Malicious Code Analysis

- Malicious Code (Malware)
  - software that fulfills malicious intent of author
  - term often used equivalent with virus (due to media coverage)
  - however, many different types exist
  - classic viruses account for only 3% of malware in the wild
- There is a wide variety of different types of malicious code
  - viruses, worms, spyware, rootkits, Trojan horses, botnets
- Common characteristic
  - perform some unwanted activity on your system
  - usually only available as binary (important for analysis)

Malware

- Computer virus
  - A virus is a program that reproduces its own code by attaching itself to other executable files in such a way that the virus code is executed when the infected executable file is executed
- Computer worm
  - Spreads autonomously like a computer virus, but needs no host program that it can infect
- Trojan horse
  - A computer program that is hidden inside another program that serves a useful purpose

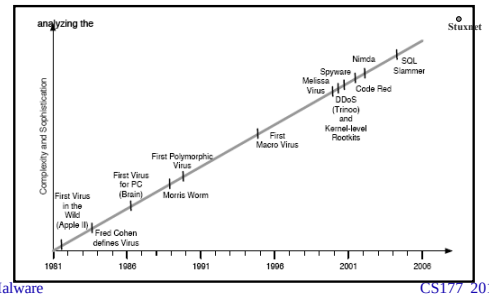
Malware (continued)

- Rootkit
  - Code introduced into system administration tools with the purpose of hiding the presence of an attacker on the system
- Spyware
  - Programs that monitor the behavior of users and steal private information, such as keystrokes or browsing habits
  - Often bundled with free software that explicitly states that spyware is installed on a user's machine
  - Information collected is sent back to the spyware distributer and used as a basis for targeted advertisements

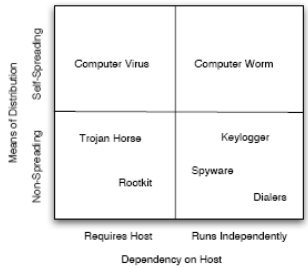
Malware (continued)

- Key-logger
  - Spyware that focuses on the recording of the keys that a user types
- Dialer
  - A computer program that creates a connection to the Internet or another computer network over the analogue phone or ISDN network
  - Increasing use of broadband Internet reduces this threat
  - Some new attacks on sophisticated cell phones
- Botnet
  - networks of remotely-controlled, compromised machines

History of Malware Development



Malicious Code Taxonomy



Reasons for Malware Prevalence

- Mixing data and code
  - violates important design property of secure systems
  - unfortunately very frequent
- Homogeneous computing base
  - Windows is just a very tempting target
- Unprecedented connectivity
  - easy to attack from safety of home
- Clueless user base
  - many targets available
- Malicious code has become profitable
  - compromised computers can be sold (e.g., spam relay, DoS)

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

Backdoors

- A **backdoor,** which is also sometimes called a **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
- When used in a normal way, this program performs completely as expected and advertised
- But if the hidden feature is activated, the program does something unexpected, often in violation of security policies, such as performing a privilege escalation
- Benign example: **Easter Eggs** in DVDs and software

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

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

Defenses against Insider Attacks

- Avoid single points of failure
- Use code walk-throughs
- Use archiving and reporting tools
- Limit authority and permissions
- Physically secure critical systems
- Monitor employee behavior
- Control software installations

Virus

- 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 Lifecycle

- Lifecycle
  - reproduce, infect, run payload
- Reproduction phase
  - viruses balance infection versus detection possibility
  - variety of techniques may be used to hide viruses
- Infection phase
  - difficult to predict when infection will take place
  - many viruses stay resident in memory
- Attack phase
  - e.g., deleting files, changing random data on disk
  - viruses often have bugs (poor coding) so damage can be done
    - Stoned virus expected 360K floppy, corrupted sectors – screwed up when 1.2M floppy or more than 96 files in root directory

Infection Strategies

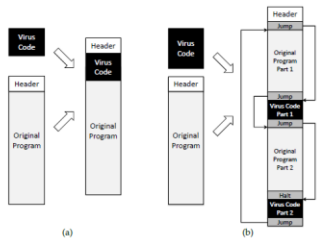
- Boot viruses
  - master boot record (MBR) of hard disk (first sector on disk)
  - boot sector of partitions
  - e.g., Pakistani Brain virus
  - rather old, but interest is growing again
    - diskless work stations, virtual machine virus (SubVirt)
- File infectors
  - simple overwrite virus (damages original program)
  - parasitic virus
    - append virus code and modify program entry point
  - cavity virus
    - inject code into unused regions of program code

Infection Strategies

- Entry Point Obfuscation
  - virus scanners quickly discovered to search around entry point
  - virus hijacks control later (after program is launched)
  - overwrite import table addresses
  - overwrite function call instructions
- Code Integration
  - merge virus code with program
  - requires disassembly of target
    - difficult task on x86 machines
  - W95/Zmist is a classic example for this technique

Degrees of Complication

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



Macro Viruses

- Many modern applications support macro languages
  - Microsoft Word, Excel, Outlook
  - macro language is powerful
  - embedded macros automatically executed on load
  - mail app. with Word as an editor
  - mail app. with Internet Explorer to render HTML

I made this program to all those people who want to write Word 2000 virii, but don't know what the hell to do.

Arms Race

- Existing defense and detection mechanisms struggle to keep up with the scale and sophistication of the attacks
- Search for detection features
  - what can be used to characterize and identify malicious code?
- Most current detection features are *syntax-based*
  - sequences of byte strings
  - sequences of instructions
  - regular expressions

Arms Race

- Heuristics
  - memory block is executed that was previously written
  - suspicious values in file (PE) header (e.g., incorrect size)
  - patched import address table
- Sandboxing
  - run untrusted applications in restricted environment
  - simplest variation, do not run as Administrator

Arms Race

- Particular problem posed by code obfuscation
- Polymorphic transformations encrypt the actual malware body and prepend a short decryption routine to the encrypted body
- *Syntactic representation* of metamorphic code creates different "versions" of code that look different but have the same semantics (i.e., do the same thing)
- Toolkits were created for doing this
  - Dark Avenger's Mutation Engine
    - viruses generated by this tool are easily found
  - ADMmutate for exploit shellcode

Polymorphism and Metamorphism

- Polymorphic viruses
  - payload is encrypted
  - using different key for each infection
  - makes static string analysis practically impossible
  - of course, encryption routine must be changed as well
  - otherwise, detection is trivial

Polymorphism and Metamorphism

- Metamorphic techniques
  - register renaming
  - dead code insertion
  - block reordering
  - command substitution

Chernobyl (CIH) Virus

```
5B 00 00 00 00      pop ebx
8D 4B 42            lea ecx, [ebx + 42h]
51                  push ecx
50                  push eax
50                  push eax
0F 01 4C 24 FE    sidt [esp - 02h]
5B                  pop ebx
83 C3 1C            add ebx, 1Ch
FA            cli
8B 2B               mov ebp, [ebx]
```

```
5B 00 00 00 00    8D 4B 42 51 50 50 0F 01 4C 24 FE 5B
83 C3 1C FA 8B 2B
```


Dead Code Insertion

```
5B 00 00 00 00      pop ebx
8D 4B 42            lea ecx, [ebx + 42h]
51                  push ecx
50                  push eax
90                  nop
50                  push eax
40                  inc eax
0F 01 4C 24 FE    sidt [esp - 02h]
48                  dec eax
5B                  pop ebx
83 C3 1C            add ebx, 1Ch
FA            cli
8B 2B               mov ebp, [ebx]
```

```
5B 00 00 00 00    8D 4B 42 51 50 90 50 40 0F 01 4C 24
FE 48 5B 83 C3 1C FA 8B 2B
```


Instruction Reordering

```
5B  00 00 00 00     pop ebx
EB  09              jmp <S1>
S2:
50                  push eax
0F 01 4C 24 FE      sidt [esp - 02h]
5B                  pop ebx
EB  07              jmp <S3>
S1:
8D 4B 42      lea ecx, [ebx + 42h]
51                  push ecx
50                  push eax
EB  F0              jmp <S2>
S3:
83 C3 1C      add ebx, 1Ch
FA            cli
8B 2B               mov ebp, [ebx]
```

```
5B 00 00 00 00    EB 09 50 0F 01 4C 24 FE 5B EB 07 8D
4B 42 51 50 EB F0 83 C3 1C FA 8B 2B
```


Instruction Substitution

```
5B 00 00 00 00      pop ebx
8D 4B 42            lea ecx, [ebx + 42h]
51                  push ecx
89 04 24            mov eax, [esp]
83 C4 04            add 04h, esp
50                  push eax
0F 01 4C 24 FE      sidt [esp - 02h]
83 04 24 0C         add 1Ch, [esp]
5B                  pop ebx
8B 2B               mov ebp, [ebx]
```

```
5B 00 00 00 00    8D 4B 42 51 89 04 24 83 C4 04 50 0F
01 4C 24 FE 83 04 24 0C 5B 8B 2B
```


Advanced Virus Defense

- Most virus techniques very effective against static analysis
- Thus, dynamic analysis techniques introduced
  - virus scanner equipped with emulation engine
  - executes actual instructions (no disassembly problems)
  - runs until polymorphic part unpacks actual virus
  - then, signature matching can be applied
  - emulation must be fast
  - Anubis (*anubis.iseclab.org)*
- Difficulties
  - virus can attempt to detect emulation engine
  - time execution, use exotic (unsupported) instructions, …
  - insert useless instructions in the beginning of code to deceive scanner

Computer Worms

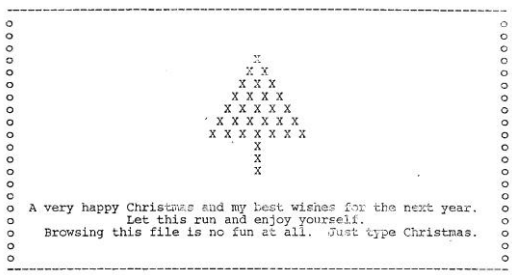
A self-replicating program able to propagate itself across networks, typically having a detrimental effect.
(Oxford English Dictionary)

- Worms either
  - exploit vulnerabilities that affect large number of hosts
  - send copies of worm body via email
- Difference from classic virus is *autonomous* spread over network
- Speed of spreading is constantly increasing
- Make use of techniques known by virus writers for long time

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

CHRISTMAs EXEC



Worm Components

- Target locator
  - how to choose new victims
- Infection propagator
  - how to obtain control of victim
  - how to transfer worm body to target system
- Life cycle manager
  - control different activities depending on certain circumstances
  - often time dependent
- Payload

Target Locator

- Email harvesting
  - consult address books (W32/Melissa)
  - files might contain email addresses
    - inbox of email client (W32/Mydoom)
    - Internet Explorer cache and personal directories (W32/Sircam)
  - even Google searches are possible
- Network share enumeration
  - Windows discovers local computers, which can be attacked
  - some worms attack everything, including network printers prints random garbage (W32/Bugbear)

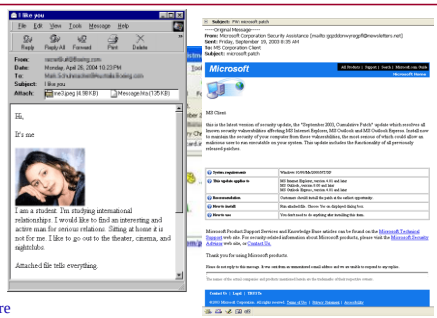
Target Locator

- Scanning
  - randomly generate IP addresses and send probes
  - interestingly, many random number generators flawed
    - static seed
    - not complete coverage of address space
  - scanning that favors local addresses (topological scanning)
  - some worms use hit-list with known targets (shorten initial phase)
- Service discovery and OS fingerprinting performed as well

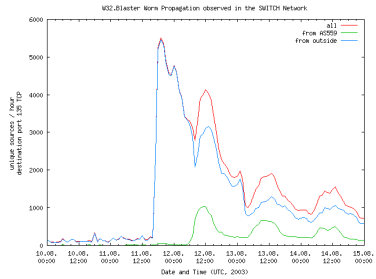
Email-Based Worms

- Often use social engineering techniques to get executed
  - fake from address
  - promise interesting pictures or applications
  - hide executable extension (.exe) behind harmless ones (.jpeg)
- Many attempt to hide from scanners
  - packed or zipped
  - sometimes even with password (ask user to unpack)
- Some exploit Internet Explorer bugs when HTML content is rendered
- Significant impact on SMTP infrastructure
- Speed of spread limited because humans are in the loop
  - can observe spread patterns that correspond to time-of-day

Email-Based Worms



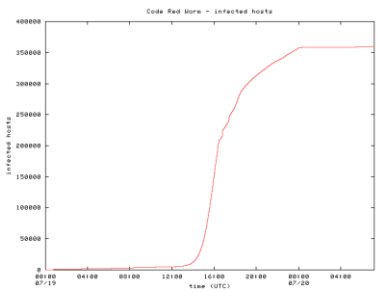
Email-Based Worms



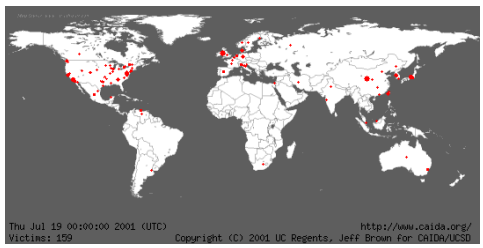
Exploit-Based Worms

- Require no human interaction
  - typically exploit well-known network services
  - can spread much faster
- Propagation speed limited either
  - by network latency
    worm thread has to establish TCP connection (Code Red)
  - by bandwidth
    worm can send (UDP) packets as fast as possible (Slammer)
- Spread can be modeled using classic disease model
  - worm starts slow (only few machines infected)
  - enters phase of exponential growth
  - final phase where only few uncompromised machines left

Exploit-Based Worms

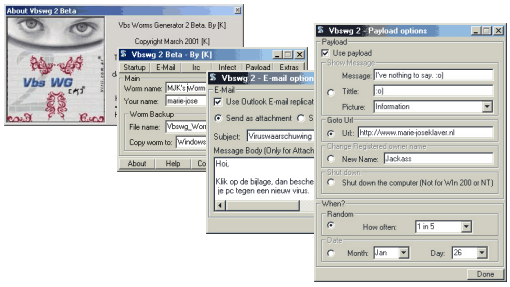


Code Red Infections



Geographic spread of Code Red worm

Worm Generators



Worm Defense

- Virus scanners
  - effective against email-based worms
  - email attachments can be scanned as part of mail processing
- Host level defense
  - mostly targeted at underlying software vulnerabilities
  - code audits
  - stack-based techniques
    - StackGuard, MS Visual C compiler extension
  - address space layout randomization (ASLR)
    - attempt to achieve diversity to increase protection

Worm Defense

- Network level defense
  - intrusion detection systems
    - scan for known attack patterns
    - automatic signature generation is active research area
  - rate limiting
    - allow only certain amount of outgoing connections
    - helps to contain worms that perform scanning
  - personal firewall
    - block outgoing SMTP connections (from unknown applications)

Malware Zombies

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

Botnets

- Recent trend in malicious code development
- Often part of payload that is downloaded as Trojan horse or part of a worm
- Definition of Bot

  An IRC user who is actually a program. On IRC, typically the robot provides some useful service. Examples are NickServ, which tries to prevent random users from adopting nicks already claimed by others.
- IRC (Internet Relay Chat)
  - instant message (communication service)
  - allows for many-to-many communication in channels

Botnets

- Bots
  - first bots were programs used for Internet Relay Chat (IRC)
  - react to events in IRC channels
  - typically offer useful services
  - malicious bots started to evolve
    - takeover wars to control certain IRC channels
    - often involved denial of service to force IRC net split
  - nowadays, term refers to remote program loaded on a computer after compromise
  - usage of IRC for command and control of these programs

Botnets

- Bot evolution
  - implementation of several commands (e.g., DDoS)
  - spreading mechanism to propagate further
  - other functionality possible
    - key logger
    - SOCKS proxy
    - spam relay
  - some bots are even open source
    - caused massive distribution and variations
    - SDBot, AgroBot
- Bots can be incorporated in network of compromised machines
  → Botnets (sizes up to hundreds of thousands)

Botnets

- Rapid development of command and control systems
  - initially, IRC-based
  - moved to HTTP
    and peer-to-peer protocols (for example, Storm)
- New propagation vectors
  - browser (drive-by) downloads
  - mass compromise of web sites and `iframe` redirects
- Goals
  - stay undetected
  - make money (spam, DoS, data theft)

Botnet Defense

- Honeypots
  - vulnerable computer that serves no purpose other than to attract attackers and study their behavior in controlled environments
  - when honeypot is compromised, bot logs into botnet
  - allows defender to study actions of botnet owners
- Attack command and control infrastructure
  - take command and control channel off-line
  - when dynamic DNS is used for central command server, route traffic to black hole

Trojan Horse

- Trojan horse is a malicious program that is disguised as legitimate software
  - software may look useful or interesting (or at the very least harmless)
  - term derived from the classical myth of the Trojan Horse

Rootkits

- Tools used by attackers after compromising a system
  - hide presence of attacker
  - allow for return of attacker at later date (trap door)
  - gather information about environment
  - attack scripts for further compromises
- Traditionally trojaned set of user-space applications
  - system logging (syslogd)
  - system monitoring (ps, top)
  - user authentication (login, sshd)

Kernel Rootkits

- Kernel-level rootkits
  - kernel controls view of system for user-space applications
  - malicious kernel code can intercept attempts by user-space detector to find rootkits
- Modifies kernel data structures
  - process listing
  - module listing
- Intercepts requests from user-space applications
  - system call boundary
  - VFS fileops struct

Linux Kernel Rootkits

- Linux kernel exports well-defined interface to modules
- Examples of legitimate operations
  - registering device with kernel
  - accesses to devices mapped into kernel memory
  - overwriting exported function pointers for event callbacks
- Kernel rootkits violate these interfaces
- Examples of illegal operations
  - replacing system call table entries (knark)
  - replacing VFS fileops (adore-ng)

Windows Kernel Rootkits

- Sony rootkit filters out any files/directories, processes and registry keys that contain `$sys$`

Windows Kernel Rootkits

- System call dispatcher
  - uses system service dispatch table (SSDT)
  - Windows NT kernel equivalent to system call table
  - entries can be manipulated to re-route call to custom function

`ZwCreateFile`
- used to create or open file

`ZwQueryDirectoryFile`
- used to list directory contents (i.e. list subdirectories and files)

`ZwQuerySystemInformation`
- used to get the list of running processes (among other things)

`ZwEnumerateKey`
- used to list the registry keys below a given key

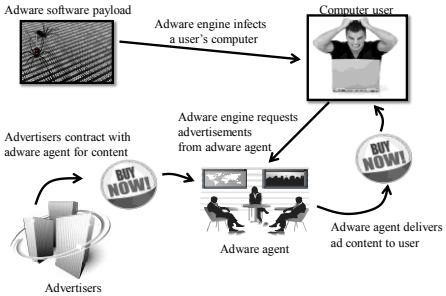
Rootkit Defense

- `tripwire`
  - user-space integrity checker
- `chkrootkit`
  - user-space, signature-based detector
- `kstat, rkstat, StMichael`
  - kernel-space, signature-based detector
  - implemented as kernel modules or use /dev/kmem
- Limitations
  - typically, rootkit must be loaded in order to detect it
  - thus, detectors can be thwarted by kernel-level rootkit
  - also suffer from limitations of signature-based detection

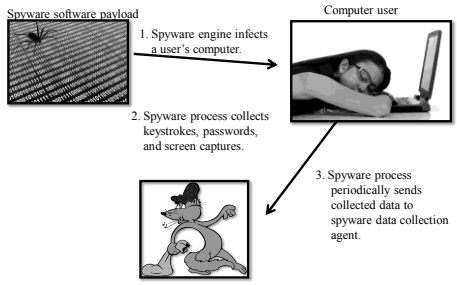
Rootkit Defense

- Kernel rootkits
  - have complete control over operating system
  - operating system is part of trusted computing base, thus applications can be arbitrarily fooled
  - this includes all rootkit or Trojan detection mechanisms
  - at best, an arms race can be started
- Proposed solutions
  - trusted computing platform
    - can enforce integrity of operating system
  - smart cards
    - attacker can not influence computations on card, but still has full control of computations performed on machine and information displayed on screen

Adware



Spyware



Malware and Vulnerable Software

- Malicious software (Malware) and benign software that can be exploited to perform malicious actions (Badware) are two facets of the same problem
  → execution of unwanted code
- Malware
  - viruses, worms, Trojan horses, rootkits, and spyware are evolving to become resilient to eradication and to evade detection
- Badware
  - services and applications (especially web-based) are vulnerable to a wide range of attacks, some of which are novel

Conclusions

- Malware
  - sophisticated technology developed for more than 25 years
  - combined with automatic spread mechanisms
  - tools to generate malware significantly lower technological barrier
- Defense Techniques
  - mostly reactive
  - using signatures to detect known instances
  - use best programming practice for application development, educate employees, keep infrastructure well maintained (patched)