

LAB 12 *Dynamic Analysis Tools*

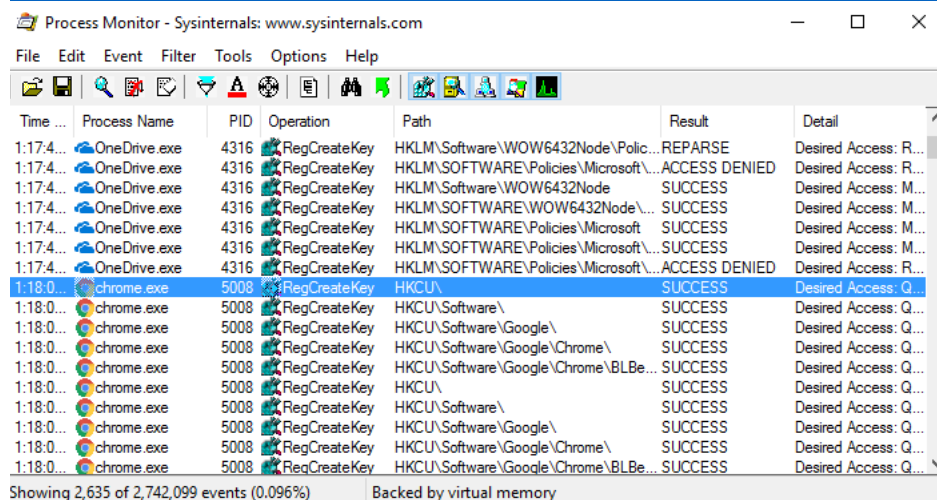
- *Process Monitor*
- *Regshot*
- *HandleDiff*

1. **Process Monitoris:** a free tool from Microsoft that displays file system, registry, process, and other activities on the system.
 - It's an invaluable tool for troubleshooting Windows problems as well as for malware forensics and analysis tasks.
 - The thoroughness of the tool is also weakness, as the amount of data captured by Process Monitor can easily overwhelm the analyst.

(We have already used this tool in the previous section, so we will not introduce it again)

Install:

- ProcessMonitor on Windows: Download on <https://docs.microsoft.com/en-us/sysinternals/downloads/procmon>

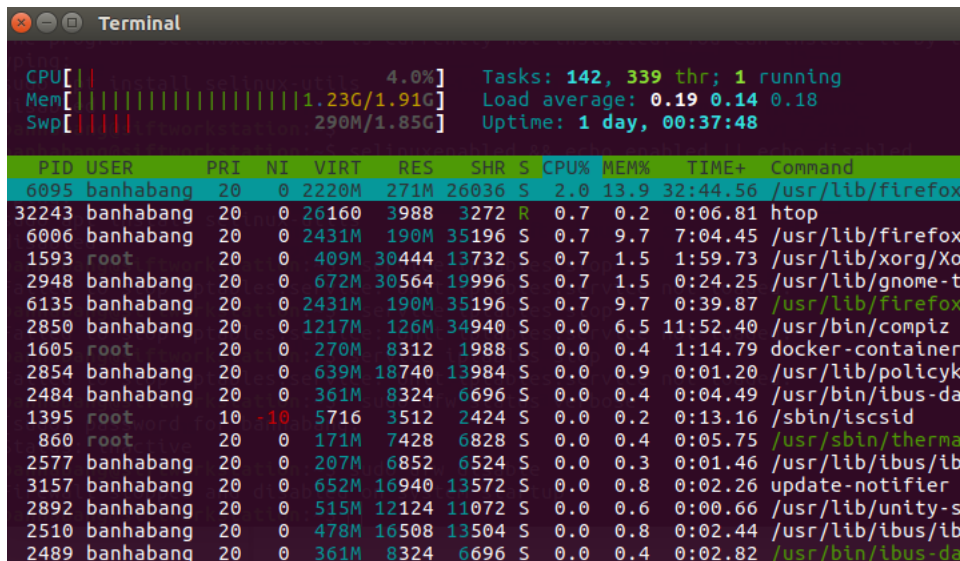


The screenshot shows the Process Monitor application window with the following data:

Time	Process Name	PID	Operation	Path	Result	Detail
1:17:4...	OneDrive.exe	4316	RegCreateKey	HKLM\Software\WOW6432Node\Polic...	REPARSE	Desired Access: R...
1:17:4...	OneDrive.exe	4316	RegCreateKey	HKLM\SOFTWARE\Policies\Microsoft\...	ACCESS DENIED	Desired Access: R...
1:17:4...	OneDrive.exe	4316	RegCreateKey	HKLM\Software\WOW6432Node	SUCCESS	Desired Access: M...
1:17:4...	OneDrive.exe	4316	RegCreateKey	HKLM\SOFTWARE\WOW6432Node\...	SUCCESS	Desired Access: M...
1:17:4...	OneDrive.exe	4316	RegCreateKey	HKLM\Software\Policies\Microsoft	SUCCESS	Desired Access: M...
1:17:4...	OneDrive.exe	4316	RegCreateKey	HKLM\SOFTWARE\Policies\Microsoft\...	SUCCESS	Desired Access: M...
1:17:4...	OneDrive.exe	4316	RegCreateKey	HKLM\SOFTWARE\Policies\Microsoft\...	ACCESS DENIED	Desired Access: R...
1:18:0...	chrome.exe	5008	RegCreateKey	HKCU\	SUCCESS	Desired Access: Q...
1:18:0...	chrome.exe	5008	RegCreateKey	HKCU\Software\	SUCCESS	Desired Access: Q...
1:18:0...	chrome.exe	5008	RegCreateKey	HKCU\Software\Google\	SUCCESS	Desired Access: Q...
1:18:0...	chrome.exe	5008	RegCreateKey	HKCU\Software\Google\Chrome\	SUCCESS	Desired Access: Q...
1:18:0...	chrome.exe	5008	RegCreateKey	HKCU\Software\Google\Chrome\VLBe...	SUCCESS	Desired Access: Q...
1:18:0...	chrome.exe	5008	RegCreateKey	HKCU\	SUCCESS	Desired Access: Q...
1:18:0...	chrome.exe	5008	RegCreateKey	HKCU\Software\	SUCCESS	Desired Access: Q...
1:18:0...	chrome.exe	5008	RegCreateKey	HKCU\Software\Google\	SUCCESS	Desired Access: Q...
1:18:0...	chrome.exe	5008	RegCreateKey	HKCU\Software\Google\Chrome\	SUCCESS	Desired Access: Q...
1:18:0...	chrome.exe	5008	RegCreateKey	HKCU\Software\Google\Chrome\VLBe...	SUCCESS	Desired Access: Q...

Showing 2,635 of 2,742,099 events (0.096%) Backed by virtual memory

- Htopon Ubuntu: `sudo apt-get install htop`



Terminal window showing system status and process list. The status bar at the top indicates CPU usage at 4.0%, memory usage at 1.23G/1.91G, swap usage at 290M/1.85G, 142 tasks, 339 threads, 1 running, load average of 0.19 0.14 0.18, and uptime of 1 day, 00:37:48.

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
6095	banhabang	20	0	2220M	271M	26036	S	2.0	13.9	32:44.56	/usr/lib/firefox
32243	banhabang	20	0	26160	3988	3272	R	0.7	0.2	0:06.81	htop
6006	banhabang	20	0	2431M	190M	35196	S	0.7	9.7	7:04.45	/usr/lib/firefox
1593	root	20	0	409M	30444	13732	S	0.7	1.5	1:59.73	/usr/lib/xorg/Xo
2948	banhabang	20	0	672M	30564	19996	S	0.7	1.5	0:24.25	/usr/lib/gnome-t
6135	banhabang	20	0	2431M	190M	35196	S	0.7	9.7	0:39.87	/usr/lib/firefox
2850	banhabang	20	0	1217M	126M	34940	S	0.0	6.5	11:52.40	/usr/bin/compiz
1605	root	20	0	270M	8312	1988	S	0.0	0.4	1:14.79	docker-container
2854	banhabang	20	0	639M	18740	13984	S	0.0	0.9	0:01.20	/usr/lib/policyk
2484	banhabang	20	0	361M	8324	6696	S	0.0	0.4	0:04.49	/usr/bin/ibus-da
1395	root	10	-10	5716	3512	2424	S	0.0	0.2	0:13.16	/sbin/iscsid
860	root	20	0	171M	7428	6828	S	0.0	0.4	0:05.75	/usr/sbin/therma
2577	banhabang	20	0	207M	6852	6524	S	0.0	0.3	0:01.46	/usr/lib/ibus/ib
3157	banhabang	20	0	652M	16940	13572	S	0.0	0.8	0:02.26	update-notifier
2892	banhabang	20	0	515M	12124	11072	S	0.0	0.6	0:00.66	/usr/lib/unity-s
2510	banhabang	20	0	478M	16508	13504	S	0.0	0.8	0:02.44	/usr/lib/ibus/ib
2489	banhabang	20	0	361M	8324	6696	S	0.0	0.4	0:02.82	/usr/bin/ibus-da

Process Monitor for Malware Analysis:

- Execute malware or malicious code.
- Using Raymond's filters on <https://zeltser.com/process-monitor-filters-for-malware-analysis/>
- It offers a convenient way to examine Process Monitor's log file for activities that are sometimes associated with malware, such as changing the file's attribute, deleting a file, creating a registry key, etc.

2. RegShot

RegShot takes a "snapshot" of your computer allowing you to compare any changes made.

- Registry changes: The malware changes the NoFolderOptionssetting in the registry, which prevents users from being able to control how Windows Explorer displays folders.

It also changes the DisableRegistryToolssetting, which prevents users from starting the default registry editor(s) that Windows provides.

- Files added: The malware adds a file named 944983008.exe and csrssc.exe to the user's temporary directory. Windows OS created the Prefetchdirectory in order to store them.

Two files named 944983008.exe and csrssc.exe executed on the system during the malware's execution.

➔The Prefetchfiles are good sources of forensic evidence

- Files deleted: The malware deleted a file named 944983008.exe from the user's Desktop.

➔ This file is the original malware sample. Thus, you can conclude that the malware deletes itself after executing

The malware does not directly modify any files.

➔ They create two files 944983008.exe or csrss.exe that use the WinINetAPI, in order to update the index.dat.

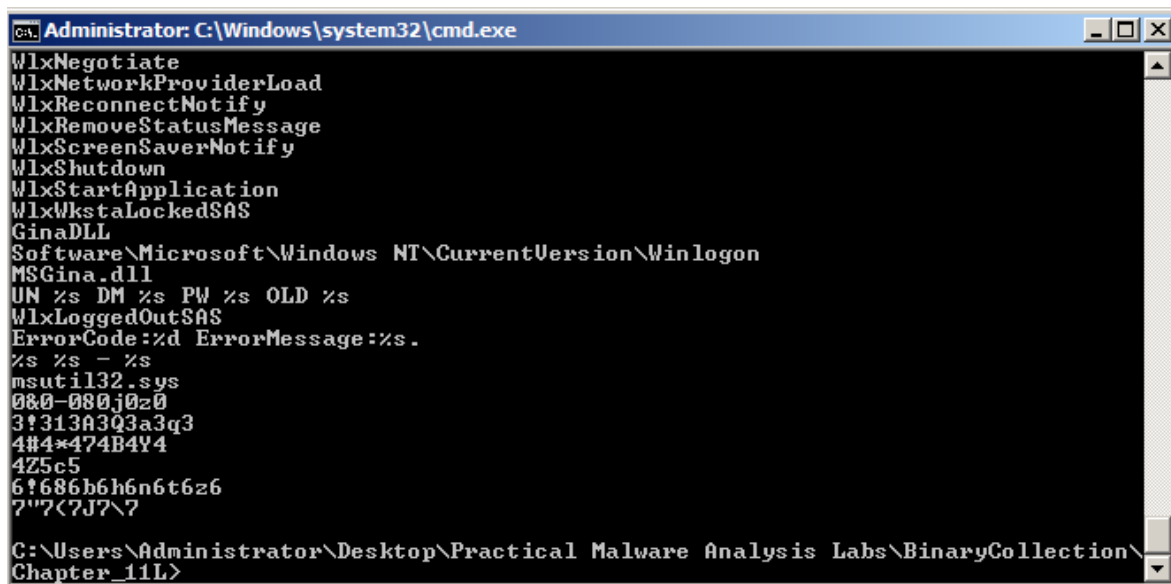
LAB 1:

What you need: The Windows 2008 Server virtual machine we have been using.

Purpose: Analyze malware behavior

Static Analysis with Strings

Examine the strings in Lab11-01.exe. You should find the two items below.

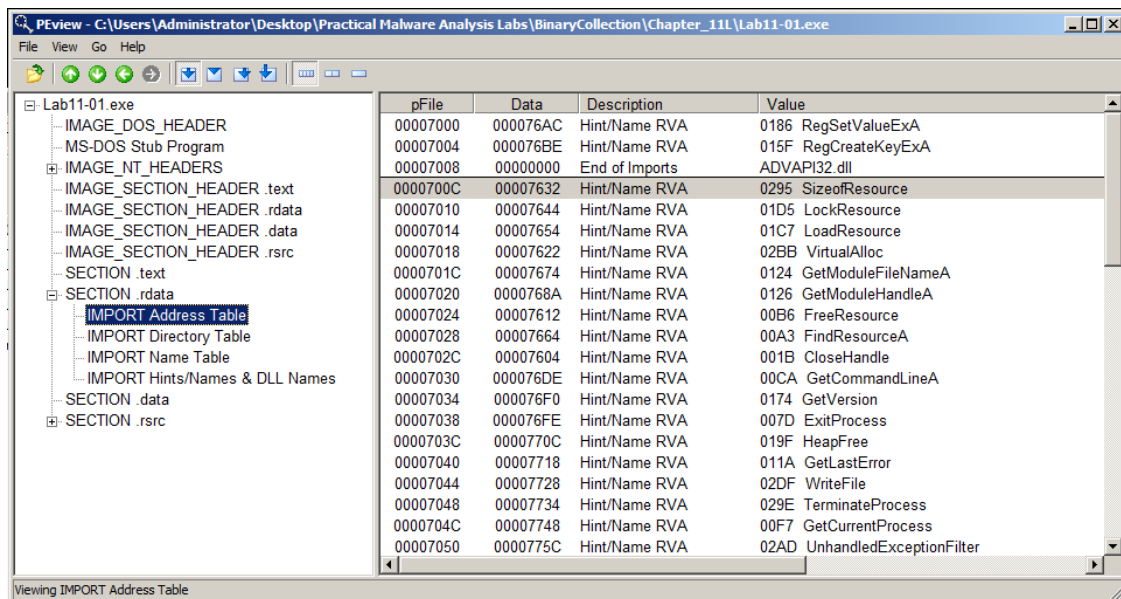


```
Administrator: C:\Windows\system32\cmd.exe
WlxNegotiate
WlxNetworkProviderLoad
WlxReconnectNotify
WlxRemoveStatusMessage
WlxScreenSaverNotify
WlxShutdown
WlxStartApplication
WlxWkstaLockedSAS
GinaDLL
Software\Microsoft\Windows NT\CurrentVersion\Winlogon
MSGina.dll
UN %s DM %s PW %s OLD %s
WlxLoggedOutSAS
ErrorCode:%d ErrorMessage:%s.
%s %s - %s
msutil32.sys
0&0-0&0j0z0
3!313A3Q3a3q3
4#4*474B4Y4
4Z5c5
6!686b6h6n6t6z6
7"7<7J7\7
C:\Users\Administrator\Desktop\Practical Malware Analysis Labs\BinaryCollection\
Chapter_11L>
```

Static Analysis with PEXview

Examine the Lab11-01.exe file in PEXview. Find the items below.

- RegSetValueExA
- RegCreateKeyExA
- SizeofResource
- LockResource
- LoadResource



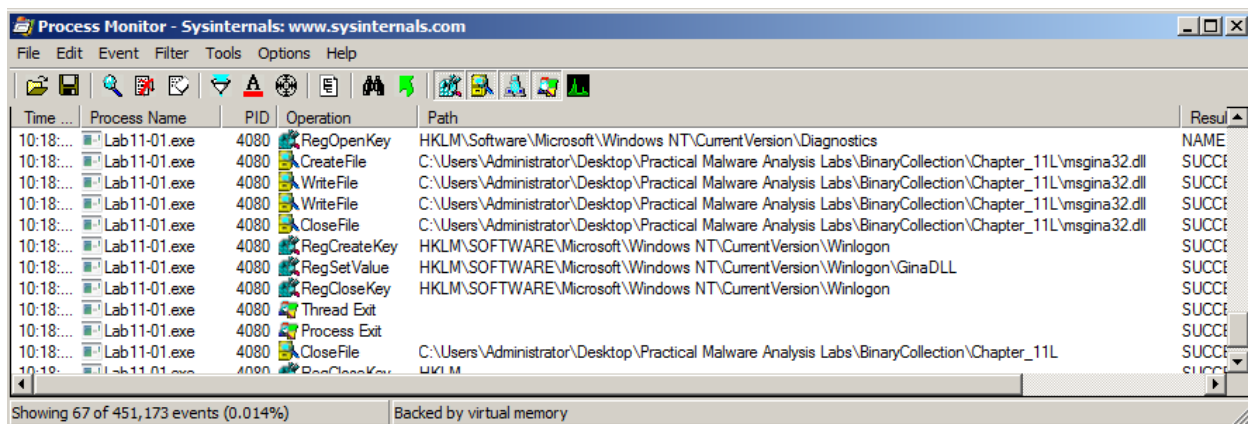
Dynamic Analysis with Procmon

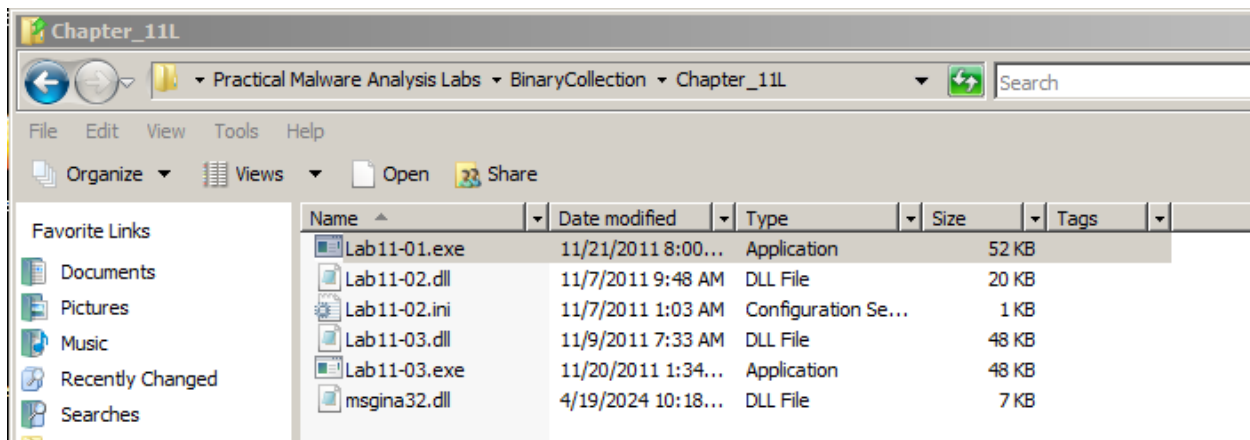
Run the malware in a virtual machine, while running Procmon to see what it does.

In Procmon, click **Filter**, "**Reset Filter**".

Click **Filter**, **Filter**. Filter for a "**Process Name**" of **Lab11-01.exe**.

- CreateFile ... msgina32.dll
- RegCreateKey HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon
- RegSetValue HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\GinaDLL





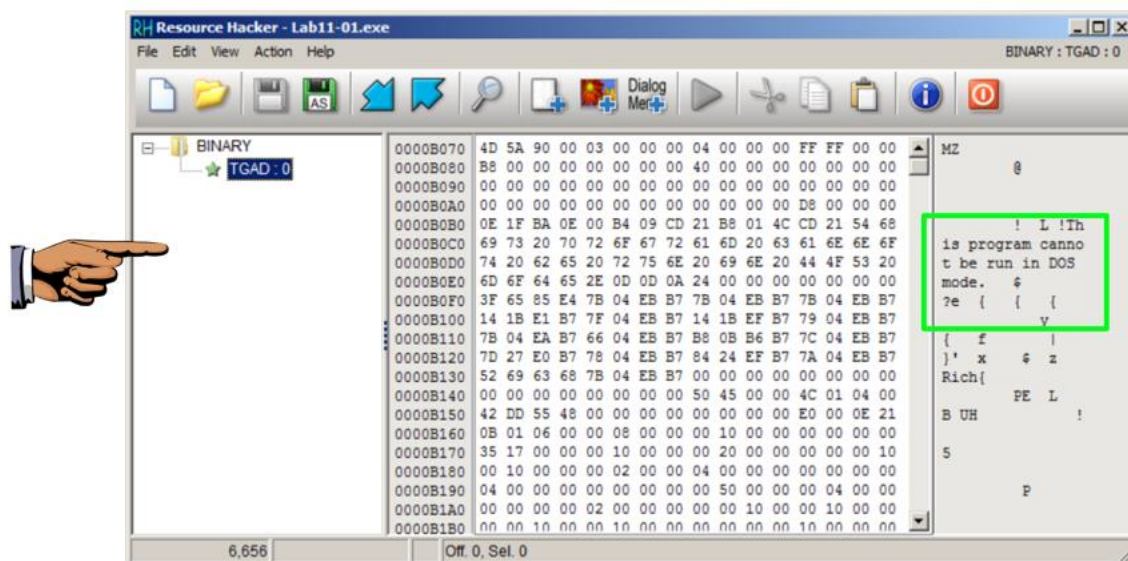
Resource Hacker

Download Resource Hacker here:

<http://www.angusj.com/resourcehacker/>

Open **Lab11-01.exe** in Resource Hacker.

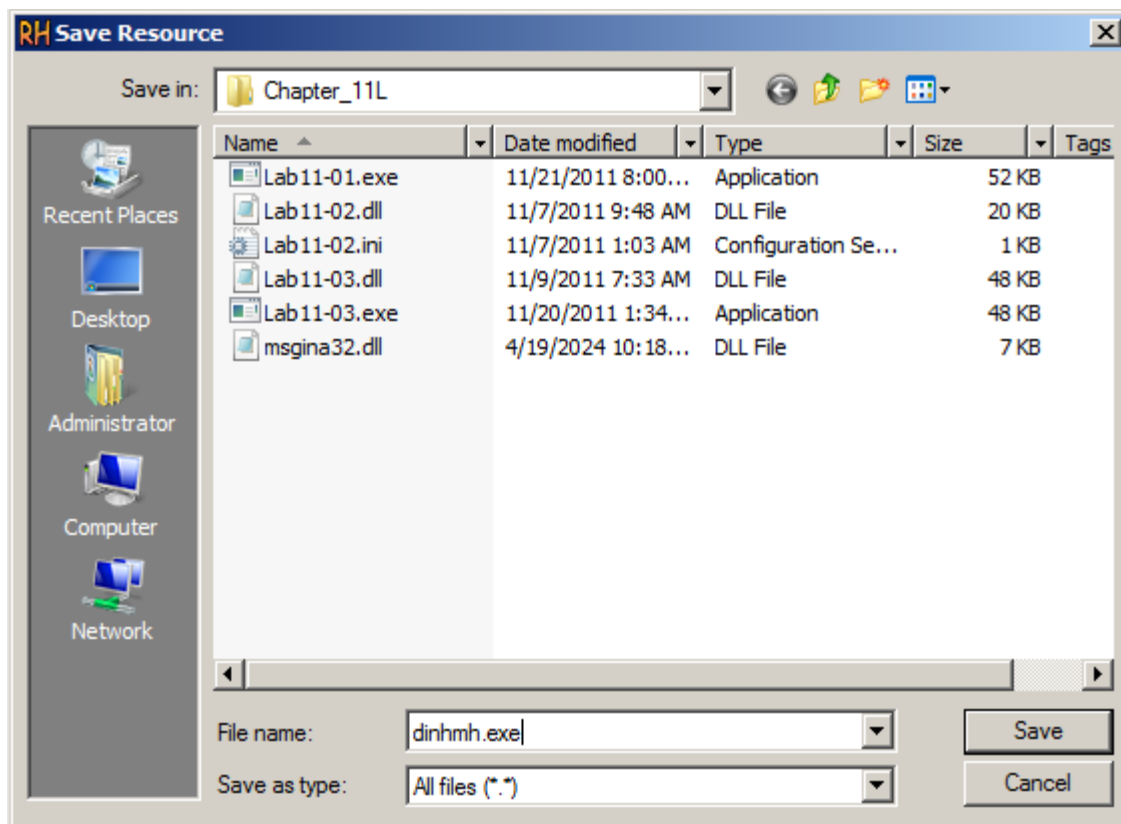
The "**BINARY TGAD 0**" starts with **MZ** and contains the telltale text "This program cannot be run in DOS mode", as shown below--this is an EXE file.



In Resource Hacker, in the left pane, click **0** to highlight it, as shown above.

Click **Action, Save Resource as a binary file...**.

Save the file as **YOURNAME-TGAD0.exe**, replacing the text "YOURNAME" with your own name.



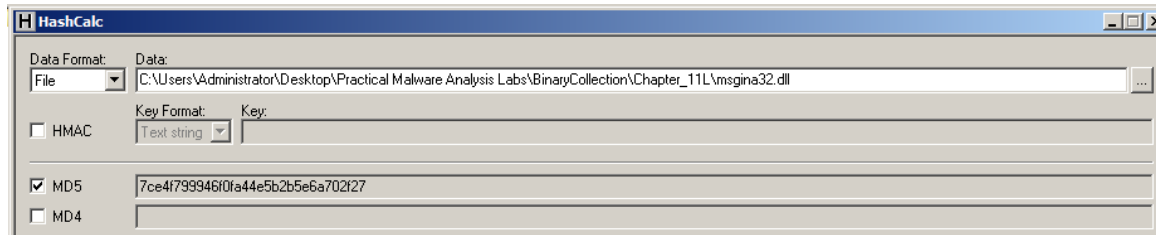
HashCalc

If you don't have it, get HashCalc here:

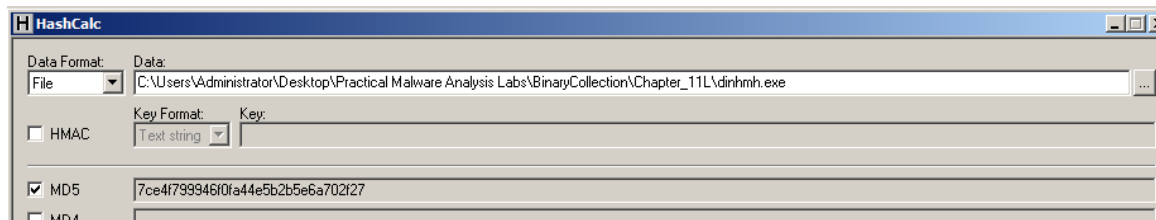
<http://www.slavasoft.com/hashcalc/>

Calculate the MD5 hash of the msgina32.dll file created by running the malware.

The MD5 hash begins with **7ce4**, as shown below.



Calculate the MD5 hash of the **YOURNAME-TGAD0.exe** file, as shown below.



LAB 2:

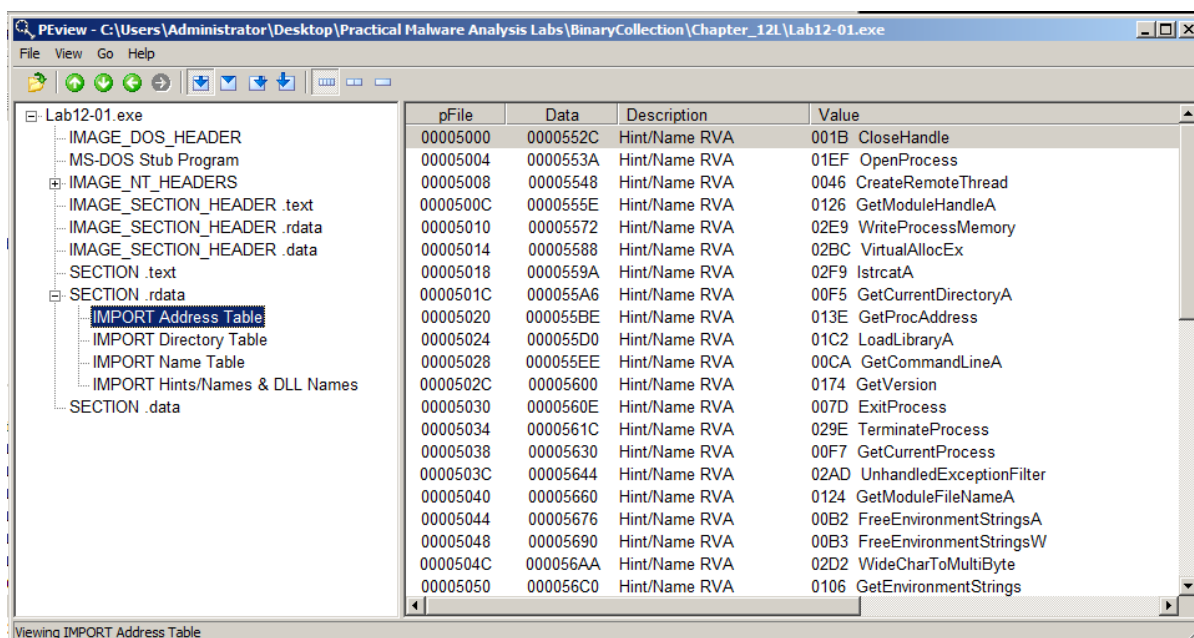
What you need: The Windows 2008 Server virtual machine we have been using.

Purpose: Analyze malware behavior

Imports

Examine **Lab12-01.exe** in PEView. Find these three imports, which are used in process injection:

- **CreateRemoteThread**
- **WriteProcessMemory**
- **VirtualAllocEx**



Strings

Examine the strings in **Lab12-01.exe**. Find these three strings, which show the process being injected, the DLL file used, and *psapi.dll*, which is used for

process enumeration:

- **explorer.exe**
- **Lab12-01.dll**
- **psapi.dll**

```
Administrator: C:\Windows\system32\cmd.exe
LCMapStringW
GetStringTypeA
GetStringTypeW
explorer.exe
<unknown>
LoadLibraryA
kernel32.dll
Lab12-01.dll
EnumProcesses
GetModuleBaseNameA
psapi.dll
EnumProcessModules
XSE
.SE
\RE
0RE
`QE
8QE
<QE
<cE
<cE

<<<<<      H

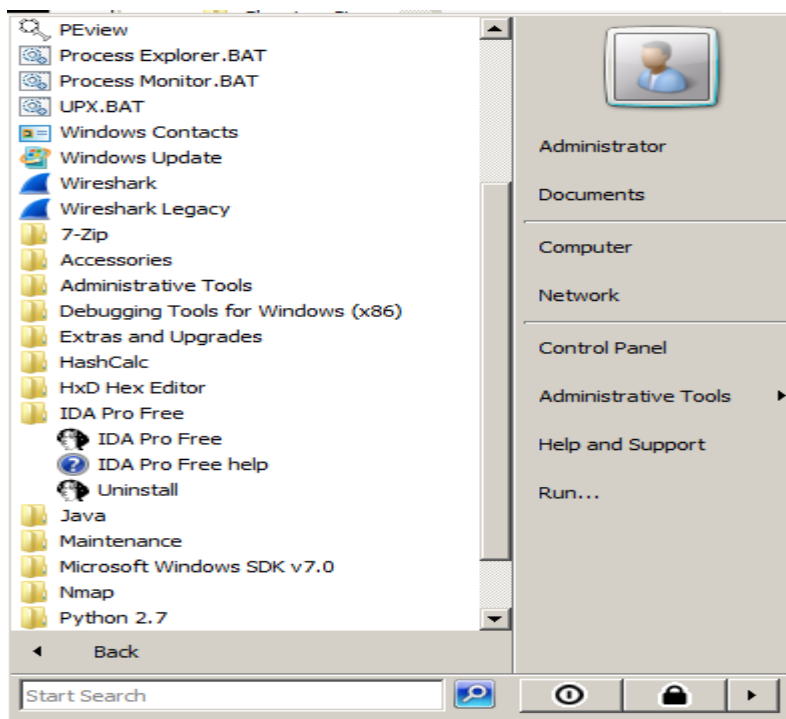
C:\Users\Administrator\Desktop\Practical Malware Analysis Labs\BinaryCollection\
Chapter_12L>
```

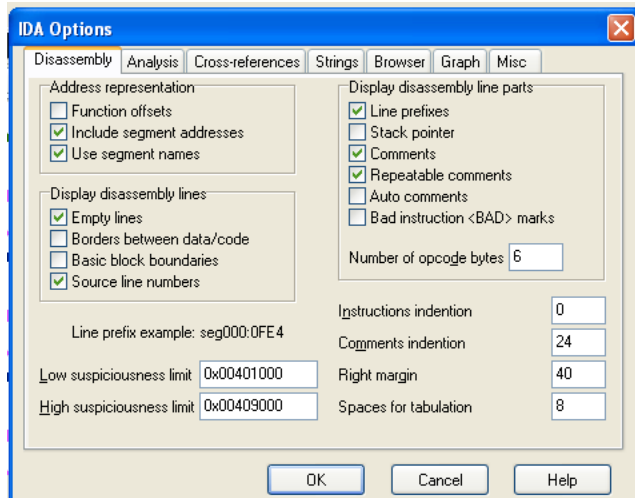
IDA Pro

Load **Lab12-01.exe** in IDA Pro Free.

Click **Options, General**.

Check "**Line Prefixes**" and set the "Number of opcode bytes" to **6**, as shown below.





Find the code shown below, near the start of main():

```

0040111F 68 80 60 40 00 push offset ProcName ; "EnumProcessModules"
00401124 68 A4 60 40 00 push offset LibFileName ; "psapi.dll"
00401129 FF 15 24 50 40 00 call ds:LoadLibraryA
0040112F 50 push eax ; hModule
00401130 FF 15 20 50 40 00 call ds:GetProcAddress
00401136 A3 14 87 40 00 mov dword_408714, eax
0040113B 68 90 60 40 00 push offset aGetmodulebasen ; "GetModuleBaseNameA"
00401140 68 A4 60 40 00 push offset LibFileName ; "psapi.dll"
00401145 FF 15 24 50 40 00 call ds:LoadLibraryA
0040114B 50 push eax ; hModule
0040114C FF 15 20 50 40 00 call ds:GetProcAddress
00401152 A3 0C 87 40 00 mov dword_40870C, eax
00401157 68 80 60 40 00 push offset aEnumprocesses ; "EnumProcesses"
0040115C 68 A4 60 40 00 push offset LibFileName ; "psapi.dll"
00401161 FF 15 24 50 40 00 call ds:LoadLibraryA
00401167 50 push eax ; hModule
00401168 FF 15 20 50 40 00 call ds:GetProcAddress
0040116E A3 10 87 40 00 mov dword_408710, eax

```

This code uses *psapi* three times to locate a Windows API function and store its address in a numerical address. This obfuscates the code, so later calls to

these functions will be difficult to recognize.

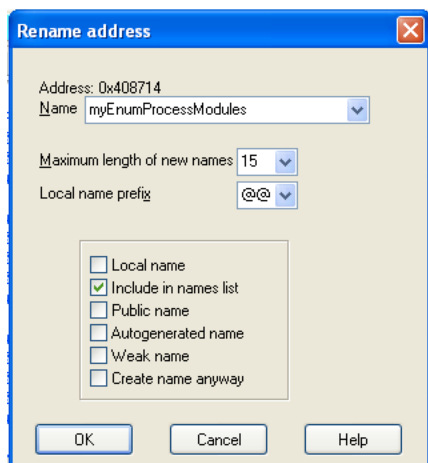
We'll assign labels to these memory addresses in IDA Pro to make later analysis easier.

The first section of code assigns a pointer to the function EnumProcessModules.

In the line starting with address 00401136, right-click **dword_408714** and click **Rename**.

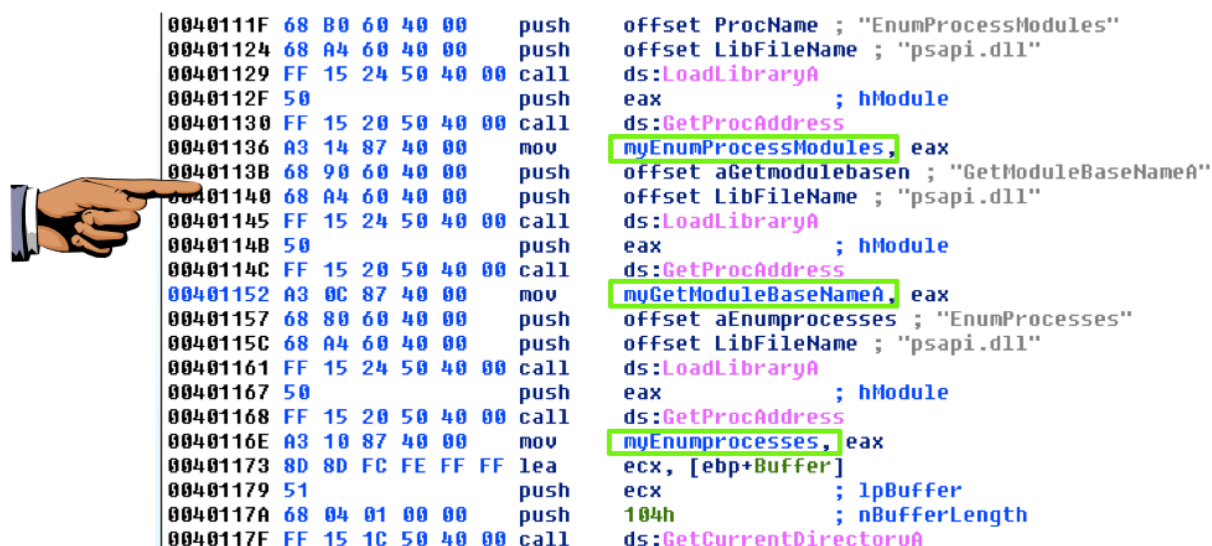
Enter a new Name of **myEnumProcessModules** in the box, as shown below. Click **OK**.

Increase the length limit when you are prompted to.

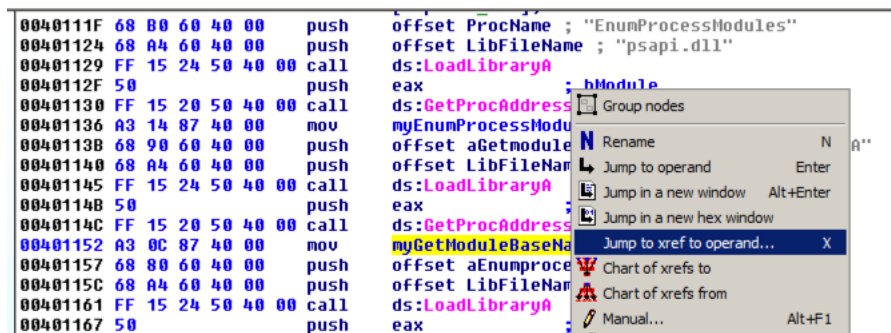


Repeat the process to rename **dword_40870C** to **myGetModuleBaseNameA**

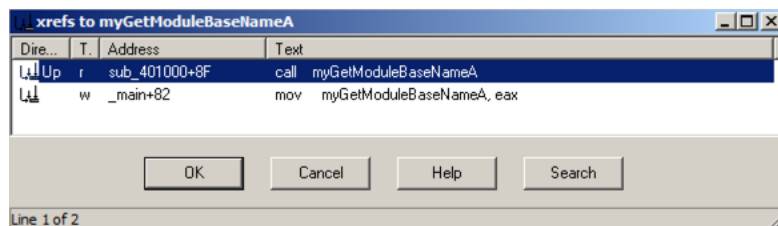
Repeat the process to rename **dword_408710** to **myEnumProcesses**



Right-click **myGetModuleBaseNameA** and click **"Jump tp xrefs of operand"** , as shown below:



An xrefs box pops up, as shown below, showing that this address is only used once, in sub_401000.

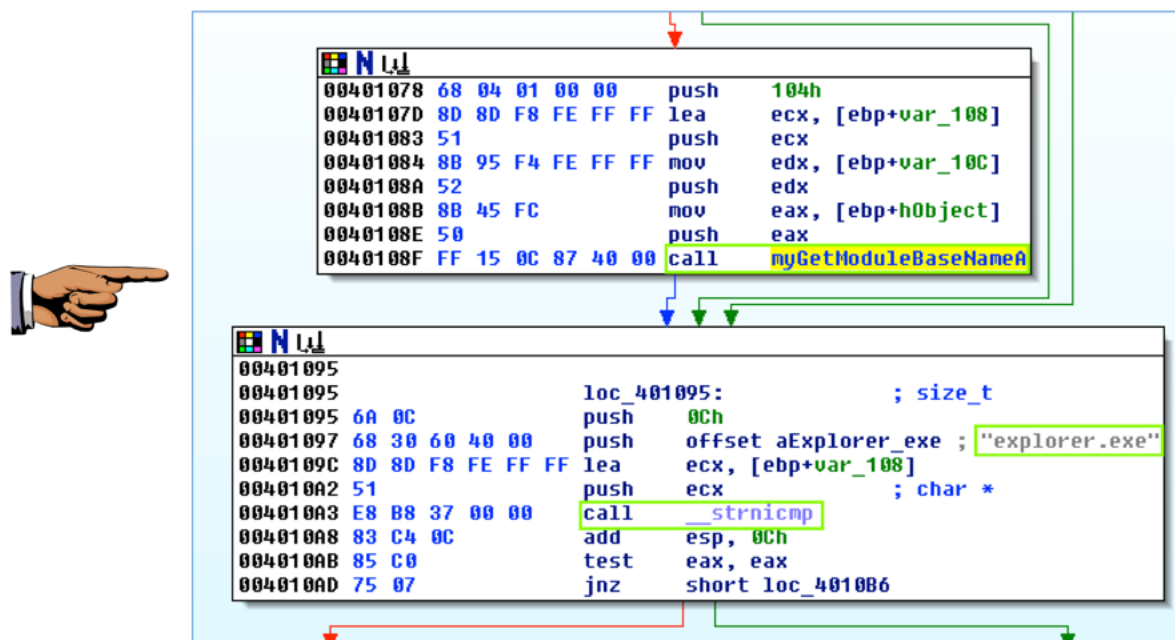


In the xrefs box, click **OK**.

This routine enumerates the modules and compares each module name to "explorer.exe", to find the module into which to inject code.

Make sure you can see these three items on your screen, as shown below:

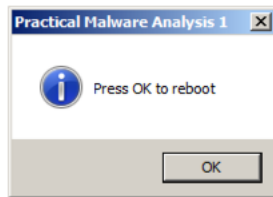
- **call myGetModuleBaseNameA**
- **"explorer.exe"**
- **call __strnicmp**



Process Explorer

Close IDA Pro. Double-click **Lab12-01.exe** to run the malware.

A box pops up saying "Press OK to reboot". as shown below. Drag this box out of the way.



Open **Process Explorer**.

In the upper pane, scroll to the bottom of the list. Click **explorer.exe** to select it.

In Process Explorer, from the menu bar, click **View** and make sure "**Show Lower Pane**" is checked.

In Process Explorer, from the menu bar, click **View**, "**Lower Pane View**", **DLLs**.

In the lower pane, find the **Lab12-01.dll** that has been injected into explorer.exe, as shown below.

