(PK_m)$. But with the recent compromise of Microsoft's public key, even this can no longer be trusted.

6.1 Virus Scanners

Protecting against viruses. Can we make one that is really effective?

Naive Idea

Naive idea: find some "signature" of virus.

- Look for some particular byte sequence that supposedly identifies a given virus.
- Works for non-polymorphic viruses.
- Works if you can find such a sequence.
- Need to have few false positives: We don't want MS Word to always look like it's infected!
- Integration with OS & email clients
  - Want to be the gate keeper for all possible entrants into computer.
  - Want to be able to check if a file is good or bad before downloading it.
  - There is a problem if the email client supports encrypted email. (You don't know if it's a good or bad file until it's already inside.)

Lifecycle of a Virus

1. created
2. released

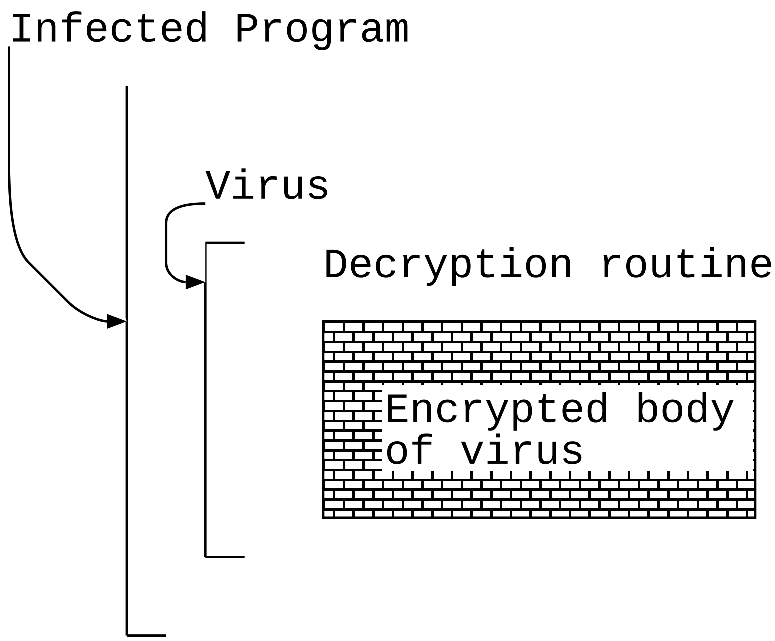

Figure 1: Structure of a Program Infected with a Polymorphic Virus

3. detected
4. reported
5. analyzed
6. sometimes an infected file can even be repaired! :)

Much of this can be done automatically now, as patches to the signature table. This is important because viruses are spreading more and more quickly; thus, speed in responding is very important.

But we still have problems with polymorphic viruses...

Polymorphic Virus [1]

How polymorphic viruses work:

They have just enough features to defeat a scanner.

Structure of a Polymorphic Virus
————————————-
—Fetch byte
—Decrypt

[1] For more info on polymorphic viruses, see the Symantec Web site. http://www.symantec.com/

—Store byte
—Repeat
—
—Body
—
—Mutation Engine – chooses new encryption key for next “mutation” of the virus.
———————————-

All sections can be obfuscated by instructions that don’t do anything. The next copy of the virus program can be completely different from its parent copy.

So how does a scanner detect polymorphic viruses?

- They are detected when the virus has decrypted itself.
- The scanner executes the virus in its own isolated virtual machine until it finds a copy of the virus that matches a bit signature in memory. In other words, the scanner runs the virus program, and checks for the fingerprint. This works if the virus is at the beginning of the program.
- But what if the virus embeds itself elsewhere in the program??? (e.g., Not at the beginning, or maybe in an IO instruction). This remains an unresolved problem.
- More desirable solutions are technologies that reduce the privileges of executables, as needed.