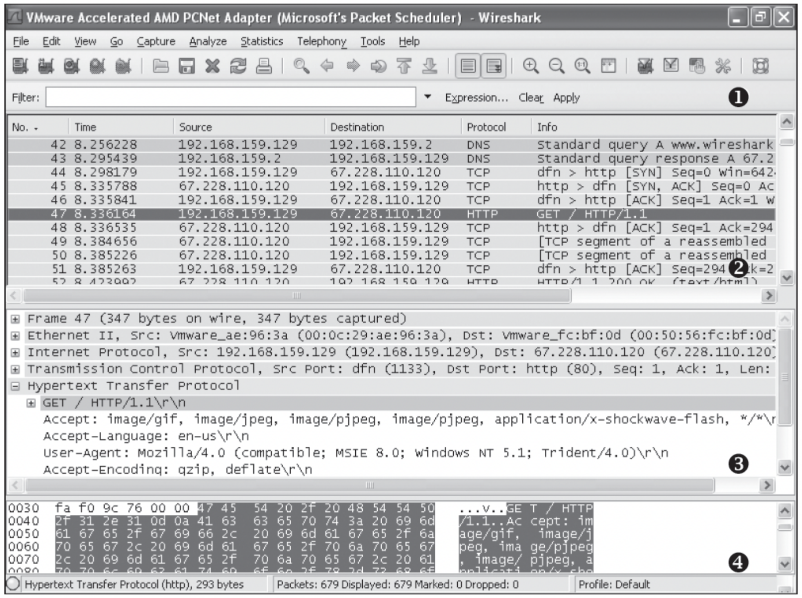
Practical Malware Analysis

The Hands-On Guide to Dissecting Malicious Software

by Michael Sikorski & Andrew Honig

errata updated to print 18

<table>
<tr><th>Page</th><th>Error</th><th>Correction</th><th>Print corrected</th></tr>
<tr><td>10</td><td><code>373e7a863a1a345c60edb9e20ec3231</code></td><td><code>373e7a863a1a345c60edb9e20ec32311</code></td><td>Print 2</td></tr>
<tr><td>54</td><td>Figure replacement</td><td>

VMware Accelerated AMD PCNet Adapter (Microsoft's Packet Scheduler) - Wireshark
File Edit View Go Capture Analyze Statistics Telephony Tools Help
Filter:
Expression... Clear Apply
❶
❷
❸
❹

<pre>
No. -   Time   Source   Destination   Protocol Info
     42 8.256228 192.168.159.129   192.168.159.2   DNS   Standard query A www.wireshark
     43 8.295439 192.168.159.2   192.168.159.129   DNS   Standard query response A 67.2
     44 8.298179 192.168.159.129   67.228.110.120   TCP   dfn > http [SYN] Seq=0 Win=642
     45 8.335788 67.228.110.120   192.168.159.129   TCP   http > dfn [SYN, ACK] Seq=0 Ac
     46 8.335841 192.168.159.129   67.228.110.120   TCP   dfn > http [ACK] Seq=1 Ack=1 W
     47 8.336164 192.168.159.129   67.228.110.120   HTTP   GET / HTTP/1.1
     48 8.336535 67.228.110.120   192.168.159.129   TCP   http > dfn [ACK] Seq=1 Ack=294
     49 8.384656 67.228.110.120   192.168.159.129   TCP   [TCP segment of a reassembled
     50 8.385226 67.228.110.120   192.168.159.129   TCP   [TCP segment of a reassembled
     51 8.385263 192.168.159.129   67.228.110.120   TCP   dfn > http [ACK] Seq=294 Ack=2
     52 8.423992 67.228.110.120   192.168.159.129   HTTP   HTTP/1.1 200 OK (text/html)
</pre>

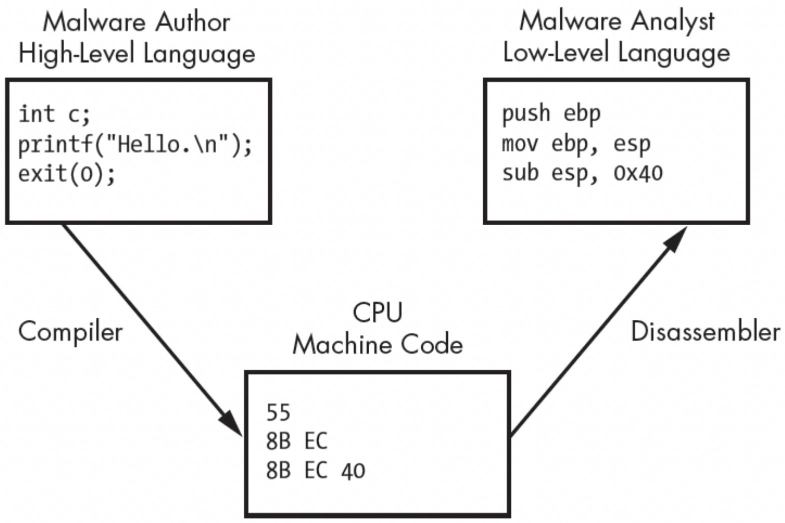
<pre>
⊞ Frame 47 (347 bytes on wire, 347 bytes captured)
⊞ Ethernet II, Src: Vmware_ae:96:3a (00:0c:29:ae:96:3a), Dst: Vmware_fc:bf:0d (00:50:56:fc:bf:0d)
⊞ Internet Protocol, Src: 192.168.159.129 (192.168.159.129), Dst: 67.228.110.120 (67.228.110.120)
⊞ Transmission Control Protocol, Src Port: dfn (1133), Dst Port: http (80), Seq: 1, Ack: 1, Len:
⊟ Hypertext Transfer Protocol
  ⊞ GET / HTTP/1.1\r\n
    Accept: image/gif, image/jpeg, image/pjpeg, image/pjpeg, application/x-shockwave-flash, */*\r
    Accept-Language: en-us\r\n
    User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0)\r\n
    Accept-Encoding: gzip, deflate\r\n
</pre>

<pre>
0030 fa f0 9c 76 00 00 47 45 54 20 2f 20 48 54 54 50   ...v..GE T / HTTP
0040 2f 31 2e 31 0d 0a 41 63 63 65 70 74 3a 20 69 6d   /1.1..Ac cept: im
0050 61 67 65 2f 67 69 66 2c 20 69 6d 61 67 65 2f 6a   age/gif, image/j
0060 70 65 67 2c 20 69 6d 61 67 65 2f 70 6a 70 65 67   peg, ima ge/pjpeg
0070 2c 20 69 6d 61 67 65 2f 70 6a 70 65 67 2c 20 61   , image/ pjpeg, a
0080 70 70 6c 69 63 61 74 69 6f 6e 2f 78 2d 73 68 6f   pplicati on/x-sho
</pre>

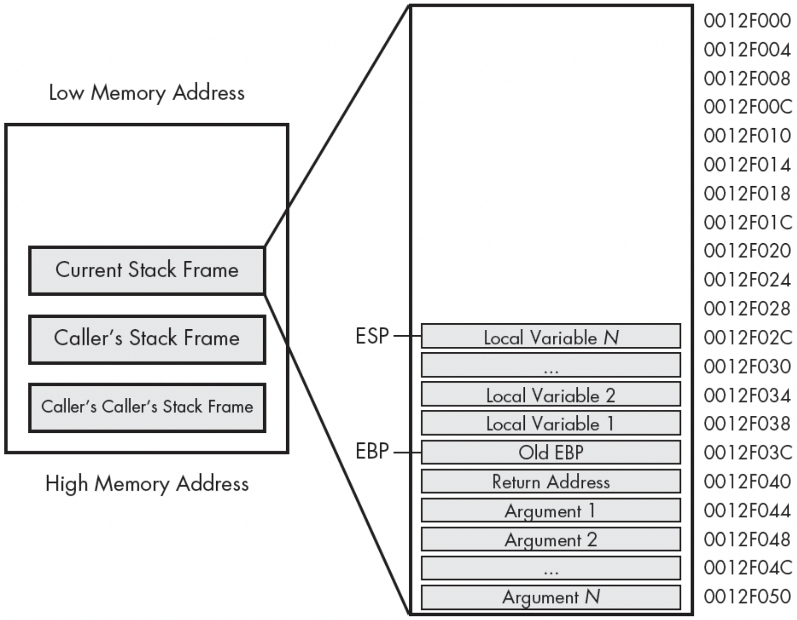
Hypertext Transfer Protocol (http), 293 bytes | Packets: 679 Displayed: 679 Marked: 0 Dropped: 0 | Profile: Default

Figure 3-10: Wireshark DNS and HTTP example
</td><td>Print 6</td></tr>
</table>

Page	Error	Correction	Print corrected

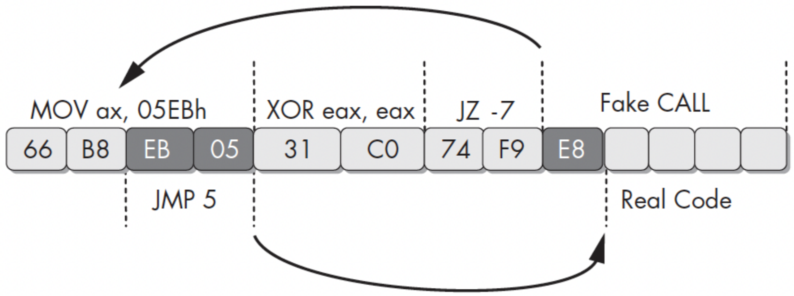
66	Figure replacement	 Malware Author
		High-Level Language
		Malware Analyst
		Low-Level Language
		Compiler
		CPU
		Machine Code
		Disassembler
		`int c;`
		`printf("Hello.\n");`
		`exit(0);`
		`push ebp`
		`mov ebp, esp`
		`sub esp, 0x40`
		`55`
		`8B EC`
		`8B EC 40`
		Figure 4-1: Code level examples
			Print 2

74	. . . instruction such as `lea ebx, [eax*5+5]`, where eax is a number, rather than a memory address. This instruction is the functional equivalent of `ebx = (eax+1)*5`, but the former is shorter or more efficient for the compiler to use instead of a total of four instructions (for example `inc eax`; `mov ecx, 5`; `mul ecx`; `mov ebx, eax`).	. . . instruction such as `lea ebx, [eax*4+4]`, where eax is a number, rather than a memory address. This instruction is the functional equivalent of `ebx = (eax+1)*4`, but the former is shorter or more efficient for the compiler to use instead of a total of four instructions (for example `inc eax`; `mov ecx, 4`; `mul ecx`; `mov ebx, eax`).	Print 14
76	The instruction `nop` is actually a pseudonym for `xhcg eax, eax` . . .	The instruction `nop` is actually a pseudonym for `xchg eax, eax` . . .	Print 7

Page	Error	Correction	Print corrected
79	Figure replacement	 Low Memory Address Current Stack Frame Caller's Stack Frame Caller's Caller's Stack Frame High Memory Address ESP EBP Local Variable *N* ... Local Variable 2 Local Variable 1 Old EBP Return Address Argument 1 Argument 2 ... Argument *N* 0012F000 0012F004 0012F008 0012F00C 0012F010 0012F014 0012F018 0012F01C 0012F020 0012F024 0012F028 0012F02C 0012F030 0012F034 0012F038 0012F03C 0012F040 0012F044 0012F048 0012F04C 0012F050 *Figure 4-8: Individual stack frame*	Print 8
82	This works in the same way as cmpsb, but it compares the byte located at address ESI to AL, rather than to EDI.	This works in the same way as cmpsb, but it compares the byte located at address EDI to AL, rather than to ESI.	Print 8
84	**Table 4-12**	**Listing 4-2**	Print 10
110, 111	`printf("total = %d\n", x);`	`printf("Total = %d\n", x);`	Print 4

111, 112	`00401006 mov dword ptr [ebp-4], 0`
	`0040100D mov dword ptr [ebp-8], 1`
		`00401006 mov dword ptr [ebp-4], 1`
		`0040100D mov dword ptr [ebp-8], 2`
			Print 2

148	The `lpStartupInfo` structure for the process stores the standard output ❶, standard input ❷, and standard error ❸ that will be used for the new process.	The `lpStartupInfo` structure for the process stores the standard output ❷, standard input ❸, and standard error ❶ that will be used for the new process.	Print 2
178	. . . and `0x411001` if the language is Chinese.	. . . and `0x41100A` if the language is Chinese.	Print 7
258	`CreateProcess(...,"svchost.exe",...,CREATE_SUSPEND,...);`	`CreateProcess(...,"svchost.exe",...,CREATE_SUSPENDED,...);`	Print 2

Page	Error	Correction	Print corrected
263	Every thread has a queue of APCs attached to it, and these are processed when the thread is in an alertable state, such as when they call functions like `WaitForSingleObjectEx`, `WaitForMultipleObjectsEx`, and `Sleep`.	Every thread has a queue of APCs attached to it, and these are processed when the thread is in an alertable state, such as when they call functions like `WaitForSingleObjectEx`, `WaitForMultipleObjectsEx`, and `SleepEx`.	Print 2
290	`cbuf = f.read()`	`cbuf = cfile.read()`	Print 5
338	Figure replacement	 MOV ax, 05EBh XOR eax, eax JZ -7 Fake CALL 66 B8 EB 05 31 C0 74 F9 E8 JMP 5 Real Code *Figure 15-5: Multilevel inward-jumping sequence*	Print 2
338	`74 F9                jz     short near ptr sub_4011C0+1`	`74 FA                jz     short near ptr sub_4011C0+2`	Print 2
339	`F9                   db 0F9h`	`FA                   db 0FAh`	Print 2
363	Because `INT 0x2D` is the way that kernel debuggers set breakpoints, the method shown in Listing 16-10 applies.	Because `INT 0x2D` is the way that kernel debuggers set breakpoints, the method shown in Listing 16-9 applies.	Print 2
376	0x5668 (`VX`)	0x5658 (`VX`)	Print 14
440	3. **At 0x4036F0, there is a function call that** takes the string `Config` error, followed a few instructions later by a call to `CxxThrowException`.	3. **The function `0x4036F0` is called multiple times and each time it** takes the string `Config` error, followed a few instructions later by a call to `CxxThrowException`.	Print 6
447	*\WOW64*	*\SysWOW64*	Print 12
448	*C:\Windows\WOW64*	*C:\Windows\SysWOW64*	Print 12
471	URL update	You can download PEview from *http://wjradburn.com/software/*	Print 2
499	**View ▶ Graphs ▶ Xrefs From**	**View ▶ Graphs ▶ User Xrefs Chart**	Print 2
514	If the call **fails**, the program exits.	If the call **succeeds**, the program exits.	Print 2

Page	Error	Correction	Print corrected
523	. . . if so, it calls the `Sleep` function to sleep for **60** seconds.	. . . if so, it calls the `Sleep` function to sleep for **about 394** seconds.	Print 6
533	*If you perform a full analysis of 0x4025**2**0 . . .*	*If you perform a full analysis of 0x4025**1**0 . . .*	Print 7
649	The two functions (`sub_4012`**`2F2`** and `sub_4013`**`69`**) . . .	The two functions (`sub_401`**`30F`** and `sub_4013`**`86`**) . . .	Print 2
675	The malware is querying the I/O communication port (0x56**6**8) . . .	The malware is querying the I/O communication port (0x56**5**8) . . .	Print 14
680	. . . as described in “Searching for Vulnerable Instructions” on page 67**0**.	. . . as described in “Searching for Vulnerable Instructions” on page 67**8**.	Print 6