

Virus Analysis

Techniques, Tools, and Research Issues

Part IV: Analysis Techniques - Advanced

Michael Venable
Arun Lakhotia

University of Louisiana at Lafayette, USA

Malware Analysis Techniques: Advanced

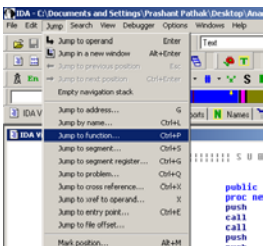
Client-Server Interaction

Debugging the Backdoor

- # Locate the Backdoor
- # Set Breakpoints
- # Prepare Client
- # Run Worm in Debugger
- # Run Client

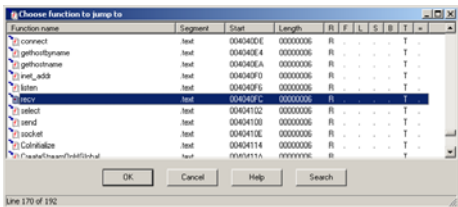
Locate the Backdoor

✦ Use IDAPro's "Jump To function..." option to find functions of interest



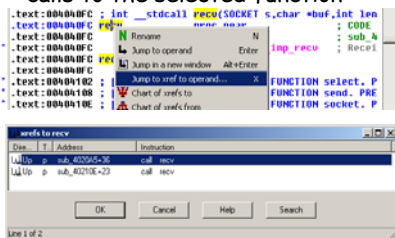
Locate the Backdoor

✦ Look for network-related functions



Locate the Backdoor

✦ Use "Jump to xref to operand..." to find calls to the selected function



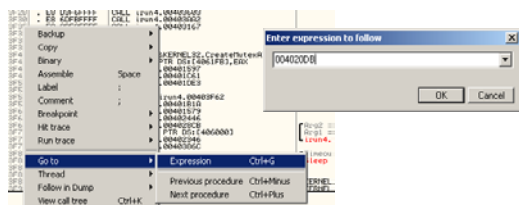
Locate the Backdoor

✪ Make note of the function call addresses

```
.text:00401210 push    eax                ; ucr  
.text:00401212 push    [ebp+5]           ; 5  
.text:00401214 call    recv              ; arg_0  
.text:00401216 test    eax, eax          ; arg_0  
.text:00401218 call    [0040200B]        ; arg_1  
.text:0040121A call    to_recv:0040200B ; arg_2  
.text:0040121C  
.text:0040121E
```

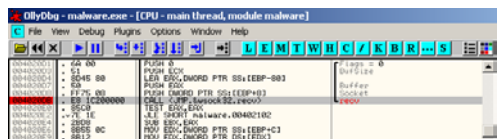
Set Breakpoints

✪ Locate the function calls in OllyDbg using "Go to Expression"



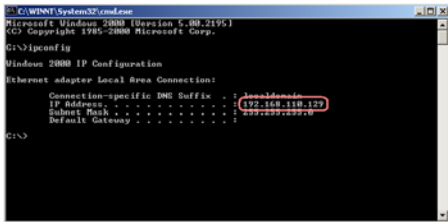
Set Breakpoints

✪ Press F2 to set a breakpoint



Prepare Client

✦ Retrieve IP address of infected machine



```
C:\WINNT\System32\cmd.exe
Microsoft Windows [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\>ipconfig

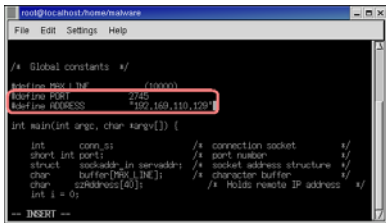
Windows 2000 IP Configuration

Ethernet adapter Local Area Connection:

    Connection-specific DNS Suffix  . : .
    IP Address. . . . . : 192.168.110.127
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . :
```

Prepare Client

✦ Configure port number & IP address in client



```
root@localhost/home/malware
File Edit Settings Help

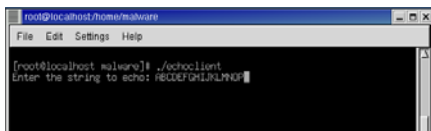
/* Global constants */
#define HOST_IP 192.168.110.129
#define PORT 2745
#define ADDRESS "192.168.110.129"

int main(int argc, char *argv[]) {
    int sock; /* connection socket */
    struct sockaddr_in serv_addr; /* port number */
    char *buffer[256]; /* socket address structure */
    char *szAddress[40]; /* character buffer */
    int i = 0; /* Holds remote IP address */

    -- INSERT --
```

Sending Data to Worm

✦ Run the worm and client
✦ Send input to worm



```
root@localhost/home/malware
File Edit Settings Help

[root@localhost malware]# ./echoclient
Enter the string to echo: rectorDLIhJhKf
```

Debugging

⌘ Make note of how the worm processes the input

```
00402910 . 9656 40 JNC BYTE PTR DS:[ESI+2]
00402911 . 75 14 JNC SHORT Ltrn4_00402920
00402912 . 8B7E 01 FF JMP BYTE PTR DS:[ESI+1],0FFF
00402913 . 75 14 JNC SHORT Ltrn4_00402920
00402914 . 661B37E 02 FF CMP WORD PTR DS:[ESI+2],0FFF
00402915 . 75 00 JNC SHORT Ltrn4_00402920
00402916 . FF75 FC PUSH DWORD PTR SS:[EBP-4]
00402917 . FF75 06 PUSH DWORD PTR SS:[EBP+0]
00402918 . E9 71FBFFFF CALL Ltrn4_00402948
00402919 . E9 02 JMP SHORT Ltrn4_00402920
0040291A . EB 02 JMP SHORT Ltrn4_00402948
0040291B . FF75 08 PUSH DWORD PTR SS:[EBP+0]
0040291C . E9 90170000 CALL Ltrn4_0040293C; closesocket()
0040291D . FF75 FC PUSH DWORD PTR SS:[EBP-4]
0040291E . E9 87E6FFFF CALL Ltrn4_00402948
0040291F . EB 02 JMP SHORT Ltrn4_00402948
Stack DS:1019FFA9J42 ("B")
```

Debugging

⌘ Control the flow of the worm by modifying register and flag values

```
00402910 . 9656 40 JNC BYTE PTR DS:[ESI+2]
00402911 . 75 14 JNC SHORT Ltrn4_00402920
00402912 . 8B7E 01 FF JMP BYTE PTR DS:[ESI+1],0FFF
00402913 . 75 14 JNC SHORT Ltrn4_00402920
00402914 . 661B37E 02 FF CMP WORD PTR DS:[ESI+2],0FFF
00402915 . 75 00 JNC SHORT Ltrn4_00402920
00402916 . FF75 FC PUSH DWORD PTR SS:[EBP-4]
00402917 . FF75 06 PUSH DWORD PTR SS:[EBP+0]
00402918 . E9 71FBFFFF CALL Ltrn4_00402948
00402919 . E9 02 JMP SHORT Ltrn4_00402920
0040291A . EB 02 JMP SHORT Ltrn4_00402948
0040291B . FF75 08 PUSH DWORD PTR SS:[EBP+0]
0040291C . E9 90170000 CALL Ltrn4_0040293C; closesocket()
0040291D . FF75 FC PUSH DWORD PTR SS:[EBP-4]
0040291E . E9 87E6FFFF CALL Ltrn4_00402948
0040291F . EB 02 JMP SHORT Ltrn4_00402948
Stack DS:1019FFA9J42 ("B")
```

Debugging

- ⌘ Press F7 to execute one instruction
- ⌘ Press F8 to execute a function, without entering it
- ⌘ Press CTRL+F9 to jump to end of current function

Debugging

- # 9th byte is compared to 0x0 and 0x0B
 - Must be greater than 0x0
 - Must be less to 0x0B

```

00402556 | .F7 08      | PUSH DWORD PTR DS:[EBP+8]
00402558 | .FF 75 0C   | CALL DWORD PTR DS:[EBP+4]
00402559 | .FF 75 0C   | PUSH DWORD PTR SS:[EBP+4]
0040255A | .E8 BCFEFFFF | CALL Ixun4.00402564
0040255B | .8B BCFEFFFF | CMP BYTE PTR SS:[EBP-152],0
0040255C | .76 09      | JBE SHORT Ixun4.0040255D
0040255D | .80BD CFEFFFF | CMP BYTE PTR SS:[EBP-152],0B
0040255E | .72 05      | JNB SHORT Ixun4.00402557
0040255F | .59 200000  | JMP Ixun4.00402559
00402560 | .6A 05      | PUSH 5
00402561 | .6A 05      | PUSH 5
00402562 | .F8 000000  | PUSH 0
Stack SS:[011CFE86]49 (*'1')

```

Debugging

- # Up to 200 bytes are read or until 0x0 is read

```

00402167 | .50         | PUSH EBX
00402168 | .FF 75 08   | PUSH DWORD PTR SS:[EBP+8]
00402169 | .C8 400000  | CALL Ixun4.00400000
0040216A | .50         | TEST EBX,EBX
0040216B | .75 39      | JLE SHORT Ixun4.0040216A
0040216C | .8B 45 14   | MOV EBX, DWORD PTR SS:[EBP+14]
0040216D | .8B 45 14   | MOV EBX, DWORD PTR SS:[EBP+14]
0040216E | .76 02      | JNO SHORT Ixun4.00402144
0040216F | .85 01      | MOV EB, 1
00402170 | .8B 55 0C   | MOV EDI, DWORD PTR SS:[EBP+C]
00402171 | .8B 55 0C   | MOV EDI, DWORD PTR DS:[EDI]
00402172 | .6A 00      | PUSH 0
00402173 | .6A 01      | PUSH 1
00402174 | .8D 45 FF   | LEA EDI, DWORD PTR SS:[EBP-1]
00402175 | .50         | PUSH EBX
00402176 | .FF 75 0C   | PUSH DWORD PTR SS:[EBP+C]
00402177 | .C8 400000  | CALL DWORD PTR DS:[EDI+0]
00402178 | .FF 75 0C   | PUSH DWORD PTR SS:[EBP+C]
00402179 | .E8 BCFEFFFF | CALL Ixun4.00402168
0040217A | .8B 45 18   | CMP EDI, DWORD PTR SS:[EBP+18]
0040217B | .72 05      | JB SHORT Ixun4.0040216E
0040217C | .EB 04      | JMP SHORT Ixun4.0040216A
0040217D | .90         | TEST EBX,EBX
0040217E | .74 BC      | JLE SHORT Ixun4.00402126
0040217F | .80        | MOV EBX,EBX
Stack SS:[011AFDF3]4A (*'J')

```

Debugging

- # Address 0x004025A8 calls a function that hashes previously read bytes
- # If hash value is incorrect, connection closes

```

004025A7 | .50         | PUSH EBX
004025A8 | .E8 BCFEFFFF | CALL Ixun4.004025A8
004025A9 | .8B 05     | MOV EBX, DWORD PTR DS:[EDI+0]
004025AA | .74 05     | JE SHORT Ixun4.004025B5
004025AB | .59 200000 | JMP Ixun4.00402590
004025AC | .FC       | CLD
004025AD | .80BD 37FFFF | LEA EDI, DWORD PTR SS:[EBP-C9]
004025AE | .8B 000000 | MOV EBX, 0
DS:[00406004]=9C0209C4
EAX=CE250B8B

```

Debugging

Beagle responds with 8 byte sequence if hash value is correct

```
00402CC2 | . 01 00604000 | MOV EAX, DWORD PTR DS:[406000]
00402CC7 | . 60 00 | STOS DWORD PTR ES:[EDI]
00402CC8 | . 60 00 | PUSH 0
00402CC9 | . 00B5 87FFFFFFF | LEA EAX, DWORD PTR SS:[EBP-C9]
00402CCD | . 59 | PUSH EAX
00402CE0 | . FF75 08 | PUSH DWORD PTR SS:[EBP+8]
00402CE3 | . E8 20180000 | CALL JMP.&sub_40402CE4
00402CE4 | . 80BD REFEFFF | CMP BYTE PTR SS:[EBP-152],2
00402CE5 | . 74 00 | JE SHORT Jrun4.00402CF1
00404108=JMP.&socket32_send7
```

Address	Hex	Asm	RECI
0120FEEF	03 00 00 00 01 03 00 00 00	...	
0120FEC7	...		

Debugging

9th byte is used as option number
Option numbers are 2, 3, 4, 8, and 10

```
00402E59 | . FF75 08 | PUSH DWORD PTR SS:[EBP+8]
00402E5C | . E8 20180000 | CALL JMP.&sub_40402CE4
00402E5F | . 80BD REFEFFF | CMP BYTE PTR SS:[EBP-152],2
00402E61 | . 74 00 | JE SHORT Jrun4.00402CF1
00402E64 | . 80BD REFEFFF | CMP BYTE PTR SS:[EBP-152],3
00402E66 | . 9F85 8A100000 | JMP JMP.&sub_40402E70
00402E68 | . 6A 00 | PUSH 0
00402E69 | . 6A 04 | PUSH 4
00402E6A | . 6A 04 | PUSH 4
00402E6C | . FF75 0C | PUSH DWORD PTR SS:[EBP+C]
00402E6F | . FF75 08 | PUSH DWORD PTR SS:[EBP+8]
00402E70 | . E8 83FEFFFF | CALL Jrun4.00402085
00402E72 | . 803D | TEST EBX, EBX
00402E73 | . 74 00 | JE SHORT Jrun4.00402E75
```

Reg5	Reg4	Reg3	Reg2	Reg1
00000000	00000004	00000004	00000004	Jrun4.00402085

Exercise

How does Beagle.J behave with option numbers 2, 3, 4, 8, 10?
Why is Beagle.J contacting websites?
What's a/the password for the backdoor?

Summary

- ✦ Analyzing backdoor opened by virus
- ✦ Need to work on both sides: Client and server
- ✦ Locate backdoor
- ✦ Create client to connect to backdoor
- ✦ Use debugger
 - to trap connection
 - trace execution
 - alter execution
