

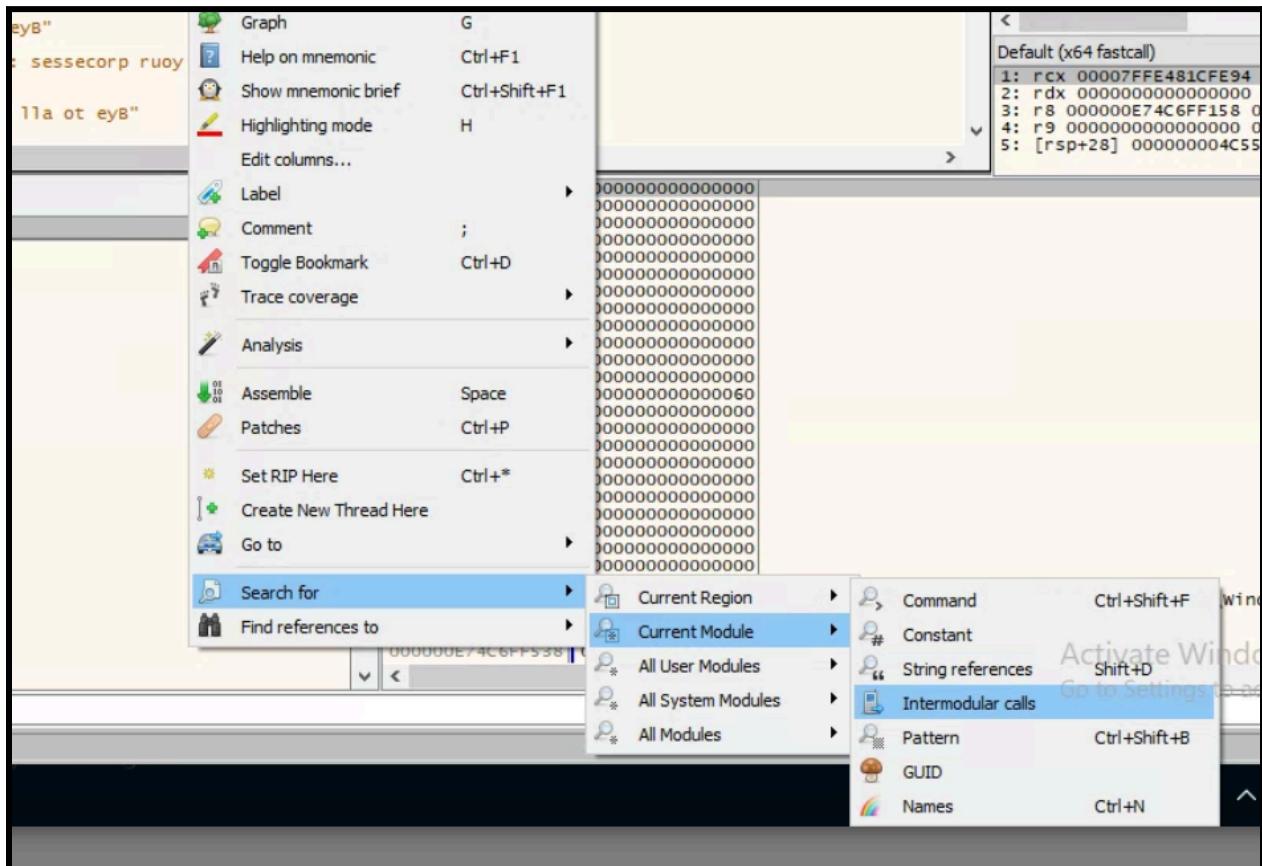
## Overview

The purpose of this lab is to patch and analyze a sample of malware that interferes with a variety of our forensic tools and messes with the mouse. Once patched, a full analysis of the sample can be done to see how the sample works and be documented.

## Deliverable 1 - Patching the Sample

### Patching the Mouse Button Switching

In order to patch the mouse button switch, you need to find the function within the sample code that switches the button. To find the function call right click, search for, current module, intermodule cells, then filter the results at the bottom with the word mouse. This can be seen in the screenshot below.



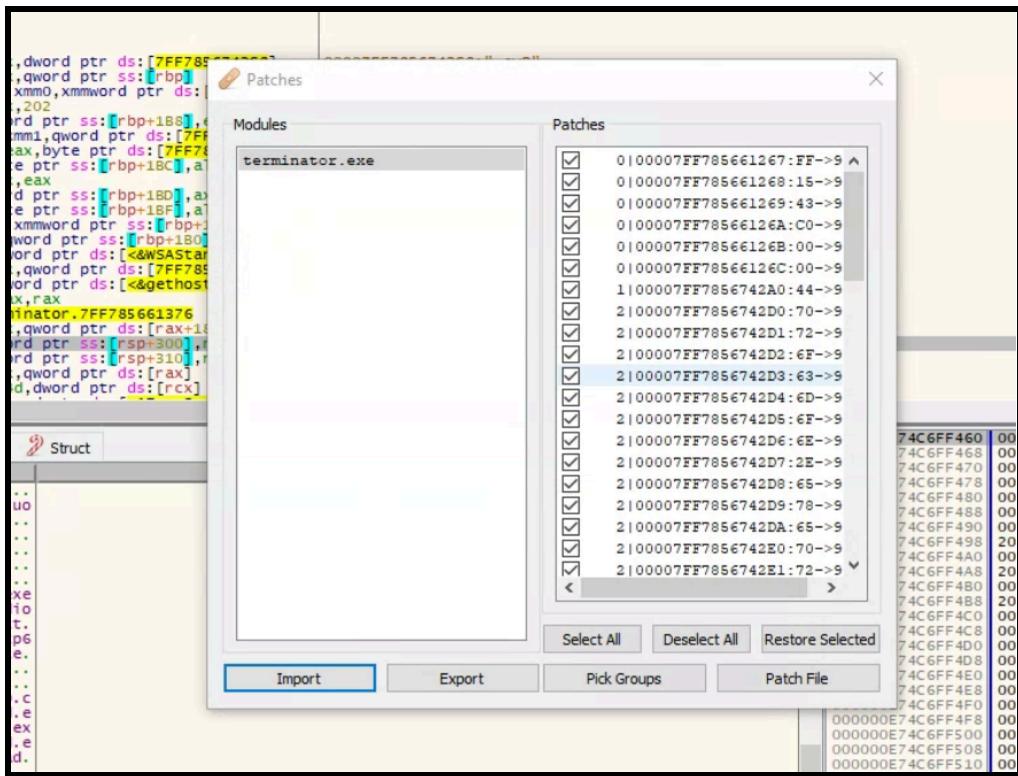
Clicking this button allows us to view all the functions / calls within x65 debug. The bottom of the window has a filter we can use to find the specific mouse button swap function.

The screenshot shows the Immunity Debugger interface with the 'Calls' tab selected. A single call instruction is highlighted: `call qword ptr ds:[<&SwapMouseButton>]`. The destination of this call is `<user32.SwapMouseButton> (00007FFE4592EA80)`. The assembly window is mostly empty, and the search bar at the bottom contains the text 'mouse'.

The filter and function can be seen in the screenshot above. We can then follow the address and nop (no operation) out the function. This can be seen in the screenshot below

```
● 00007FF78566124E 45:3C0 xor r8d,r8d
● 00007FF785661251 FF15 A9BD0000 call qword ptr ds:[<&RegSetValueExA>]
● 00007FF785661257 48:84C24 58 mov rcx,qword ptr ss:[rsp+58]
● 00007FF78566125C FF15 AEBD0000 call qword ptr ds:[<&RegCloseKey>]
● 00007FF785661262 B9 01000000 mov ecx,1
● 00007FF785661267 90 nop
● 00007FF785661268 90 nop
● 00007FF785661269 90 nop
● 00007FF78566126A 90 nop
● 00007FF78566126B 90 nop
● 00007FF78566126C 90 nop
● 00007FF78566126D 8B05 55300100 mov eax,dword ptr ds:[7FF7856742C8] 00007FF7856742C8:" eyB"
● 00007FF785661273 48:8D55 00 lea rdx,qword ptr ss:[rbp]
● 00007FF785661277 0F1005 32300100 movups xmm0,xmmword ptr ds:[7FF7856742B0(00007FF7856742B0):] sessecorp ruoy 1
● 00007FF78566127E B9 02020000 mov ecx,202
● 00007FF785661283 8985 B8010000 mov dword ptr ss:[rbp+188],eax
```

This screenshot shows the strings of the sample, we can then right click and follow in disassembler where it says procmon.exe in the actual code. Once in the code we can nop out the first letter of procmon to patch by placing in the value 90. The end result can be seen below.



The screenshot above shows the patching of the application closes within the malware. This can be done by finding the reference in the strings and rewriting the strings with 90.

Now that the malware has been patched static and dynamic analysis can be done.

## Deliverable 2 - Static Analysis

### Metadata

**Sample Name:** Terminator.exe

**SHA 256 File Hash:**

3ECC3DE52A8BECF12965DAC256B02D7995B653FCE805359D27452B6DF8790F25

**Compiler Stamp:** Fri Sep 27 12:08:09 2019 | UTC

**Subsystem Type:** Console

## Libraries / DLLS

library (5)	flag (2)	bound (0)	type (1)	functions (86)	description
kernel32.dll	-	-	implicit	77	Windows NT BASE API Client DLL
user32.dll	-	-	implicit	1	Multi-User Windows USER API Client DLL
advapi32.dll	-	-	implicit	4	Advanced Windows 32 Base API
iphlpapi.dll	x	-	implicit	2	IP Helper API
ws2_32.dll	x	-	implicit	2	Windows Socket 2.0 32-Bit DLL

This screenshot shows the different imported dlls within the sample. The iphlpapi.dll and ws2\_32.dll libraries are a bit suspicious as they both have to do with IP and creating windows sockets. The functions from these should be looked into.

## Strings

encoding (2)	size (bytes)	location	flag (23)	hint (107)	group (12)	value (2894)
ascii	13	0x00013D62	x	function	registry	<a href="#">RegSetValueEx</a>
ascii	14	0x00013D74	x	function	registry	<a href="#">RegCreateKeyEx</a>
ascii	12	0x00013DA2	x	function	network	<a href="#">IcmpSendEcho</a>
ascii	14	0x00013DB2	x	function	network	<a href="#">IcmpCreateFile</a>
ascii	8	0x00013C6A	x	function	file	<a href="#">MoveFile</a>
ascii	10	0x00013CE8	x	function	file	<a href="#">DeleteFile</a>
ascii	9	0x0001405E	x	function	file	<a href="#">WriteFile</a>
ascii	15	0x00014122	x	function	file	<a href="#">FindFirstFileEx</a>
ascii	12	0x00014136	x	function	file	<a href="#">FindNextFile</a>
ascii	14	0x00013C76	x	function	execution	<a href="#">Process32First</a>
ascii	16	0x00013C88	x	function	execution	<a href="#">TerminateProcess</a>
ascii	11	0x00013C9C	x	function	execution	<a href="#">OpenProcess</a>
ascii	24	0x00013CAA	x	function	execution	<a href="#">CreateToolhelp32Snapshot</a>
ascii	13	0x00013CF6	x	function	execution	<a href="#">Process32Next</a>
ascii	13	0x00013D14	x	function	execution	<a href="#">CreateProcess</a>
ascii	19	0x00013EA4	x	function	execution	<a href="#">GetCurrentProcessId</a>
ascii	18	0x00013EBA	x	function	execution	<a href="#">GetCurrentThreadId</a>
ascii	21	0x00014170	x	function	execution	<a href="#">GetEnvironmentStrings</a>
ascii	22	0x000141A4	x	function	execution	<a href="#">SetEnvironmentVariable</a>
ascii	14	0x0001403C	x	function	exception	<a href="#">RaiseException</a>
ascii	17	0x000140BA	x	function	dynamic-library	<a href="#">GetModuleHandleEx</a>
ascii	22	0x00013DF2	x	function	diagnostic	<a href="#">RtlLookupFunctionEntry</a>
ascii	3	0x0001D9E0	x	-	-	212

Some notable strings are RegSetValueEx, CreateKey, ICMP Create File, Moving, writing, and deleting file, etc. As mentioned before we can see some of the different functions from the suspicious dlls within the strings. ICMPsendEcho is from the iphlpapi.dll. (This function is explained in deliverable 3)

```

log10
Software\Microsoft\Windows\CurrentVersion\Policies\System
DisableTaskMgr
): sessecorp ruoy lla ot eyB
procmon.exe
procmon64.exe
wireshark.exe
procexp.exe
CFF Explorer.exe
pestudio.exe
bintext.exe
procexp64.exe
cmd.exe
explorer.exe
notepad.exe
www.malware430.com
notpad.exe
paint.exe
notpad.exe x
Notepad
GCTL

```

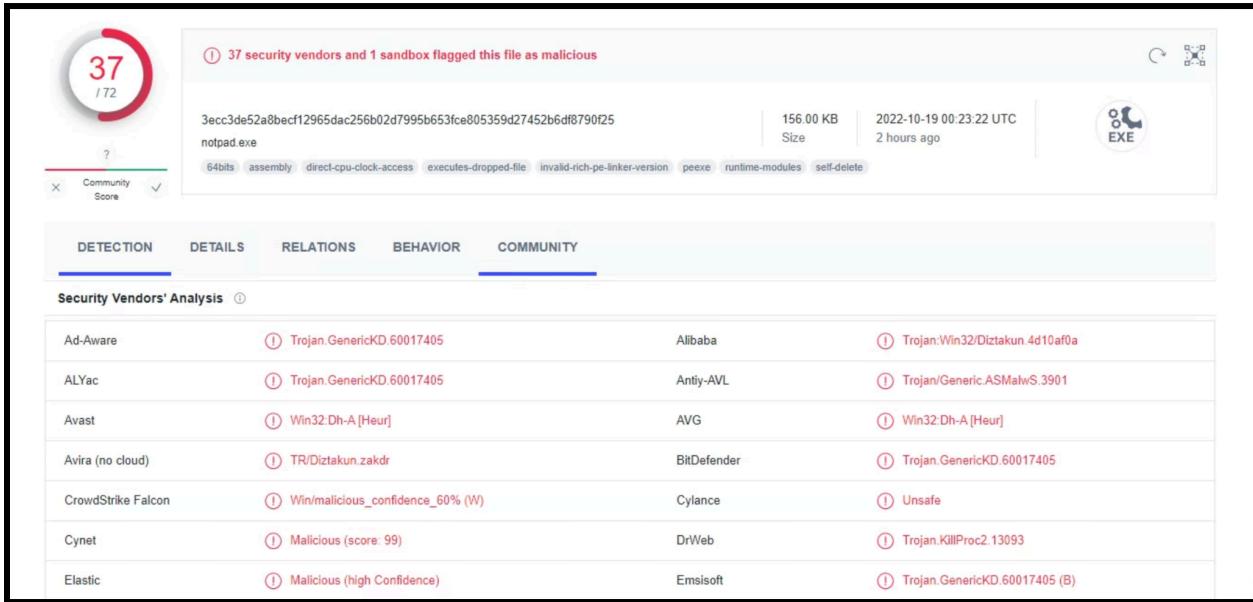
Some more strings can be seen above. A registry key location, disabling task manager and a list of many forensic programs. These values are good to investigate. Since we patched the sample we know their purpose was to close the active processes with these in it.

## Entropy and Packing

Entropy		Bytes				
Regions		Offset	Size	Entropy	Status	Name
0000000000000000	00000000000000400			2.92231	not packed	PE Header
00000000000000400	000000000000b200			6.37322	not packed	Section(0)['.text']
000000000000b600	0000000000008e00			4.77450	not packed	Section(1)['.rdata']
00000000000014400	000000000000a00			1.82425	not packed	Section(2)['.data']
00000000000014e00	000000000000e00			4.58258	not packed	Section(3)['.pdata']
00000000000015c00	00000000000010c00			4.87956	not packed	Section(4)['.rsrc']
00000000000026800	0000000000000800			4.74870	not packed	Section(5)['.reloc']

Using Detect It Easy, we can see that there are 5 different sections within the sample. They use standard names and the entropy suggests the sample is not packed.

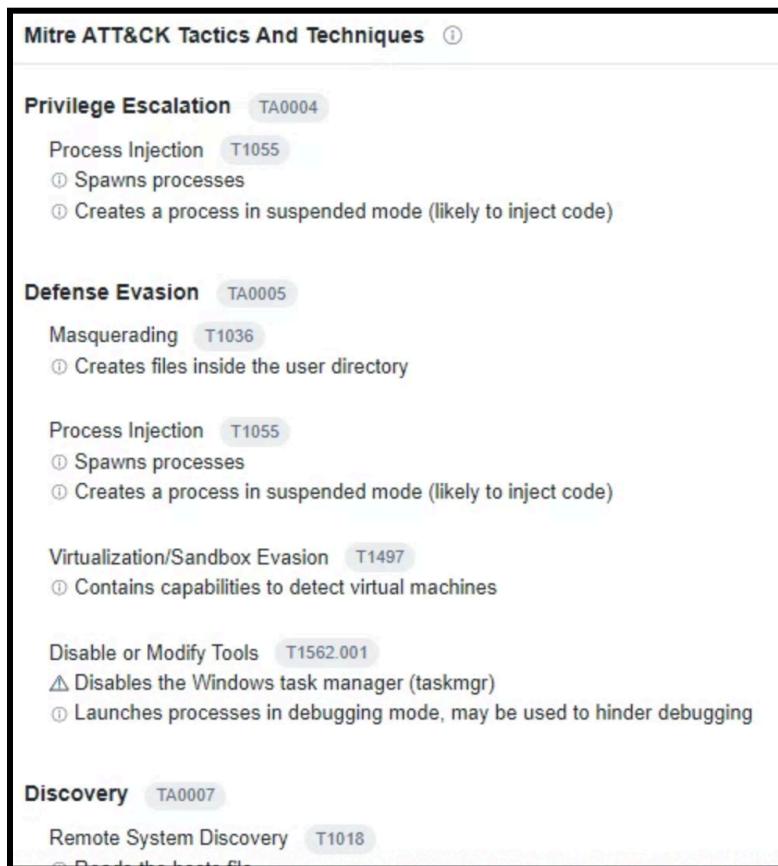
# Virus Total



The screenshot shows the Virus Total analysis interface. The top left displays a circular icon with a '37' and '72' score. The top center shows a message: '37 security vendors and 1 sandbox flagged this file as malicious'. Below this is the file information: '3ecc3de52a8becf12965dac256b02d7995b653fce805359d27452b6df8790f25 notepad.exe', '156.00 KB', '2022-10-19 00:23:22 UTC', '2 hours ago', and a file type icon for 'EXE'. The bottom section is a table titled 'Security Vendors' Analysis' with columns for vendor, detection, and notes. The table lists 10 vendors and their findings.

Vendor	Detection	Notes
Ad-Aware	① Trojan.GenericKD.60017405	Alibaba ① Trojan:Win32/Diztakun.4d10af0a
ALYac	① Trojan.GenericKD.60017405	Antiy-AVL ① Trojan/Generic.ASMalwS.3901
Avast	① Win32.Dh-A [Heur]	AVG ① Win32.Dh-A [Heur]
Avira (no cloud)	① TR/Diztakun.zakdr	BitDefender ① Trojan.GenericKD.60017405
CrowdStrike Falcon	① Win/malicious_confidence_60% (W)	Cylance ① Unsafe
Cynet	① Malicious (score: 99)	DrWeb ① Trojan.KillProc2.13093
Elastic	① Malicious (high Confidence)	Emsisoft ① Trojan.GenericKD.60017405 (B)

The sample has a 37 out of 72 score on virus total. It is thought to be a form of trojan. We know from analyzing the assembly that it closes any process with the names of forensic programs, switches the mouse button, and adds persistence registry keys.

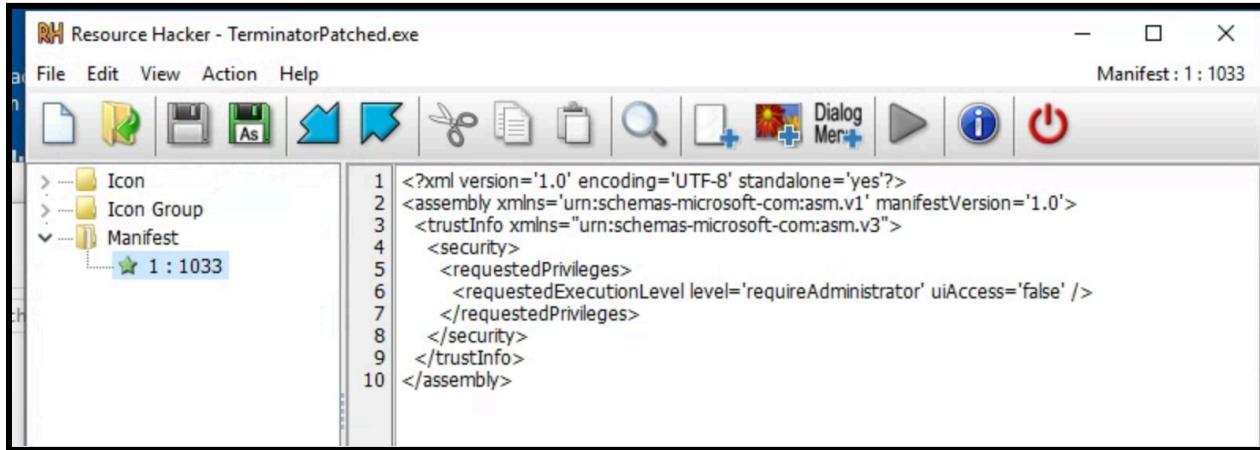


The screenshot shows the Mitre ATT&CK Tactics And Techniques analysis interface. The top section is titled 'Mitre ATT&CK Tactics And Techniques'. Below it, the 'Privilege Escalation' section lists 'Process Injection' (T1055) and 'Masquerading' (T1036). The 'Defense Evasion' section lists 'Process Injection' (T1055) and 'Virtualization/Sandbox Evasion' (T1497). The 'Disable or Modify Tools' section lists 'Disable or Modify Tools' (T1562.001). The 'Discovery' section lists 'Remote System Discovery' (T1018).

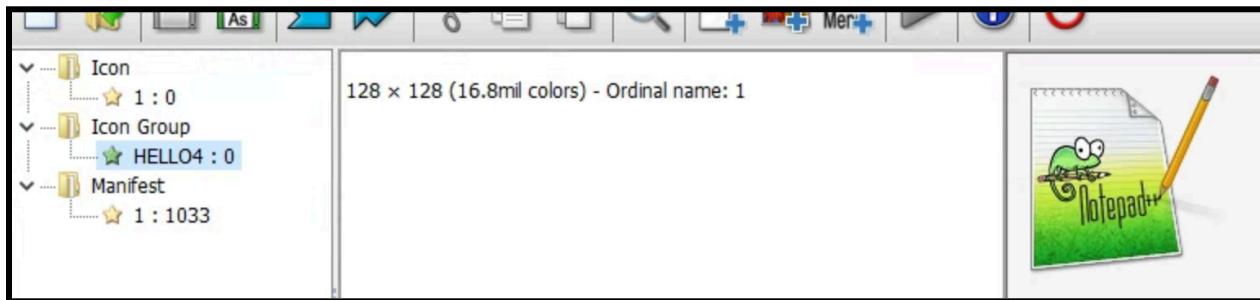
This screenshot shows some of the Mitre Att&ck Techniques that the sample uses such as process injection, masquerading, sandbox evasion, and more.

## Resource Hacker

Resource hacker is helpful when looking at the manifest of the file. This gives information such as the schema used, the execution levels, and sometimes more. The manifest for the sample can be seen in the screenshot below.



The next screenshot is the image file for the samples icon. It is a notepad++ icon. This lets us know that the program is trying to disguise itself as notepad.exe. This is further correlated in the fact that it creates a new process called notepad once it is run.

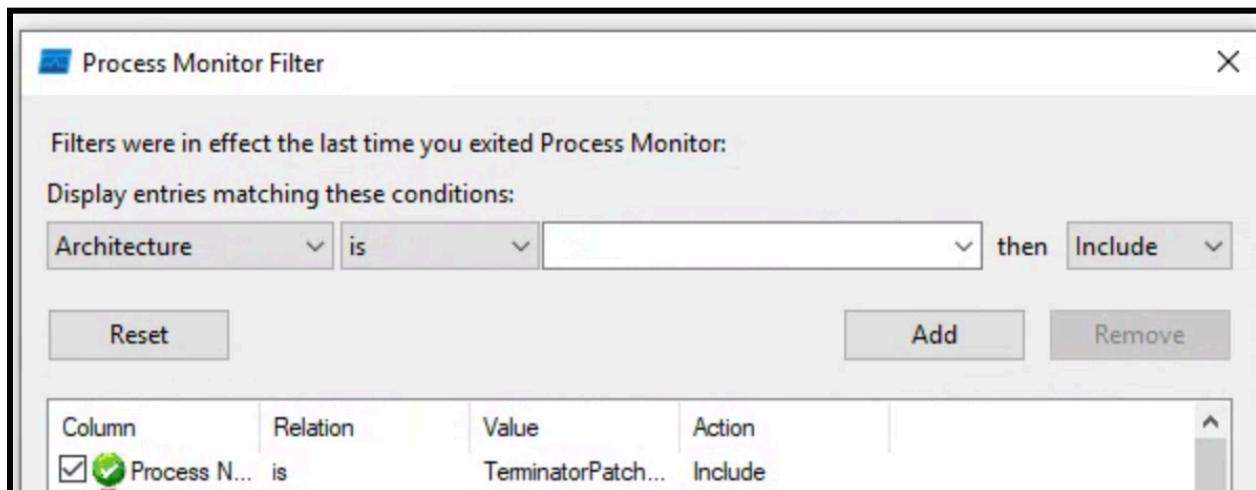


## Deliverable 2 - Dynamic Analysis

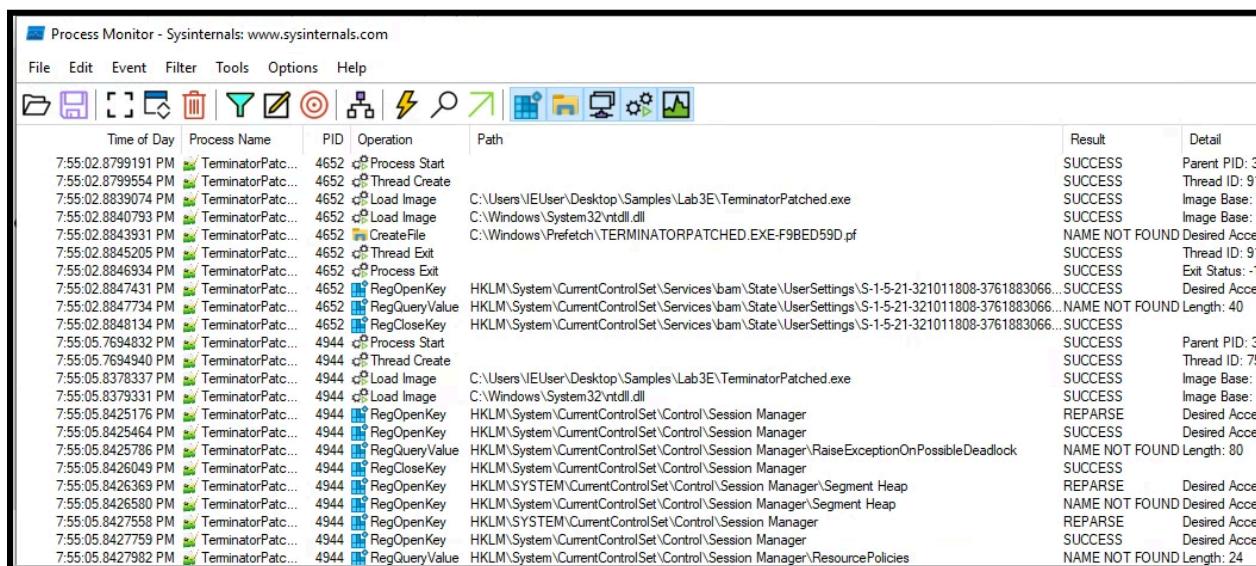
The first step to dynamic analysis is to run the samples with various tools that can capture information regarding the samples. One tool that does this is Process Monitor. Process Monitor can capture artifacts like registry activity, file system activity, network activity, and process activity.

### Process Monitor

In order to set up Process Monitor to capture the samples activity, a filter with samples process name should be applied. This can be seen in the screenshot below next to the green check mark. A filter with the process name TerminatorPatched.exe has been applied.



Next, analysis of the activity should take place. This can be seen in the screenshot below.

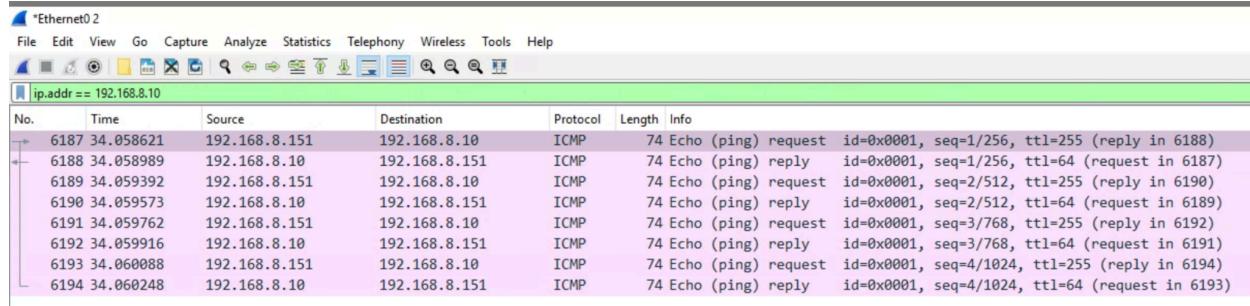


Time of Day	Process Name	PID	Operation	Path	Result	Detail
7:55:02.8799191 PM	TerminatorPatched...	4652	Process Start		SUCCESS	Parent PID: 3
7:55:02.8799554 PM	TerminatorPatched...	4652	Thread Create		SUCCESS	Thread ID: 91
7:55:02.8839074 PM	TerminatorPatched...	4652	Load Image	C:\Users\IEUser\Desktop\Samples\Lab3E\TerminatorPatched.exe	SUCCESS	Image Base: 0
7:55:02.8840793 PM	TerminatorPatched...	4652	Load Image	C:\Windows\System32\ntdll.dll	SUCCESS	Image Base: 0
7:55:02.8843931 PM	TerminatorPatched...	4652	CreateFile	C:\Windows\Prefetch\TERMINATORPATCHED.EXE-F9BED59D.pdf	NAME NOT FOUND	Desired Access
7:55:02.8845205 PM	TerminatorPatched...	4652	Thread Exit		SUCCESS	Thread ID: 91
7:55:02.8846934 PM	TerminatorPatched...	4652	Process Exit		SUCCESS	Exit Status: -1
7:55:02.8847431 PM	TerminatorPatched...	4652	RegOpenKey	HKEY\SYSTEM\CurrentControlSet\Services\bam\State\UserSettings\S-1-5-21-321011808-3761883066...	SUCCESS	Desired Access
7:55:02.8847734 PM	TerminatorPatched...	4652	RegQueryValue	HKEY\SYSTEM\CurrentControlSet\Services\bam\State\UserSettings\S-1-5-21-321011808-3761883066...	NAME NOT FOUND	Length: 40
7:55:02.8848134 PM	TerminatorPatched...	4652	RegCloseKey	HKEY\SYSTEM\CurrentControlSet\Services\bam\State\UserSettings\S-1-5-21-321011808-3761883066...	SUCCESS	
7:55:05.7694940 PM	TerminatorPatched...	4944	Process Start		SUCCESS	Parent PID: 3
7:55:05.7694940 PM	TerminatorPatched...	4944	Thread Create		SUCCESS	Thread ID: 75
7:55:05.8378337 PM	TerminatorPatched...	4944	Load Image	C:\Users\IEUser\Desktop\Samples\Lab3E\TerminatorPatched.exe	SUCCESS	Image Base: 0
7:55:05.8379331 PM	TerminatorPatched...	4944	Load Image	C:\Windows\System32\ntdll.dll	SUCCESS	Image Base: 0
7:55:05.8425176 PM	TerminatorPatched...	4944	RegOpenKey	HKEY\SYSTEM\CurrentControlSet\Control\Session Manager	REPARSE	Desired Access
7:55:05.8425464 PM	TerminatorPatched...	4944	RegOpenKey	HKEY\SYSTEM\CurrentControlSet\Control\Session Manager	SUCCESS	Desired Access
7:55:05.8425786 PM	TerminatorPatched...	4944	RegQueryValue	HKEY\SYSTEM\CurrentControlSet\Control\Session Manager\RaiseExceptionOnPossibleDeadlock	NAME NOT FOUND	Length: 80
7:55:05.8426049 PM	TerminatorPatched...	4944	RegCloseKey	HKEY\SYSTEM\CurrentControlSet\Control\Session Manager	SUCCESS	
7:55:05.8426363 PM	TerminatorPatched...	4944	RegOpenKey	HKEY\SYSTEM\CurrentControlSet\Control\Session Manager\Segment Heap	REPARSE	Desired Access
7:55:05.8426580 PM	TerminatorPatched...	4944	RegOpenKey	HKEY\SYSTEM\CurrentControlSet\Control\Session Manager\Segment Heap	NAME NOT FOUND	Desired Access
7:55:05.8427558 PM	TerminatorPatched...	4944	RegOpenKey	HKEY\SYSTEM\CurrentControlSet\Control\Session Manager	REPARSE	Desired Access
7:55:05.8427759 PM	TerminatorPatched...	4944	RegOpenKey	HKEY\SYSTEM\CurrentControlSet\Control\Session Manager	SUCCESS	Desired Access
7:55:05.8427982 PM	TerminatorPatched...	4944	RegQueryValue	HKEY\SYSTEM\CurrentControlSet\Control\Session Manager\ResourcePolicies	NAME NOT FOUND	Length: 24

Instead of parsing through each piece of information individually, another program called ProcDot will be used to visualize the execution steps of the sample. Before ProcDot can be run, another two tools need to be run first.

## Wireshark

Many forms of malware will attempt to reach out to foreign network addresses to pull files, create reverse connections, and more. In order to capture this traffic we can use a tool called Wireshark. This tool allows for the capturing of network packets and the screenshot below shows the important ones for this sample.

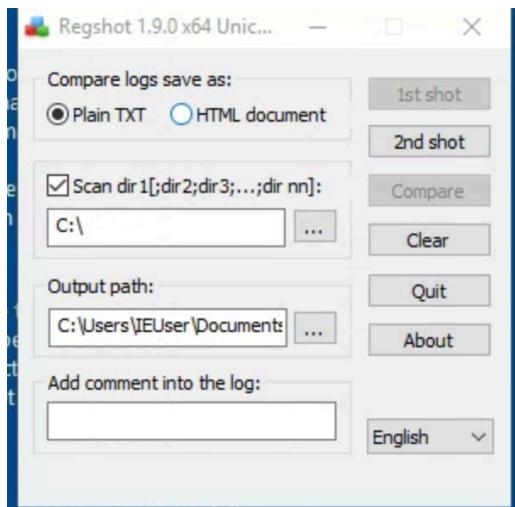


No.	Time	Source	Destination	Protocol	Length	Info
6187	34.058621	192.168.8.151	192.168.8.10	ICMP	74	Echo (ping) request id=0x0001, seq=1/256, ttl=255 (reply in 6188)
6188	34.058989	192.168.8.10	192.168.8.151	ICMP	74	Echo (ping) reply id=0x0001, seq=1/256, ttl=64 (request in 6187)
6189	34.059392	192.168.8.151	192.168.8.10	ICMP	74	Echo (ping) request id=0x0001, seq=2/512, ttl=255 (reply in 6190)
6190	34.059573	192.168.8.10	192.168.8.151	ICMP	74	Echo (ping) reply id=0x0001, seq=2/512, ttl=64 (request in 6189)
6191	34.059762	192.168.8.151	192.168.8.10	ICMP	74	Echo (ping) request id=0x0001, seq=3/768, ttl=255 (reply in 6192)
6192	34.059916	192.168.8.10	192.168.8.151	ICMP	74	Echo (ping) reply id=0x0001, seq=3/768, ttl=64 (request in 6191)
6193	34.060088	192.168.8.151	192.168.8.10	ICMP	74	Echo (ping) request id=0x0001, seq=4/1024, ttl=255 (reply in 6194)
6194	34.060248	192.168.8.10	192.168.8.151	ICMP	74	Echo (ping) reply id=0x0001, seq=4/1024, ttl=64 (request in 6193)

The screenshot above has an IP filter on it for the malware430 domain. In an actual case, to find this information for a filter it could be found in strings or slowly filtering the packet capture. This information can be saved as a pcap file and loaded into procmon to visualize what the sample might be doing.

## RegShot

The next thing that needs to be done is taking a RegShot image before and after running the sample and compare it. This is helpful later on when we know more specifics on how the program works and can look at the specific keys that were changed.



The screenshot above shows the menu for regshot. The screenshot below shows a piece of the compared registry image files output from regshot.

```

~res-x64.txt - Notepad
File Edit Format View Help
Computer: MSEDEWIN10 , MSEDEWIN10
Username: IEUser , IEUser

-----
Keys added: 7
-----
HKU\S-1-5-21-321011808-3761883066-353627080-1000\Software\Microsoft\Windows\CurrentVersion\Explorer\SessionInfo\1\ApplicationViewManagement\W32:000000000002C0918
HKU\S-1-5-21-321011808-3761883066-353627080-1000\Software\Microsoft\Windows\CurrentVersion\Explorer\SessionInfo\1\ApplicationViewManagement\W32:00000000000350342
HKU\S-1-5-21-321011808-3761883066-353627080-1000\Software\Microsoft\Windows\CurrentVersion\Explorer\SessionInfo\1\ApplicationViewManagement\W32:000000000003607E4
HKU\S-1-5-21-321011808-3761883066-353627080-1000\Software\Microsoft\Windows\CurrentVersion\Explorer\SessionInfo\1\ApplicationViewManagement\W32:0000000000044091C
HKU\S-1-5-21-321011808-3761883066-353627080-1000\Software\Microsoft\Windows\CurrentVersion\Explorer\SessionInfo\1\ApplicationViewManagement\W32:0000000000048079A
HKU\S-1-5-21-321011808-3761883066-353627080-1000\Software\Microsoft\Windows\CurrentVersion\Explorer\SessionInfo\1\ApplicationViewManagement\W32:00000000000F904C2
HKU\S-1-5-21-321011808-3761883066-353627080-1000\Software\Microsoft\Windows\CurrentVersion\Policies\System

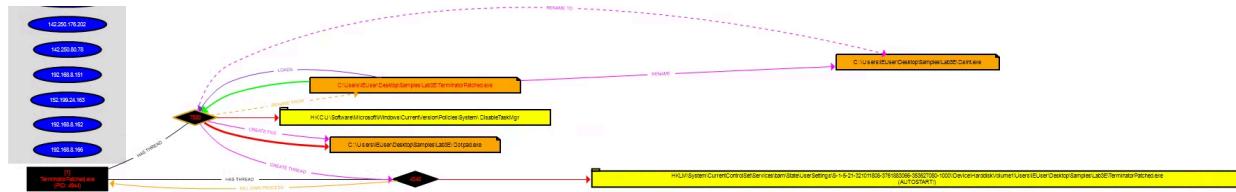
-----
Values added: 663
-----
HKLM\SYSTEM\ControlSet001\Services\bam\State\UserSettings\S-1-5-21-321011808-3761883066-353627080-1000\Device\HarddiskVolume1\Users\IEUser\AppData\Local\Temp\procexp1
HKLM\SYSTEM\ControlSet001\Services\bam\State\UserSettings\S-1-5-21-321011808-3761883066-353627080-1000\Device\HarddiskVolume1\Users\IEUser\Desktop\Samples\Lab3E\Term:
HKLM\SYSTEM\CurrentControlSet\Services\bam\State\UserSettings\S-1-5-21-321011808-3761883066-353627080-1000\Device\HarddiskVolume1\Users\IEUser\AppData\Local\Temp\procexp1
HKLM\SYSTEM\CurrentControlSet\Services\bam\State\UserSettings\S-1-5-21-321011808-3761883066-353627080-1000\Device\HarddiskVolume1\Users\IEUser\Desktop\Samples\Lab3E\Term:
HKU\S-1-5-21-321011808-3761883066-353627080-1000\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\FirstFolder\4: 43 00 3A 00 5C 00 55 00 73 00 65 00 72 00
HKU\S-1-5-21-321011808-3761883066-353627080-1000\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\OpenSavePidIMRU\hiv\4: 14 00 1F 50 E0 4F D0 20 EA 3A 69
00 68 00 6F 00 74 00 31 00 2E 00 68 00 69 00 76 00 75 00 00 00 20 00 00 00
HKU\S-1-5-21-321011808-3761883066-353627080-1000\Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs\69: 53 00 61 00 6D 00 70 00 6C 00 65 00 5F 00 53 00 68
HKU\S-1-5-21-321011808-3761883066-353627080-1000\Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs\hiv\4: 53 00 61 00 6D 00 70 00 6C 00 65 00 5F 00 53 00 68
HKU\S-1-5-21-321011808-3761883066-353627080-1000\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\{CEBFF5CD-ACE2-4F4F-9178-9926F41749E4}\Count\p:\VHfreef\l\l
HKU\S-1-5-21-321011808-3761883066-353627080-1000\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\{CEBFF5CD-ACE2-4F4F-9178-9926F41749E4}\Count\p:\VHfreef\l\l
HKU\S-1-5-21-321011808-3761883066-353627080-1000\Software\Microsoft\Windows\CurrentVersion\Explorer\SessionInfo\1\ApplicationViewManagement\W32:000000000002C0918\Virtu:
HKU\S-1-5-21-321011808-3761883066-353627080-1000\Software\Microsoft\Windows\CurrentVersion\Explorer\SessionInfo\1\ApplicationViewManagement\W32:0000000000350342\Virtu:
HKU\S-1-5-21-321011808-3761883066-353627080-1000\Software\Microsoft\Windows\CurrentVersion\Explorer\SessionInfo\1\ApplicationViewManagement\W32:00000000003607E4\Virtu:

```

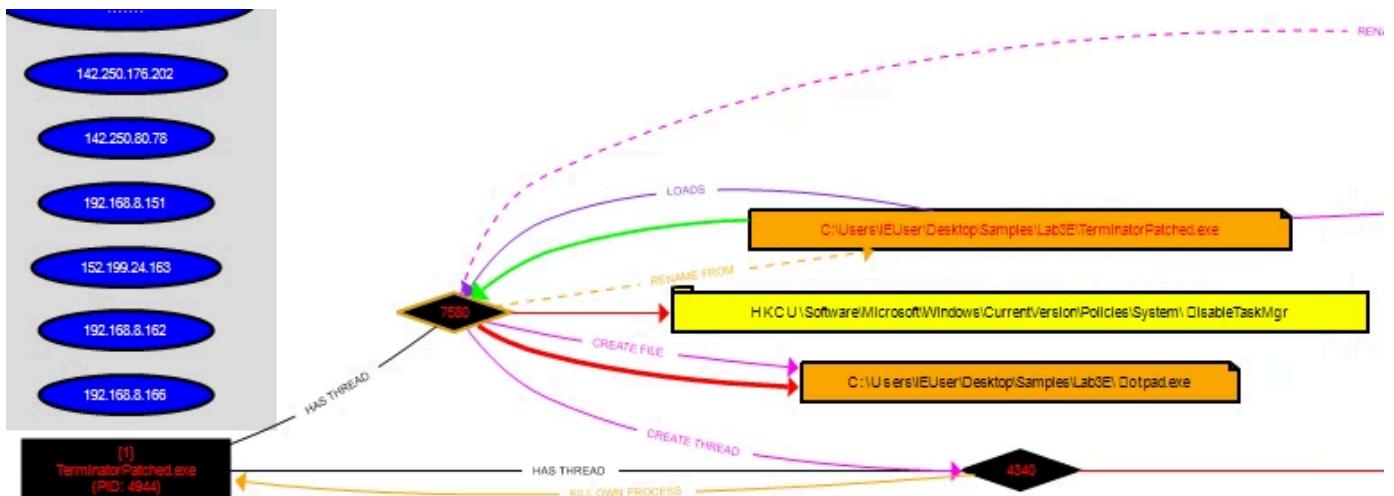
We will come back to this screenshot later when looking at the files.

## ProcDot

ProcDot is a great tool for showing how samples work behind the scenes by connecting their processes, registry activity, and more from the previously talked about tools into a visual graph and timeline



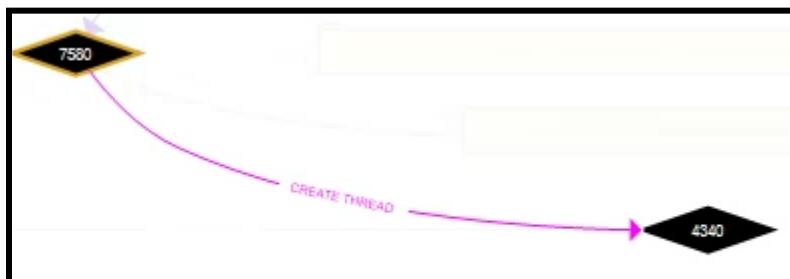
This screenshot shows the whole flow of the sample. It is hard to see so it is broken down below.



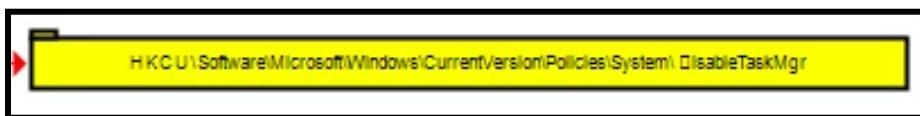
This screenshot shows many of the steps the sample took. We will look at what it does step by step next.



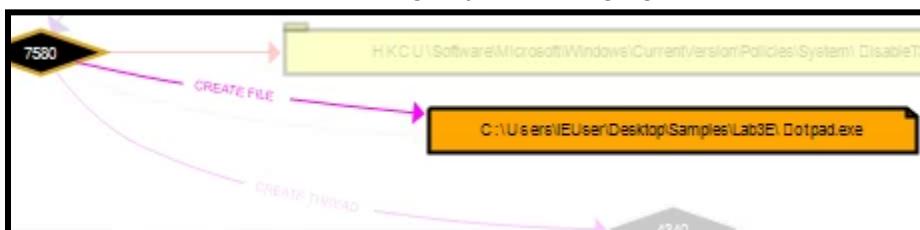
First the sample TerminatorPatched.exe loads itself into thread 7580.



The exe then creates a new thread 4340



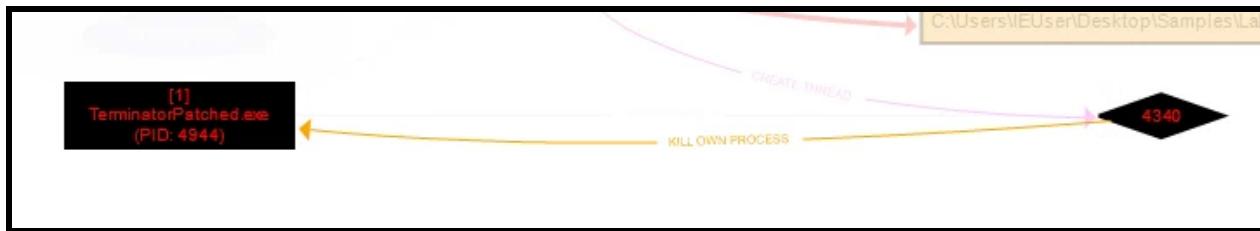
Once the new thread is created, it then sets the registry key to DisableTaskMgr. However, because we patched the sample, it searches for ~\HKCU\<path>\isableTaskManager. Without the first letter. This results in the reg key not changing as it cannot find the file.



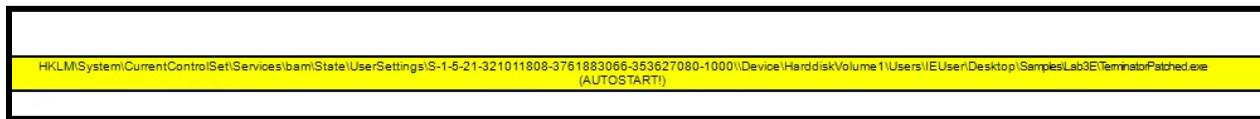
After it sets the registry key, it creates a file called notepad.exe. This is because we removed the first letter of notepad.exe when patching. The thread then reads data from the patched exe and places it in the new file it created.



The sample then renames the file terminatorPatched.exe to aint.exe. This is similar to the new file above, but because we nulled out the first letter when patching it doesn't say paint.exe.

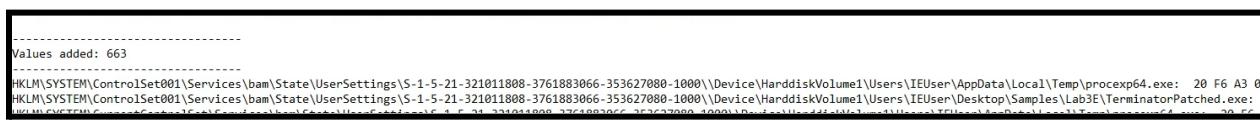


After reading a bit more information from the TerminatorPatched.exe file, it then kills its own process.



The new thread it created earlier then sets an auto start registry key under:  
`HKLM\System\CurrentControlSet\Services\bam\State\UserSettings\S-1-5-21-321011808-3761883066-353627080-1000\Device\HarddiskVolume1\Users\EUser\Desktop\Samples\Lab3E\TerminatorPatched.exe`  
 This allows TerminatorPatched.exe to start

This is a perfect time to look at our regshot registry image data. We can see in the screenshot below the registry key mentioned above and that it had a value added.



## Deliverable 3 - Function Descriptions

Check at least 5 imported functions on [MSDN](#) that are important to the behavior of this given sample and explain what they are used for. Use the table below to add your response. An example is listed below.

API (function)	DLL	Explanation	Reference
RegSetValueExA	advapi32.dll	This function is used to set the data and type of the specified value in / under a registry key. This function has a few parameters: hkey, valuename, reserved, type, and data. The only one that is not optional is the value name.	<a href="#">Link</a>
RegCreateKeyExA	advapi32.dll	This function is used to create a specified registry key. If the key already exists, the key is then opened instead. This function has 9 different	<a href="#">Link</a>

		parameters. Hkey, subkey, reserved, class, options, desired sam, security attributes, result, and disposition.	
ICMPSendEcho	iphlpapi.dll	This function sends out an ICMP IPv4 echo request and will return any response replies. This function takes in 8 parameters ICMP handle (the open handle that is returned), Destination address, request data (a pointer to a buffer that contains data to send in the request), request size (the size in bytes of the request data buffer pointed to), reply buffer, reply size and timeout.	<a href="#">Link</a>
ICMPCreateFile	iphlpapi.dll	This function opens a handle on which ICMP IPv4 echo requests can be issued. On success, it returns an open handle on success. On failure, the function returns INVALID_HANDLE_VALUE.	<a href="#">Link</a>
WriteFileA	kernal32.dll	This functions writes data to the specified file or input/output device. It is designed for both synchronous and asynchronous operation. It has a few parameters hfile handle, lpbuffer, number of bytes to write, number of bytes written, and overlapped. The first three are the only required ones.	<a href="#">Link</a>

## Deliverable 4 - Reflection

This lab for some reason took me the longest of any lab so far. I decided to focus a bit more on my formatting and explanations. I had a lot of fun patching the sample and decided to challenge myself and patch the mouse button swapping. Hopefully I went into enough detail about the different artifacts and tools. Overall, fun lab!