Chapter 3 Labs

Lab 3-1

Lab 3-1

Analyze the malware found in the file *Lab03-01.exe* using basic dynamic analysis tools.

Questions

1. What are this malware's imports and strings?
2. What are the malware's host-based indicators?
3. Are there any useful network-based signatures for this malware? If so, what are they?

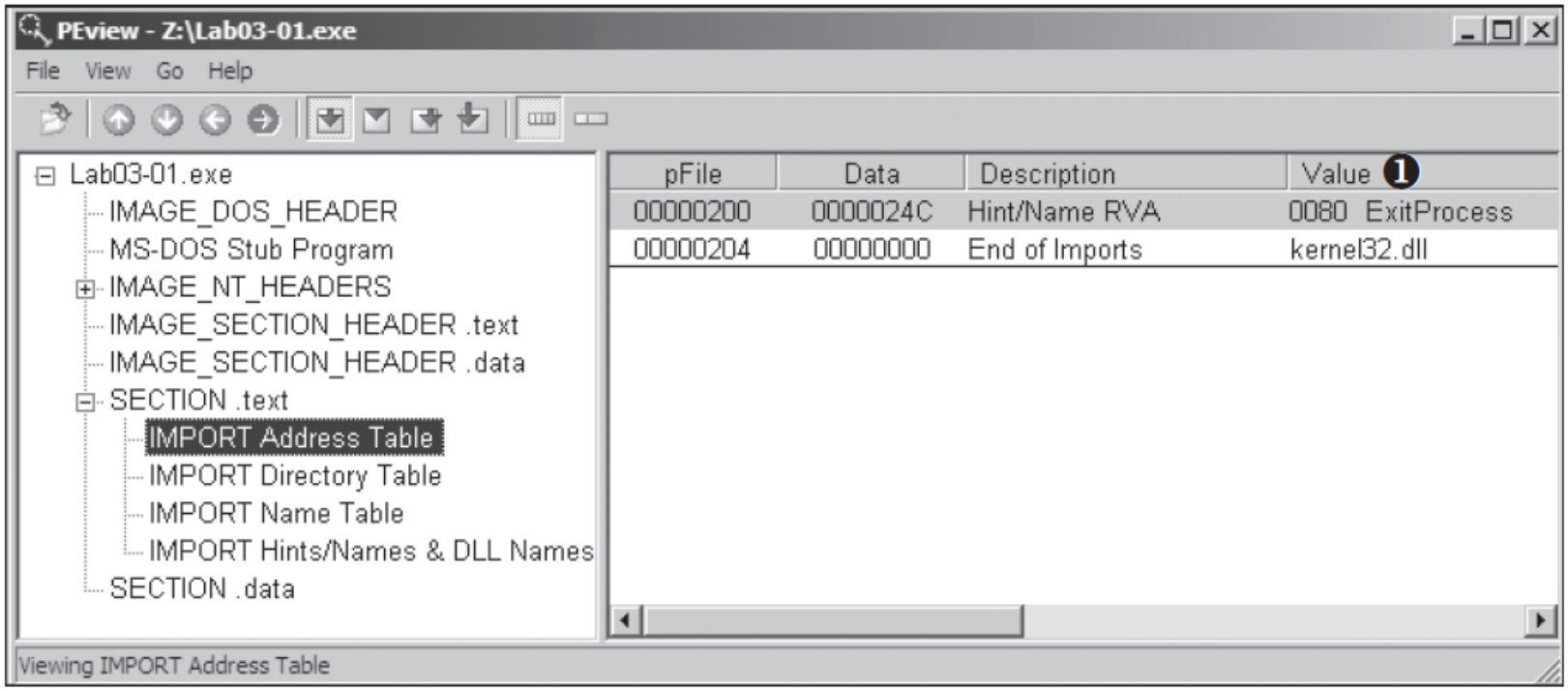
Lab 3-1

Answer 3-1-1

The malware appears to be packed. The only import is `ExitProcess`, although the strings appear to be mostly clear and not obfuscated.

Lab 3-1

We begin with basic static analysis techniques, by looking at the malware's PE file structure and strings. Figure 3-1L shows that only *kernel32.dll* is imported.



There is only one import to this binary, `ExitProcess`, as seen at ❶ in the import address table. Without any imports, it is tough to guess the program's functionality. This program may be packed, since the imports will likely be resolved at runtime.

Lab 3-1

Next, we look at the strings, as shown in the following listing.

```
StubPath
SOFTWARE\Classes\http\shell\open\commandV
Software\Microsoft\Active Setup\Installed Components\
test
www.practicalmalwareanalysis.com
admin
VideoDriver
WinVMX32-
vmx32to64.exe
SOFTWARE\Microsoft\Windows\CurrentVersion\Run
SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders
AppData
```

We wouldn't expect to see strings, since the imports led us to believe that the file is packed, but there are many interesting strings, such as registry locations and a domain name, as well as `WinVMX32`, `VideoDriver`, and `vmx32to64.exe`. Let's see if basic dynamic analysis techniques will show us how these strings are used.

Lab 3-1

Answer 3-1-2

Before we run the malware, we run procmon and clear out all events; start Process Explorer; and set up a virtual network, including ApateDNS, Netcat (listening on ports 80 and 443), and network capturing with Wireshark.

Once we run the malware, we start examining the process in Process Explorer, as shown in Figure 3-2L. We begin by clicking *Lab03-01.exe* in the process listing and select **View ▸ Lower Pane View ▸ Handles**. In this view, we can see that the malware has created the mutex named `WinVMX32` at ❶. We also select **View ▸ Lower Pane View ▸ DLLs** and see that the malware has dynamically loaded DLLs such as *ws2_32.dll* and *wshtcpip.dll*, which means that it has networking functionality.

Lab 3-1

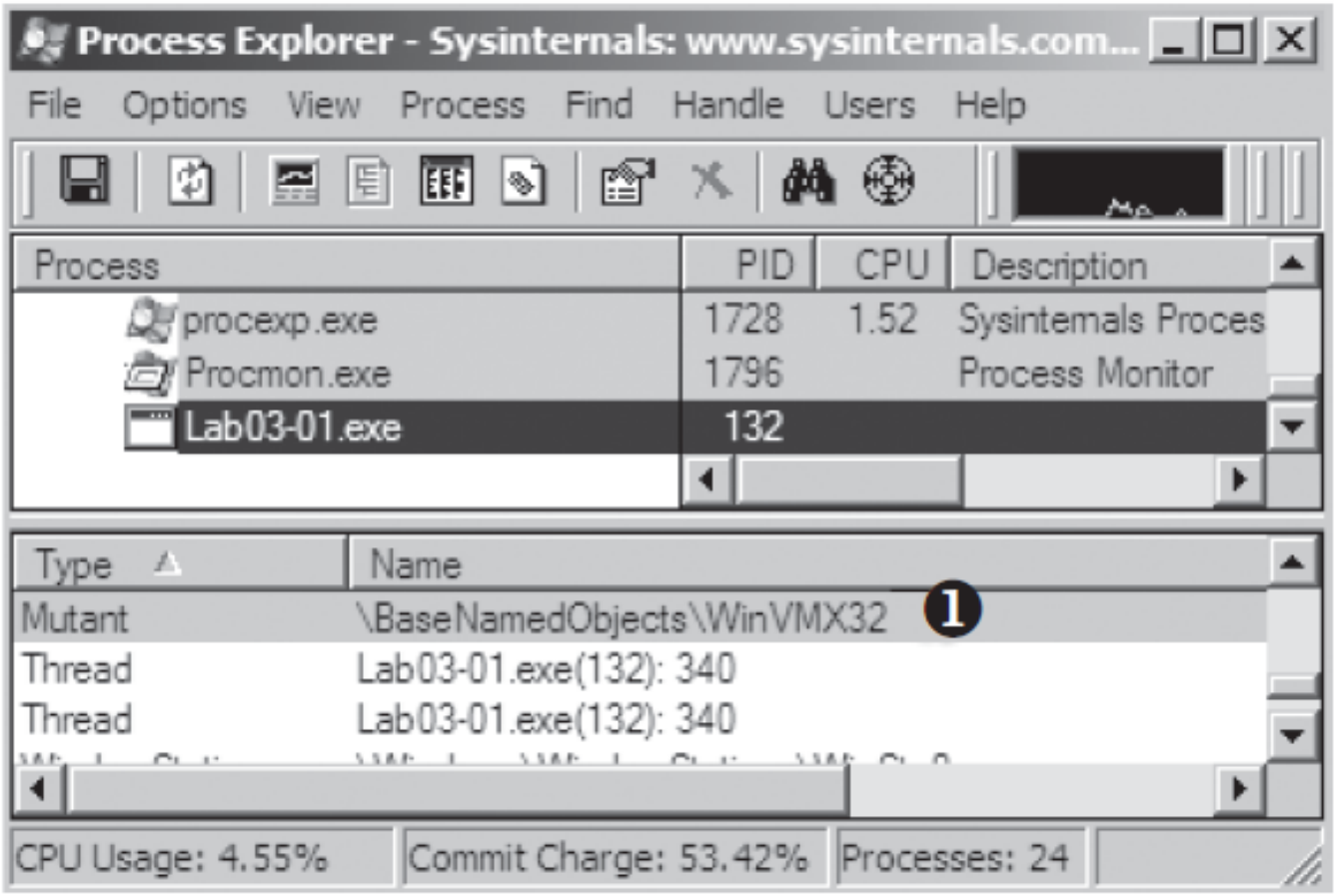


Figure 3-2L: Process Explorer view of Lab03-01.exe *showing the mutex it creates*

Lab 3-1

Next, we use procmon to look for additional information. We bring up the Filter dialog by selecting **Filter ▸ Filter**, and then set three filters: one on the Process Name (to show what *Lab03-01.exe* does to the system), and two more on Operation, as shown in Figure 3-3L. We include `RegSetValue` and `WriteFile` to show changes the malware makes to the filesystem and registry.

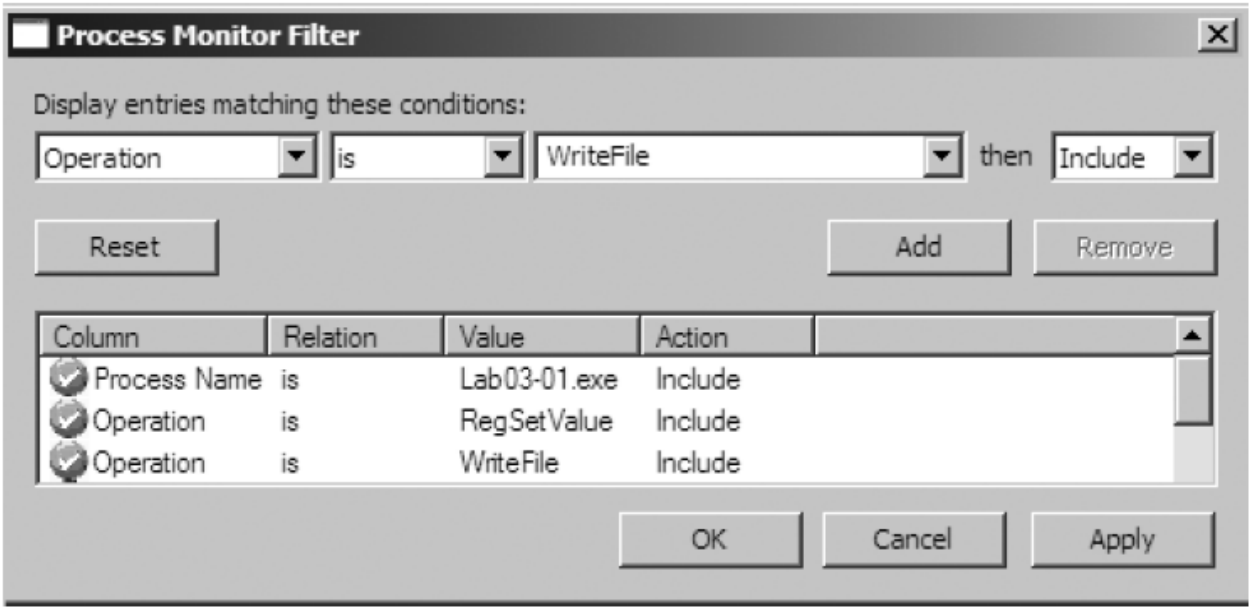


Figure 3-3L: Process Monitor Filter dialog showing filters set on Process Name and Operation

Lab 3-1

Having set our filters, we click **Apply** to see the filtered result. The entries are reduced from thousands to just the 10 seen in Figure 3-4L. Notice that there is only one entry for `WriteFile`, and there are nine entries for `RegSetValue`.

Seq.	Time ...	Process Name	PID	Operation	Path	Result	Detail
0	6:26:4...	Lab03-01.exe	132	RegSetValue	HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed	SUCCESS	Type: REG_BINARY, Length:
1	6:26:4...	Lab03-01.exe	132	WriteFile	C:\WINDOWS\system32\vmx32to64.exe ❶	SUCCESS	Offset: 0, Length: 7,168
2	6:26:4...	Lab03-01.exe	132	RegSetValue	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\VideoDriver ❷	CCESS	Type: REG_SZ, Length: 510,
3	6:26:4...	Lab03-01.exe	132	RegSetValue	HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed	SUCCESS	Type: REG_BINARY, Length:
4	6:26:4...	Lab03-01.exe	132	RegSetValue	HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed	SUCCESS	Type: REG_BINARY, Length:
5	6:26:4...	Lab03-01.exe	132	RegSetValue	HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed	SUCCESS	Type: REG_BINARY, Length:
6	6:26:4...	Lab03-01.exe	132	RegSetValue	HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed	SUCCESS	Type: REG_BINARY, Length:
7	6:26:4...	Lab03-01.exe	132	RegSetValue	HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed	SUCCESS	Type: REG_BINARY, Length:
8	6:26:4...	Lab03-01.exe	132	RegSetValue	HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed	SUCCESS	Type: REG_BINARY, Length:
9	6:26:4...	Lab03-01.exe	132	RegSetValue	HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed	SUCCESS	Type: REG_BINARY, Length:

Figure 3-4L: Procmon filtered results (with three filters set)

As discussed in Chapter 3, we often need to filter out a certain amount of noise, such as entries 0 and 3 through 9 in Figure 3-4L. The `RegSetValue` on `HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed` is typical noise in the results because the random number generator seed is constantly updated in the registry by software.

Lab 3-1

We are left with two interesting entries, as shown in Figure 3-4L at ❶ and ❷. The first is the `WriteFile` operation at ❶. Double-clicking this entry tells us that it wrote 7,168 bytes to *C:\WINDOWS\system32\vmx32to64.exe*, which happens to be the same size as that of the file *Lab03-01.exe*. Opening Windows Explorer and browsing to that location shows that this newly created file has the same MD5 hash as *Lab03-01.exe*, which tells us that the malware has copied itself to that name and location. This can be a useful host-based indicator for the malware because it uses a hard-coded filename.

Lab 3-1

Next, we double-click the entry at ❷ in the figure, and see that the malware wrote the following data to the registry:

```
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\VideoDriver:C:\WINDOWS\system32\vmx32to64.exe
```

This newly created registry entry is used to run *vmx32to64.exe* on system startup using the `HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run` location and creating a key named `VideoDriver`. We can now bring up procmon's Filter dialog, remove the Operation filters, and slowly comb through the entries for any information we may have missed.

Lab 3-1

Answer 3-1-3

Next, we turn our attention to the network analysis tools we set up for basic dynamic analysis. First we check ApateDNS to see if the malware performed any DNS requests. Examining the output, we see a request for *www.practicalmalwareanalysis.com*, which matches the strings listing shown earlier. (To be sure that the malware has a chance to make additional DNS requests, if any, perform the analysis process a couple of times to see if the DNS request changes or use the NXDOMAIN functionality of ApateDNS.)

We complete the network analysis by examining the Netcat results, as shown in the following listing.

```
C:\>nc -l -p 443
\7⌠ëÅ¿A :°I,j!Yûöí?Ç:1ƒh↕0±ⁿ)∝←ɛg%┬└#xp╧0+╚3Ω☺åiE☼?═■p}»╝/
º_∞~]ò£»ú¿¼─F^"Äµ▒├
♦└ªòj╡<û(y!└♫5Z☺!♀va╪┴╗úI┤ßX╤â8╬²ñö'i¢k╢╓(√Q‼%O¶╡9.▐♂Åw♀‼±Wm^┐#ñæ╬°☻/
[⌠│⌡xH╫▲É╢‼
x?╦Æº│ºLƒ↕x┌gYΦ< └§☻µºx)╤SBxè↕◄╟♂4AÇ
```

Lab 3-1

It looks like we got lucky: The malware appears to beacon out over port 443, and we were listening with Netcat over ports 80 and 443. (Use INetSim to listen on all ports at once.) We run this test several times, and the data appears to be random each time.

A follow-up in Wireshark tells us that the beacon packets are of consistent size (256 bytes) and appear to contain random data not related to the SSL protocol that normally operates over port 443.

Lab 3-2

Lab 3-2

Analyze the malware found in the file *Lab03-02.dll* using basic dynamic analysis tools.

Questions

1. How can you get this malware to install itself?
2. How would you get this malware to run after installation?
3. How can you find the process under which this malware is running?
4. Which filters could you set in order to use procmon to glean information?
5. What are the malware's host-based indicators?
6. Are there any useful network-based signatures for this malware?

Lab 3-2

Answer 3-2-1

We begin with basic static analysis by looking at the PE file structure and strings. Figure 3-5L shows that this DLL has five exports, as listed from ❶ and below. The export `ServiceMain` suggests that this malware needs to be installed as a service in order to run properly.

Lab 3-2

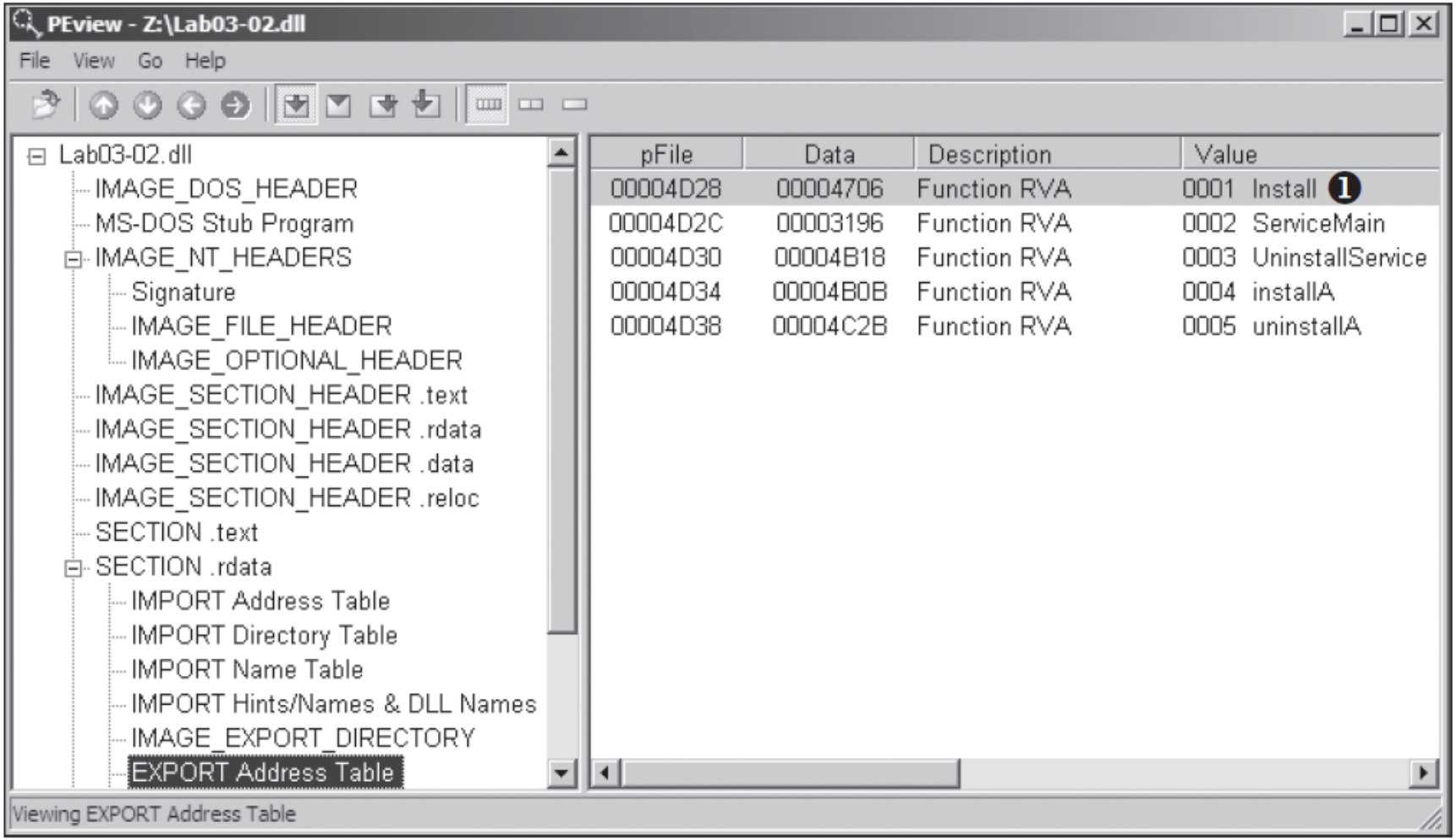


Figure 3-5L: PEview of Lab03-02.dll *exports*

Lab 3-2

The following listing shows the malware's interesting imported functions in bold.

```
OpenService
DeleteService
OpenSCManager
CreateService
RegOpenKeyEx
RegQueryValueEx
RegCreateKey
RegSetValueEx
InternetOpen
InternetConnect
HttpOpenRequest
HttpSendRequest
InternetReadFile
```

These include service-manipulation functions, such as `CreateService`, and registry-manipulation functions, such as `RegSetValueEx`. Imported networking functions, such as `HttpSendRequest`, suggest that the malware uses HTTP.

Lab 3-2

Next, we examine the strings, as shown in the following listing.

```
Y29ubmVjdA==
practicalmalwareanalysis.com
serve.html
dW5zdXBwb3JO
c2xlZXA=
Y21k
cXVpdA==
Windows XP 6.11
HTTP/1.1
quit
exit
getfile
cmd.exe /c
Depends INA+, Collects and stores network configuration and location
information, and notifies applications when this information changes.
%SystemRoot%\System32\svchost.exe -k
SYSTEM\CurrentControlSet\Services\
Intranet Network Awareness (INA+)
%SystemRoot%\System32\svchost.exe -k netsvcs
netsvcs
SOFTWARE\Microsoft\Windows NT\CurrentVersion\Svchost
IPRIP
```

We see several interesting strings, including registry locations, a domain name, unique strings like `IPRIP` and `serve.html`, and a variety of encoded strings. Basic dynamic techniques may show us how these strings and imports are used.

Lab 3-2

Answer 3-2-2

The results of our basic static analysis techniques lead us to believe that this malware needs to be installed as a service using the exported function `installA`. We'll use that function to attempt to install this malware, but before we do that, we'll launch Regshot to take a baseline snapshot of the registry and use Process Explorer to monitor the processes running on the system. After setting up Regshot and Process Explorer, we install the malware using *rundll32.exe*, as follows:

```
C:\>rundll32.exe Lab03-02.dll,installA
```

After installing the malware, we use Process Explorer to confirm that it has terminated by making sure that *rundll32.exe* is no longer in the process listing. Next, we take a second snapshot with Regshot to see if the malware installed itself in the registry.

Lab 3-2

The edited Regshot results are shown in the following listing.

```
----------------------------------
Keys added
----------------------------------
HKLM\SYSTEM\CurrentControlSet\Services\IPRIP ❶
----------------------------------
Values added
----------------------------------
HKLM\SYSTEM\CurrentControlSet\Services\IPRIP\Parameters\ServiceDll:
     "z:\Lab03-02.dll"
HKLM\SYSTEM\CurrentControlSet\Services\IPRIP\ImagePath:
     "%SystemRoot%\System32\svchost.exe -k netsvcs" ❷
HKLM\SYSTEM\CurrentControlSet\Services\IPRIP\DisplayName:
     "Intranet Network Awareness (INA+)" ❸
HKLM\SYSTEM\CurrentControlSet\Services\IPRIP\Description:
     "Depends INA+, Collects and stores network configuration and location
information, and notifies applications when this information changes." ❹
```

Lab 3-2

The `Keys added` section shows that the malware installed itself as the service `IPRIP` at ❶. Since the malware is a DLL, it depends on an executable to launch it. In fact, we see at ❷ that the `ImagePath` is set to `svchost.exe`, which means that the malware will be launched inside an *svchost.exe* process. The rest of the information, such as the `DisplayName` and `Description` at ❸ and ❹, creates a unique fingerprint that can be used to identify the malicious service.

If we examine the strings closely, we see `SOFTWARE\Microsoft\Windows NT\CurrentVersion\SvcHost` and a message `"You specify service name not in Svchost//netsvcs, must be one of following"`. If we follow our hunch and examine the `\SvcHost\netsvcs` registry key, we can see other potential service names we might use, like `6to4 AppMgmt`. Running `Lab03-02.dll,installA 6to4` will install this malware under the `6to4` service instead of the `IPRIP` service, as in the previous listing.

Lab 3-2

After installing the malware as a service, we could launch it, but first we'll set up the rest of our basic dynamic tools. We run procmon (after clearing out all events); start Process Explorer; and set up a virtual network, including ApateDNS and Netcat listening on port 80 (since we see HTTP in the strings listing).

Since this malware is installed as the `IPRIP` service, we can start it using the `net` command in Windows, as follows:

```
c:\>net start IPRIP
The Intranet Network Awareness (INA+) service is starting.
The Intranet Network Awareness (INA+) service was started successfully.
```

The fact that the display name (`INA+`) matches the information found in the registry tells us that our malicious service has started.

Lab 3-2

Answer 3-2-3

Next, we open Process Explorer and attempt to find the process in which the malware is running by selecting **Find ▸ Find Handle or DLL** to open the dialog shown in Figure 3-6L. We enter `Lab03-02.dll` and click **Search**. As shown in the figure, the result tells us that *Lab03-02.dll* is loaded by *svchost.exe* with the PID 1024. (The specific PID may differ on your system.)

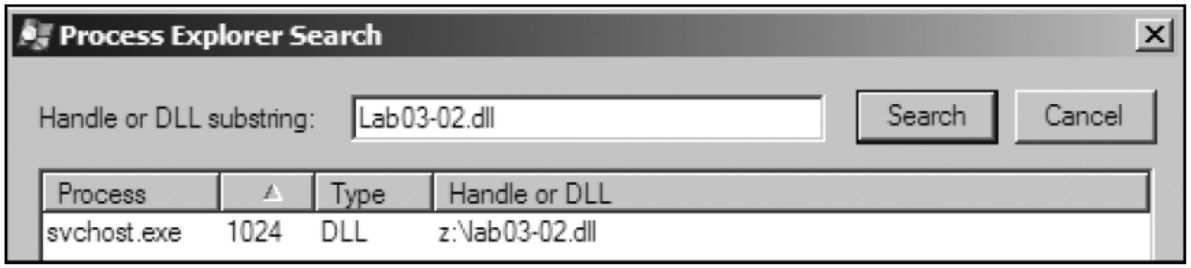


Figure 3-6L: Searching for a DLL in Process Explorer

In Process Explorer, we select **View ▸ Lower Pane View ▸ DLLs** and choose the *svchost.exe* running with PID 1024. Figure 3-7L shows the result. The display name `Intranet Network Awareness (INA+)` shown at ❶ confirms that the malware is running in *svchost.exe*, which is further confirmed when we see at ❷ that *Lab03-02.dll* is loaded.

Lab 3-2

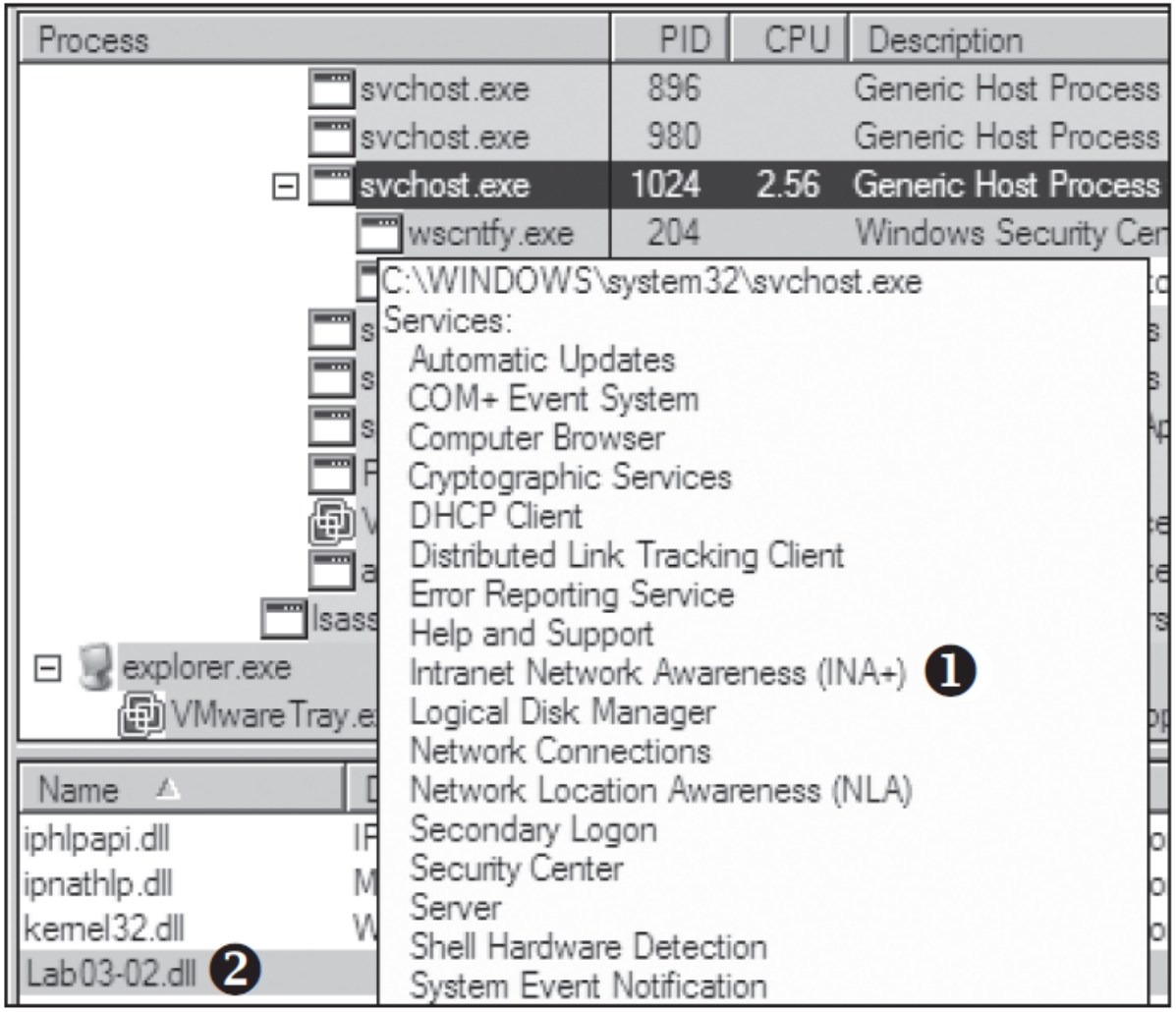


Figure 3-7L: Examining service malware in Process Explorer

Lab 3-2

Answer 3-2-3 and Answer 3-2-4

In procmon you can filter on the PID you found using Process Explorer.

Answer 3-2-6

Next, we turn our attention to our network analysis tools. First, we check ApateDNS to see if the malware performed any DNS requests. The output shows a request for *practicalmalwareanalysis.com*, which matches the strings listing shown earlier.

NOTE *It takes 60 seconds after starting the service to see any network traffic (the program does a* `Sleep(60000)` *before attempting network access). If the networking connection fails for any reason (for example, you forgot to set up ApateDNS), it waits 10 minutes before attempting to connect again.*

Lab 3-2

We complete our network analysis by examining the Netcat results, as follows:

```
c:\>nc -l -p 80
GET /serve.html HTTP/1.1
Accept: */*
User-Agent: MalwareAnalysis2 Windows XP 6.11
Host: practicalmalwareanalysis.com
```

We see that the malware performs an HTTP `GET` request over port 80 (we were listening over port 80 with Netcat since we saw HTTP in the string listing). We run this test several times, and the data appears to be consistent across runs.

Lab 3-3

Execute the malware found in the file *Lab03-03.exe* while monitoring it using basic dynamic analysis tools in a safe environment.

Questions

1. What do you notice when monitoring this malware with Process Explorer?
2. Can you identify any live memory modifications?
3. What are the malware's host-based indicators?
4. What is the purpose of this program?

Lab 3-3

Answer 3-3-1

For this lab, we begin by launching Process Explorer and procmon. When procmon starts, the events stream by quickly, so we use **File ▸ Capture Events** to toggle event capture on and off. (It's best to keep event capture off until all dynamic analysis programs are started and you're ready to execute the program.) We use **Filter ▸ Filter** to open the Filter dialog, and then ensure that only the default filters are enabled by clicking the **Reset** button.

Lab 3-3

Lab03-03.exe can be run from the command prompt or by double-clicking its icon. Once run, *Lab03-03.exe* should be visible inside Process Explorer. Notice how it creates the subprocess *svchost.exe*, and then exits, but leaves the *svchost.exe* process running as an orphaned process, as shown in Figure 3-8L. (An *orphaned process* has no parent process listed in the process tree structure.) The fact that *svchost.exe* is orphaned is highly unusual and highly suspicious.

Process	PID	CPU	Private Bytes	Working Set	Description	Company Name
System Idle Process	0	100.00	0 K	28 K		
explorer.exe	1528		17,672 K	14,808 K	Windows Explorer	Microsoft Corporation
svchost.exe	388		868 K	2,208 K	Generic Host Process for Wi...	Microsoft Corporation

Figure 3-8L: Process Explorer view of orphaned svchost.exe

We investigate further by right-clicking and selecting **Properties** for the orphaned *svchost.exe* process. As shown in Figure 3-8L, the process appears to be a valid *svchost.exe* process with PID 388, but this *svchost.exe* is suspicious because *svchost.exe* is typically a child of *services.exe*.

Lab 3-3

Answer 3-3-2

From this same properties page, we select **Strings** to show the strings in both the executable image on disk and in memory. Toggling between the **Image** and **Memory** radio buttons shows significant discrepancies between the images. As shown in Figure 3-9L, the strings in memory on the right contain `practicalmalwareanalysis.log` and `[ENTER]`, seen at ❶ and ❷, neither of which is found in a typical Windows *svchost.exe* file on disk, as seen on the left.

Lab 3-3

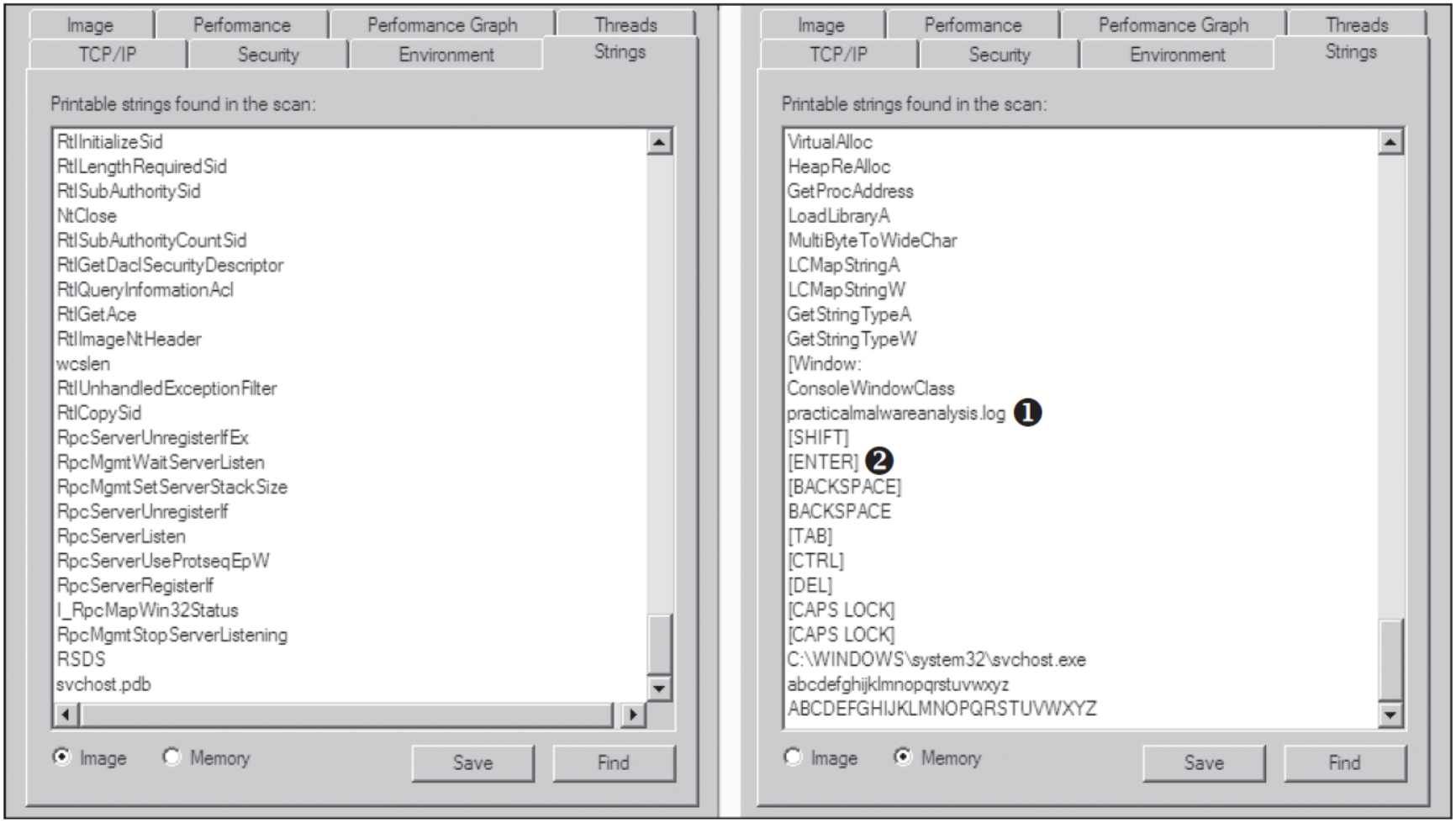


Figure 3-9L: Process Explorer shows strings that are not normally contained in svchost.exe.

Lab 3-3

Answer 3-3-3 and Answer 3-3-4

The presence of the string `practicalmalwareanalysis.log`, coupled with strings like `[ENTER]` and `[CAPS LOCK]`, suggests that this program is a keylogger. To test our assumption, we open Notepad and type a short message to see if the malware will perform keylogging. To do so, we use the PID (found in Process Explorer) for the orphaned *svchost.exe* to create a filter in procmon to show only events from that PID (388). As you can see in Figure 3-10L, the `CreateFile` and `WriteFile` events for *svchost.exe* are writing to the file named *practicalmalwareanalysis.log*. (This same string is visible in the memory view of the orphaned *svchost.exe* process.)

Lab 3-3

Process Name	PID	Operation	Path
svchost.exe	388	CreateFile	C:\WINDOWS\practicalmalwareanalysis.log
svchost.exe	388	QueryStandardInformationFile	C:\WINDOWS\practicalmalwareanalysis.log
svchost.exe	388	WriteFile	C:\WINDOWS\practicalmalwareanalysis.log
svchost.exe	388	WriteFile	C:\WINDOWS\practicalmalwareanalysis.log
svchost.exe	388	WriteFile	C:\WINDOWS\practicalmalwareanalysis.log
svchost.exe	388	WriteFile	C:\WINDOWS\practicalmalwareanalysis.log
svchost.exe	388	WriteFile	C:\WINDOWS\practicalmalwareanalysis.log
svchost.exe	388	CloseFile	C:\WINDOWS\practicalmalwareanalysis.log
svchost.exe	388	CreateFile	C:\WINDOWS\practicalmalwareanalysis.log
svchost.exe	388	QueryStandardInformationFile	C:\WINDOWS\practicalmalwareanalysis.log
svchost.exe	388	WriteFile	C:\WINDOWS\practicalmalwareanalysis.log
svchost.exe	388	CloseFile	C:\WINDOWS\practicalmalwareanalysis.log
svchost.exe	388	CreateFile	C:\WINDOWS\practicalmalwareanalysis.log

Figure 3-10L: Procmon output of svchost.exe with PID 388

Opening *practicalmalwareanalysis.log* with a simple text editor reveals the keystrokes you entered in Notepad. We conclude that this malware is a key-logger that uses process replacement on *svchost.exe*.

Lab 3-4

Analyze the malware found in the file *Lab03-04.exe* using basic dynamic analysis tools. (This program is analyzed further in the Chapter 9 labs.)

Questions

1. What happens when you run this file?
2. What is causing the roadblock in dynamic analysis?
3. Are there other ways to run this program?

Lab 3-4

Answer 3-4-1 and Answer 3-4-2

We begin with basic static analysis, examining the PE file structure and strings. We see that this malware imports networking functionality, service-manipulation functions, and registry-manipulation functions. In the following listing, we notice a number of interesting strings.

Lab 3-4

```
SOFTWARE\Microsoft \XPS
\kernel32.dll
 HTTP/1.0
GET
NOTHING
DOWNLOAD
UPLOAD
SLEEP
cmd.exe
 >> NUL
/c del
http://www.practicalmalwareanalysis.com
NT AUTHORITY\LocalService
 Manager Service
.exe
%SYSTEMROOT%\system32\
k:%s h:%s p:%s per:%s
-cc
-re
-in
```

Lab 3-4

We see strings such as a domain name and the registry location `SOFTWARE\Microsoft \XPS`. Strings like `DOWNLOAD` and `UPLOAD`, combined with the `HTTP/1.0` string, suggest that this malware is an HTTP backdoor. The strings `-cc`, `-re`, and `-in` could be command-line parameters (for example `-in` may stand for install). Let's see if basic dynamic techniques show us how these strings are used.

Before we run the malware, we run procmon and clear out all events, start Process Explorer, and set up a virtual network. When we run the malware, it appears to immediately delete itself, and we see nothing else of interest while watching with Process Explorer.

Next, we use procmon with a filter on the process name *Lab03-04.exe*. There aren't any interesting `WriteFile` or `RegSetValue` entries, but upon further digging, we find an entry for `Process Create`. Double-clicking this entry brings up the dialog shown in Figure 3-11L, and we see that the malware is deleting itself from the system using `"C:\WINDOWS\system32\cmd.exe" /c del Z:\Lab03-04.exe >> NUL`, as seen at ❶.

Lab 3-4

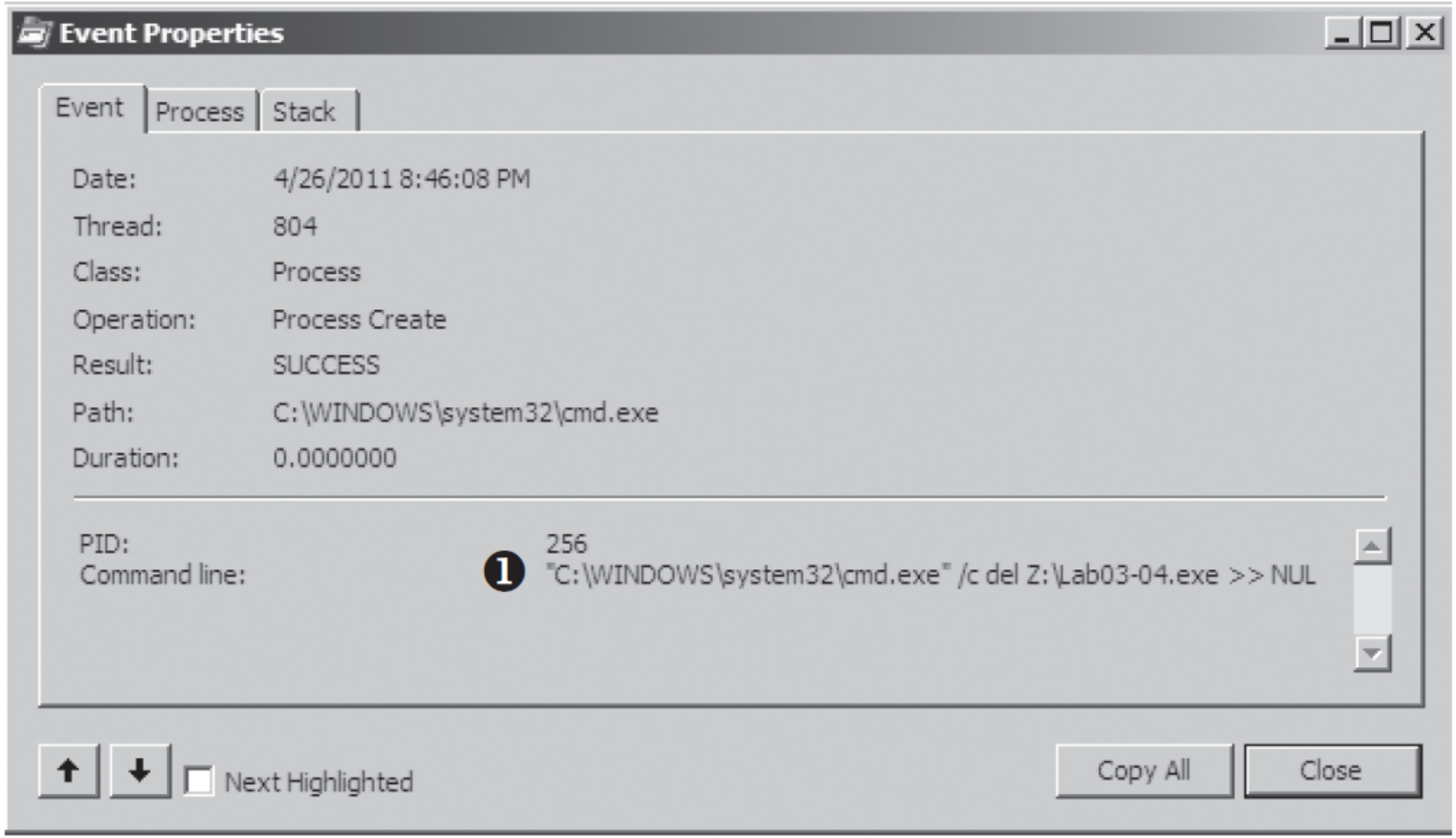


Figure 3-11L: Procmon view of the Process Create performed for self-deletion

Lab 3-4

Answer 3-4-3

We can try to run the malware from the command line using the command-line options we saw in the strings listing (`-in`, `-re`, and `-cc`), but all of them fail and result in the program deleting itself. There isn't much more we can do with basic dynamic techniques at this point, until we dig deeper into the malware. (We will revisit this malware in the Chapter 9 labs.)