Malware
Spring 2012
Presented by Shishir Nagaraja
Welcome to the zoo

- What malware are
- How do they infect hosts
- How do they hide
- How do they propagate
- Zoo visit!
- How to detect them
- Worms
What is a malware?

A Malware is a set of instructions that run on your computer and make your system do something that an attacker wants it to do.
What it is good for?

- Steal personal information
- Delete files
- Click fraud
- Steal software serial numbers
- Use your computer as relay
The Malware Zoo

- Virus
- Backdoor
- Trojan horse
- Rootkit
- Scareware
- Adware
- Worm
What is a Virus?

A program that can infect other programs by modifying them to include a, possibly evolved, version of itself.

Fred Cohen 1983
Some Virus Type

- **Polymorphic**: uses a polymorphic engine to mutate while keeping the original algorithm intact (packer)
- **Metamorphic**: Change after each infection
What is a trojan

A trojan describes the class of malware that appears to perform a desirable function but in fact performs undisclosed malicious functions that allow unauthorized access to the victim computer.

Wikipedia
What is a rootkit

A root kit is a component that uses stealth to maintain a persistent and undetectable presence on the machine

Symantec
What is a worm

A computer worm is a self-replicating computer program. It uses a network to send copies of itself to other nodes and do so without any user intervention.
Almost 30 years of Malware
History

- 1981 First reported virus: Elk Cloner (Apple 2)
- 1983 Virus get defined
- 1986 First PC virus MS DOS
- 1988 First worm: Morris worm
- 1990 First polymorphic virus
- 1998 First Java virus
- 1998 Back orifice
- 1999 Melissa virus
- 1999 Zombie concept
- 1999 Knark rootkit
- 2000 love bug
- 2001 Code Red Worm
- 2001 Kernel Intrusion System
- 2001 Nimda worm
- 2003 SQL Slammer worm

Melissa spread by email and share
Knark rootkit made by creed demonstrate the first ideas
love bug vb script that abused a weakness in outlook
Kernel intrusion by optyx gui and efficient hiding mechanisms
At the end of 2011, this number was 26 million!
Malware Repartition

- Trojan: 74%
- Worm: 13%
- Other: 9%
- Adware: 1%
- Spyware: 3%
Infection methods
Outline

• What malware are
• How do they infect hosts
• How do they propagate
• Zoo visit!
• How to detect them
• Worms
What to Infect

• Executable
• Interpreted file
• Kernel
• Service
• MBR
• Hypervisor
Overwriting malware
prepending malware
appending malware
Cavity malware

Malware

Targeted Executable

Infected host Executable
Multi-Cavity malware
Packers

<table>
<thead>
<tr>
<th>Packer</th>
<th>Payload</th>
</tr>
</thead>
<tbody>
<tr>
<td>Malware</td>
<td>Infected host Executable</td>
</tr>
</tbody>
</table>

Packers are a type of malware that encrypts or otherwise protects the payload before delivering it to the infected host.
Packer functionalities

- Compress
- Encrypt
- Randomize (polymorphism)
- Anti-debug technique (int / fake jmp)
- Add-junk
- Anti-VM
- Virtualization
Auto start

- Folder auto-start: C:\Documents and Settings\[user_name]\Start Menu\Programs\Startup
- Win.ini: run=[backdoor]" or "load=[backdoor]".
- System.ini: shell="myexplorer.exe"
- Wininit
- Config.sys
Auto start cont.

- Assign known extension (.doc) to the malware
- Add a Registry key such as `HKCU\SOFTWARE\Microsoft\Windows \CurrentVersion\Run`
- Add a task in the task scheduler
- Run as service
Unix autostart

- Init.d
- /etc/rc.local
- .login .xsession
- crontab
- crontab -e
- /etc/crontab
Macro virus

- Use the builtin script engine
- Example of call back used (word)
  - AutoExec()
  - AutoClose()
  - AutoOpen()
  - AutoNew()
Document based malware

- MS Office
- Open Office
- Acrobat
Userland root kit

- Perform
  - login
  - sshd
  - passwd
- Hide activity
  - ps
  - netstat
  - ls
  - find
  - du
Subverting the Kernel

Kernel task

• Process management
• File access
• Memory management
• Network management

What to hide

➡ Process
➡ Files
➡ Network traffic
Kernel rootkit

Hardware:
HD, keyboard, mouse, NIC, GPU
Subverting techniques

- Kernel patch
- Loadable Kernel Module
- Kernel memory patching (/dev/kmem)
Windows Kernel

Win32 subsystem DLLs
User32.dll, Gdi32.dll and Kernel32.dll

Other Subsystems
(OS/2 Posix)

Ntdll.dll

ntoskrnl.exe

Executive
Underlying kernel

Hardware Abstraction Layer (HAL.dll)

Hardware
Kernel Device driver

Win32 subsystem DLLs

Ntdll.dll

ntoskrnl.exe

Interrupt Hook
System service dispatcher
System service dispatch table
New pointer
Driver Overwriting functions
Driver Replacing Functions
MBR/Bootkit

Bootkits can be used to avoid all protections of an OS, because OS consider that the system was in trusted stated at the moment the OS boot loader took control.
BIOS | MBR | VBS | NT Boot Sector
---|---|---|---
WINLOAD.EXE | BOOTMGR.EXE | Windows 7 kernel HAL.DLL
Vboot

• Work on every Windows (vista,7)
• 3ko
• Bypass checks by letting them run and then do in-flight patching
• Communicate via ping
Hypervisor rootkit
Hypervisor rootkit

- Rogue app
- Host OS
- Virtual machine monitor
- Target OS
- App
- App
- Hardware
Propagation Vector
Vector
Vector
Vector
Vector
Vector
Vector
Outline

- What malware are
- How do they infect hosts
- How do they propagate
- Zoo visit!
- How to detect them
- Worms
Shared folder
Email propagation

Read other people's SMS with
http://www.spyonmobiles.com

250 OK 9C/62-06241-E7AFFE
QUIT
221 serin.channel4.local closing
QUIT

Free program for reading sms
http://www.spyonmobiles.com

.250 2.0.0 n3FF1hvj004993 Message accepted for
delivery
QUIT
221 2.0.0 chiesmta1-4.messageone.com closing
connection
QUIT

from pandalab blog
Valentine day ...
Email again

Are you interested in reading other people’s sms?

Get Your Free 30-Day Trial!

Do you want to test your partner or just to read somebody's SMS? This program is exactly what you need then! It's so easy! You don't need to install it at the mobile phone of your partner.

Just download the program and you will able to read all SMS when you are online. Be aware of everything! This is an extremely new service!

http://[Removed].com/freetrial.exe

Download Free Trial
© SMS Spy. All rights reserved
Fake codec
Fake antivirus
Hijack you browser

from pandalab blog
Fake page!
P2P Files

- Popular query
- 35.5% are malwares (Kalafut 2006)
Backdoor
Basic

Infected Host

TCP

Attacker
Reverse

Infected Host \(\rightarrow\) TCP \(\rightarrow\) Attacker
covert

Infected Host ▸ ICMP ▷ Attacker
Rendez-vous backdoor

- RDV Point
- Infected Host
- Attacker
Bestiary
Outline

• What malware are
• How do they infect hosts
• How do they propagate
• **Zoo visit!**
• How to detect them
• Worms
Adware
Modern stuff

- Spam campaigns
- Phishing sites
- Key loggers
- Social malware
Netbus

- 1998
- Used for “prank”
Symantec pcAnywhere
Browser Toolbar ...
Toolbar again

Welcome!

The Oldface community toolbar is now installed.

Now you'll always have the best of our site delivered right to your browser. You'll get our latest news, links, alerts, and more. It's a great way to stay connected!

Check out the Toolbar Options, where you can choose these cool components and more...

Web search and more!

Radio Player

Stop

Volume

Play/Pause

Add Stations

My Stations

Local Stations

Station.ca

Internet
Ransomware

- Trj/SMSlock.A
- Russian ransomware
- April 2009

To unlock you need to send an SMS with the text 4121800286 to the number 3649. Enter the resulting code. Any attempt to reinstall the system may lead to loss of important information and computer damage.
Detection
Outline

• What malware are
• How do they infect hosts
• How do they propagate
• Zoo visit!
• How to detect them
• Worms
Anti-virus

- Analyze system behavior
- Analyze binary to decide if it a virus
- Type :
  - Scanner
  - Real time monitor
Impossibility result

- It is not possible to build a perfect virus/malware detector (Cohen)
Impossibility result

- Diagonal argument
- P is a perfect detection program
- V is a virus
- V can call P
  - if P(V) = true -> halt
  - if P(V) = false -> spread
Virus signature

- Find a string that can identify the virus
- Fingerprint like
Heuristics

- Analyze program behavior
- Network access
- File open
- Attempt to delete file
- Attempt to modify the boot sector
Checksum

- Compute a checksum for
  - Good binary
  - Configuration file
- Detect change by comparing checksum
- At some point there will more malware than “goodware” ...
Sandbox analysis

• Running the executable in a VM
• Observe it
  • File activity
  • Network
  • Memory
Dealing with Packer

- Launch the exe
- Wait until it is unpack
- Dump the memory
Worms
Outline

• What malware are
• How do they infect hosts
• How do they propagate
• Zoo visit!
• How to detect them
• Worms
A worm is self-replicating software designed to spread through the network
- Typically, exploit security flaws in widely used services
- Can cause enormous damage
  - Launch DDOS attacks, install bot networks
  - Access sensitive information
  - Cause confusion by corrupting the sensitive information

Worm vs Virus vs Trojan horse
- A virus is code embedded in a file or program
- Viruses and Trojan horses rely on human intervention
- Worms are self-contained and may spread autonomously
Cost of worm attacks

- **Morris worm, 1988**
  - Infected approximately 6,000 machines
    - 10% of computers connected to the Internet
  - Cost ~ $10 million in downtime and cleanup

- **Code Red worm, July 16 2001**
  - Direct descendant of Morris’ worm
  - Infected more than 500,000 servers
    - Programmed to go into infinite sleep mode July 28
  - Caused ~ $2.6 Billion in damages,

- Love Bug worm: $8.75 billion

Statistics: Computer Economics Inc., Carlsbad, California
Internet Worm (First major attack)

- Released November 1988
  - Program spread through Digital, Sun workstations
  - Exploited Unix security vulnerabilities
    - VAX computers and SUN-3 workstations running versions 4.2 and 4.3 Berkeley UNIX code

Consequences
- No immediate damage from program itself
- Replication and threat of damage
  - Load on network, systems used in attack
  - Many systems shut down to prevent further attack
Some historical worms of note

<table>
<thead>
<tr>
<th>Worm</th>
<th>Date</th>
<th>Distinction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Morris</td>
<td>11/88</td>
<td>Used multiple vulnerabilities, propagate to “nearby” sys</td>
</tr>
<tr>
<td>ADM</td>
<td>5/98</td>
<td>Random scanning of IP address space</td>
</tr>
<tr>
<td>Ramen</td>
<td>1/01</td>
<td>Exploited three vulnerabilities</td>
</tr>
<tr>
<td>Lion</td>
<td>3/01</td>
<td>Stealthy, rootkit worm</td>
</tr>
<tr>
<td>Cheese</td>
<td>6/01</td>
<td>Vigilante worm that secured vulnerable systems</td>
</tr>
<tr>
<td>Code Red</td>
<td>7/01</td>
<td>First sig Windows worm; Completely memory resident</td>
</tr>
<tr>
<td>Walk</td>
<td>8/01</td>
<td>Recompiled source code locally</td>
</tr>
<tr>
<td>Nimda</td>
<td>9/01</td>
<td>Windows worm: client-to-server, c-to-c, s-to-s, …</td>
</tr>
<tr>
<td>Scalper</td>
<td>6/02</td>
<td>11 days after announcement of vulnerability; peer-to-peer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>network of compromised systems</td>
</tr>
<tr>
<td>Slammer</td>
<td>1/03</td>
<td>Used a single UDP packet for explosive growth</td>
</tr>
</tbody>
</table>

Kienzle and Elder
Increasing propagation speed

Code Red, July 2001
- Affects Microsoft Index Server 2.0,
  - Windows 2000 Indexing service on Windows NT 4.0.
  - Windows 2000 that run IIS 4.0 and 5.0 Web servers
- Exploits known buffer overflow in Idq.dll
- Vulnerable population (360,000 servers) infected in 14 hours

SQL Slammer, January 2003
- Affects in Microsoft SQL 2000
- Exploits known buffer overflow vulnerability
  - Server Resolution service vulnerability reported June 2002
  - Patched released in July 2002 Bulletin MS02-39
- Vulnerable population infected in less than 10 minutes
Code Red

- Initial version released July 13, 2001
  - Sends its code as an HTTP request
  - HTTP request exploits buffer overflow
  - Malicious code is not stored in a file
    - Placed in memory and then run
- When executed,
  - Worm checks for the file C:\Notworm
    - If file exists, the worm thread goes into infinite sleep state
  - Creates new threads
    - If the date is before the 20th of the month, the next 99 threads attempt to exploit more computers by targeting random IP addresses
Code Red of July 13 and July 19

- Initial release of July 13
  - 1\textsuperscript{st} through 20\textsuperscript{th} month: Spread
    - via random scan of 32-bit IP addr space
  - 20\textsuperscript{th} through end of each month: attack.
    - Flooding attack against 198.137.240.91 (www.whitehouse.gov)
  - Failure to seed random number generator $\Rightarrow$ linear growth

  - White House responds to threat of flooding attack by changing the address of www.whitehouse.gov
  - Causes Code Red to die for date $\geq$ 20\textsuperscript{th} of the month.
  - But: this time random number generator correctly seeded
Infection rate
Measuring activity: network telescope

- Monitor cross-section of Internet address space, measure traffic
  - “Backscatter” from DOS floods
  - Attackers probing blindly
  - Random scanning from worms
- LBNL’s cross-section: 1/32,768 of Internet
- UCSD, UWisc’s cross-section: 1/256.
Spread of Code Red

Network telescopes estimate of # infected hosts: 360K. (Beware DHCP & NAT)

For chinese IIS server: create 600 threads and spread for 48 hours, otherwise create 300 threads and spread for 24 hours

Note: larger the vulnerable population, faster the worm spreads.

That night (⇒ 20th), worm dies …
...
... except for hosts with inaccurate clocks!

It just takes one of these to restart the worm on August 1st …
Return of Code Red Worm

New Hosts Per Minute

Hours (PDT) Since Midnight, July 31

July 31, 2001

August 1, 2001

Slides: Vern Paxson
Code Red 2

- Comment in code: “Code Red 2.”
  - But in fact completely different code base.
- Payload: a root backdoor, resilient to reboots.
- Bug: crashes NT, only works on Windows 2000.
- Localized scanning: prefers nearby addresses.


Slides: Vern Paxson
Striving for Greater Virulence: Nimda

- Released September 18, 2001.
- Multi-mode spreading:
  - attack IIS servers via infected clients
  - email itself to address book as a virus
  - copy itself across open network shares
  - modifying Web pages on infected servers w/ client exploit
  - scanning for Code Red II backdoors (!)
- Worms form an ecosystem!
- Leaped across firewalls.

Slides: Vern Paxson
Code Red 2 kills off Code Red 1

CR 1 returns thanks to bad clocks

Code Red 2 settles into weekly pattern

Nimda enters the ecosystem

Code Red 2 dies off as programmed

Days Since July 18, 2001

Slides: Vern Paxson
How do worms propagate?

- **Scanning** worms: Worm chooses "random" address
- **Coordinated** scanning: Different worm instances scan different addresses
- **Flash** worms
  - Assemble tree of vulnerable hosts in advance, propagate along tree
    - Not observed in the wild, yet
    - Potential for 106 hosts in < 2 sec! [Staniford]
- **Meta-server** worm: Ask server for hosts to infect (e.g., Google for "powered by phpbb")
- **Topological** worm: Use information from infected hosts (web server logs, email address books, config files, SSH "known hosts")
- **Contagion** worm: Propagate parasitically along with normally initiated communication
slammer

- 01/25/2003
- Vulnerability disclosed: 25 June 2002
- Better scanning algorithm
- UDP Single packet: 380 bytes
Slammer propagation

Aggregated Scans/Second in the first 5 minutes based on Incoming Connections To the WAIL Tarpit

- X-axis: Minutes After the Initial Outbreak
- Y-axis: Millions of Packets Per Second
Number of scans/sec
Packet loss
A server view
Consequences

- ATM systems not available
- Phone network overloaded (no 911!)
- 5 DNS root down
- Planes delayed
Worm Detection and Defense

Detect via honeyfarms: collections of “honeypots” fed by a network telescope.

- Any outbound connection from honeyfarm = worm.
  (at least, that’s the theory)
- Distill signature from inbound/outbound traffic.
- If telescope covers N addresses, expect detection when worm has infected 1/N of population.

Thwart via scan suppressors: network elements that block traffic from hosts that make failed connection attempts to too many other hosts

- 5 minutes to several weeks to write a signature
- Several hours or more for testing
Need for automation

- Current threats can spread faster than defenses can react
- Manual capture/analyze/signature/rollout model too slow

Slide: Carey Nachenberg, Symantec
Signature inference

Challenge

- need to automatically learn a content “signature” for each new worm – potentially in less than a second!

Some proposed solutions

- Singh et al, Automated Worm Fingerprinting, OSDI ’04
- Kim et al, Autograph: Toward Automated, Distributed Worm Signature Detection, USENIX Sec ‘04
Signature inference

- Monitor network and look for strings common to traffic with worm-like behavior
  - Signatures can then be used for content filtering

---

**Packet Header**

SRC: 11.12.13.14 9090 3920 5000
DST: 132.239.13.24
Prot: TCP

**Packet Payload (Content)**

- Kibvu.B signature captured by Earlybird on May 14th, 2004
Content sifting

Assume there exists some (relatively) unique invariant bitstring $W$ across all instances of a particular worm (true today, not tomorrow...)

Two consequences

- **Content Prevalence**: $W$ will be more common in traffic than other bitstrings of the same length
- **Address Dispersion**: the set of packets containing $W$ will address a disproportionate number of distinct sources and destinations

**Content sifting**: find $W$’s with high content prevalence and high address dispersion and drop that traffic
Observation:
High-prevalence strings are rare

Only 0.6% of the 40 byte substrings repeat more than 3 times in a minute

(Stefan Savage, UCSD *)
The basic algorithm

Detector in network

Prevalence Table

Address Dispersion Table
Sources
Destinations

(Stefan Savage, UCSD *)
Detector in network

Prevalence Table

Address Dispersion Table Table

Sources

1 (A)

Destinations

1 (B)

(Stefan Savage, UCSD *)
Prevalence Table

Detector in network

Address Dispersion Table

Sources       Destinations

1 (A)       1 (B)
1 (C)       1 (A)

(Stefan Savage, UCSD *)
### Prevalence Table

<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>1</th>
</tr>
</thead>
</table>

### Address Dispersion Table

<table>
<thead>
<tr>
<th>Sources</th>
<th>Destinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 (A,B)</td>
<td>2 (B,D)</td>
</tr>
<tr>
<td>1 (C)</td>
<td>1 (A)</td>
</tr>
</tbody>
</table>

(Stefan Savage, UCSD *)
### Detector in network

#### Prevalence Table

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

#### Address Dispersion Table

<table>
<thead>
<tr>
<th>Sources</th>
<th>Destinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A, B, D)</td>
<td>(B, D, E)</td>
</tr>
<tr>
<td>1</td>
<td>1'</td>
</tr>
</tbody>
</table>

(Stefan Savage, UCSD *)
Challenges

- **Computation**
  - To support a 1Gbps line rate we have 12us to process each packet, at 10Gbps 1.2us, at 40Gbps…
    - Dominated by memory references; state expensive
  - Content sifting requires looking at every byte in a packet

- **State**
  - On a fully-loaded 1Gbps link a naïve implementation can easily consume 100MB/sec for table
  - Computation/memory duality: on high-speed (ASIC) implementation, latency requirements may limit state to on-chip SRAM

*(Stefan Savage, UCSD *)