<table>
<thead>
<tr>
<th>Page</th>
<th>Error</th>
<th>Correction</th>
<th>Print Corrected</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>373e7a863a1a345c60edb9e20ec3231</td>
<td>373e7a863a1a345c60edb9e20ec3231</td>
<td>Print 2</td>
</tr>
<tr>
<td>54</td>
<td>Figure replacement</td>
<td></td>
<td>Print 6</td>
</tr>
</tbody>
</table>

Figure 3-10: Wireshark DNS and HTTP example
The instruction *nop* is actually a pseudonym for *xchg* eax, eax...
This works in the same way as cmpps, but it compares the byte located at address ESI to AL, rather than to EDI.

---

The `lpStartupInfo` structure for the process stores the standard output, standard input, and standard error that will be used for the new process.

---

. . . and `0x411001` if the language is Chinese.

---

CreateProcess(...,svchost.exe,...,CREATE_SUSPEND,...);
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`.

cbuf = f.read()

cbuf = cfile.read()

Figure replacement

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.

74 F9      jz      short near ptr sub_4011C0+1
74 FA      jz      short near ptr sub_4011C0+2

F9       db 0F9h
FA       db 0FAh

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`.

\Wow64
\SysWow64

C:\Windows\Wow64
C:\Windows\SysWow64

URL update

You can download PEview from http://wjradburn.com/software/

View ▶ Graphs ▶ Xrefs ▶ From

View ▶ Graphs ▶ User Xrefs Chart

If the call fails, the program exits.

If the call succeeds, the program exits.
<table>
<thead>
<tr>
<th>Page</th>
<th>Error</th>
<th>Correction</th>
<th>Print corrected</th>
</tr>
</thead>
<tbody>
<tr>
<td>523</td>
<td>... if so, it calls the Sleep function to sleep for <strong>60</strong> seconds.</td>
<td>... if so, it calls the Sleep function to sleep for <strong>about 394</strong> seconds.</td>
<td>Print 6</td>
</tr>
<tr>
<td>533</td>
<td>If you perform a full analysis of 0x402520 . . .</td>
<td>If you perform a full analysis of 0x402510 . . .</td>
<td>Print 7</td>
</tr>
<tr>
<td>649</td>
<td>The two functions (sub_4012F2 and sub_401369) . . .</td>
<td>The two functions (sub_40130F and sub_401386) . . .</td>
<td>Print 2</td>
</tr>
<tr>
<td>675</td>
<td>The malware is querying the I/O communication port (0x5668) . . .</td>
<td>The malware is querying the I/O communication port (0x5658) . . .</td>
<td>Print 14</td>
</tr>
<tr>
<td>680</td>
<td>... as described in “Searching for Vulnerable Instructions” on page 670.</td>
<td>... as described in “Searching for Vulnerable Instructions” on page 678.</td>
<td>Print 6</td>
</tr>
</tbody>
</table>