Review of Windows 7 as a malware analysis environment

GIAC (GREM) Gold Certification

Author: Adam Kramer, adamkramer@hotmail.com
Advisor: Mark Stingley

Accepted: January 2nd, 2014

Abstract
The SANS course "FOR610 – Reverse Engineering of Malware" is designed around the use of Windows XP as a malware analysis environment. Mainstream support for this operating system ended over 4 years ago and for those with extended support contracts, these are due to end April 2014. This will prompt both enterprises and home users to consider the transition to a newer version of Windows which has ongoing support and regular updates. As users transition, the malware and therefore analyst will need to follow. In order to be ahead of the game and ready for the coming changes we will look at how each of many of the main tools used by malware analysis perform under Windows 7 x64 detailing any issues encountered and giving consideration to possible solutions.

1. Introduction

The SANS course "FOR610: Reverse Engineering of Malware" is designed using Windows XP as the malware analysis environment (SANS Institute, 2013). According to the Microsoft support lifecycle, their mainstream support for Windows XP ended over four years ago and for those with extended support contracts, they are due to end in April 2014 (Microsoft, 2013). In order to have access to software updates and maintain security, enterprises and consumers alike will be required to update to a newer version. As users upgrade to a new operating system, malware authors and therefore the malware analyst will need to follow. The cautious corporate administrator has the option of upgrading to either the latest version "Windows 8" or the more mature "Windows 7". The online publication techrepublic.com has run a survey every 6 months since 2009 to try and identify the trend in operating system migration away from Windows XP and what people are choosing to move to. The indication in March 2013 was that companies had progressively been moving to "Windows 7" and 85% either had no plans or had decided against moving to "Windows 8" (Tech Republic, 2013).

In this review, the tools and techniques taught in FOR610 will be tested under Windows 7 x64 to establish on a tool by tool basis whether they are fully functional, can be made to work with modifications to the system or whether they will not work at all and in which case what alternatives are available.

2. Test environment

2.1 Design of tool testing program

A benign program called "Harmless_Malware.exe" was written and compiled in C++ to emulate the Windows system API calls and functions found in common malware. Access to the source code allows anyone conducting the tests to verify what the results should be (available as Appendix A).

This program has the following functionality:

1. The process will identify the path it was executed from and add itself to the "Run" key in the registry in order that it be automatically started when the system loads.

2. The process will attempt to write the EICAR standard antivirus test string (EICAR, 2003) to a file ("avtest.txt"). This will emulate the functionality of a "dropper".

3. The process will attempt to open an internet connection to www.sans.org and retrieve the homepage.

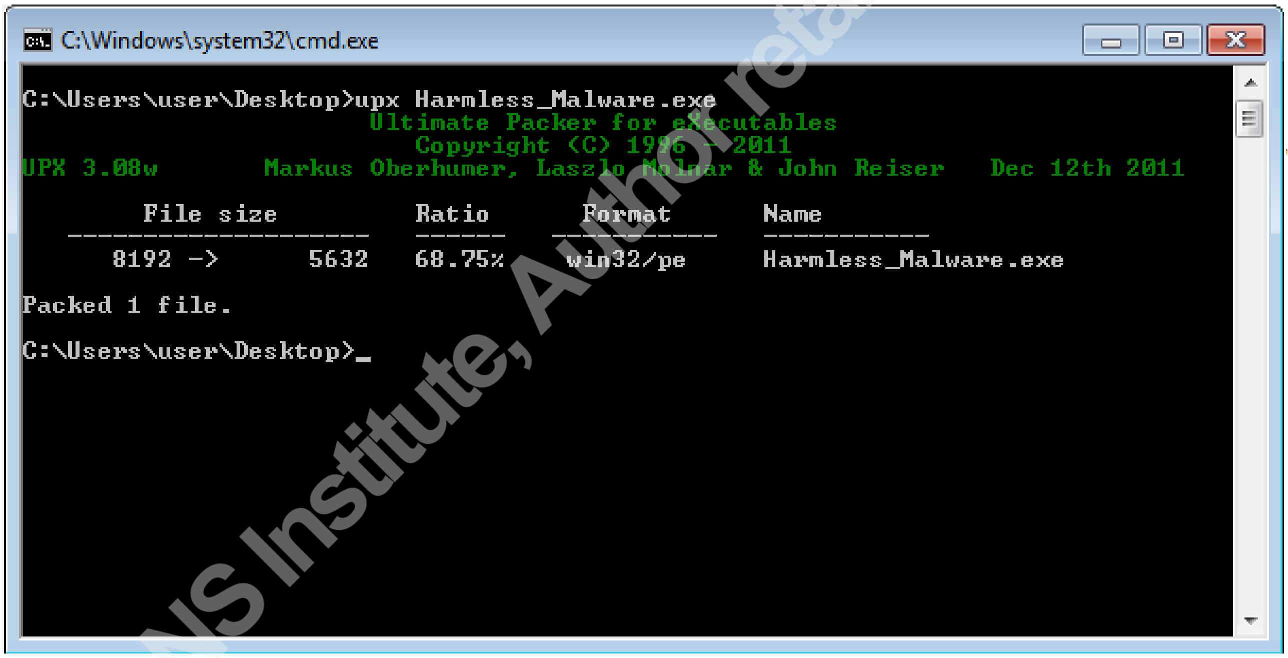
These three tasks will cover persistence, file system modification and network communication which are three tasks common to malware functionality.

The test program was created with the support of the MSDN online Windows API support pages and is available freely for anyone wishing to conduct similar testing (MSDN, 2013).

Adam Kramer, adamkramer@hotmail.com

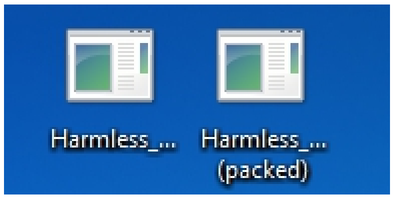
Consideration was given to the architecture in which to compile this code to. It was decided that as x86 programs can be run on both x86 and x64 platforms that this would be the favored choice of the malware author as their product would work on a higher proportion of systems. A x86 program run on x64 Windows uses the WoW64 system which supports the backwards compatibility of architecture execution (Microsoft - Dev Center (A), 2013).

Following the compilation of the code into **"Harmless_Malware.exe"** UPX packer (UPX, 2013) was used to create a packed version of the program which was named **"Harmless_Malware (packed).exe"**. This would become useful when testing tools and techniques which work on packed executables.



```
C:\Users\user\Desktop>upx Harmless_Malware.exe
                       Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2011
UPX 3.08w       Markus Oberhumer, Laszlo Molnar & John Reiser   Dec 12th 2011

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
      8192 ->      5632   68.75%    win32/pe     Harmless_Malware.exe

Packed 1 file.

C:\Users\user\Desktop>_
```

The resultant executables were then ready for the testing process to begin.



Adam Kramer, adamkramer@hotmail.com

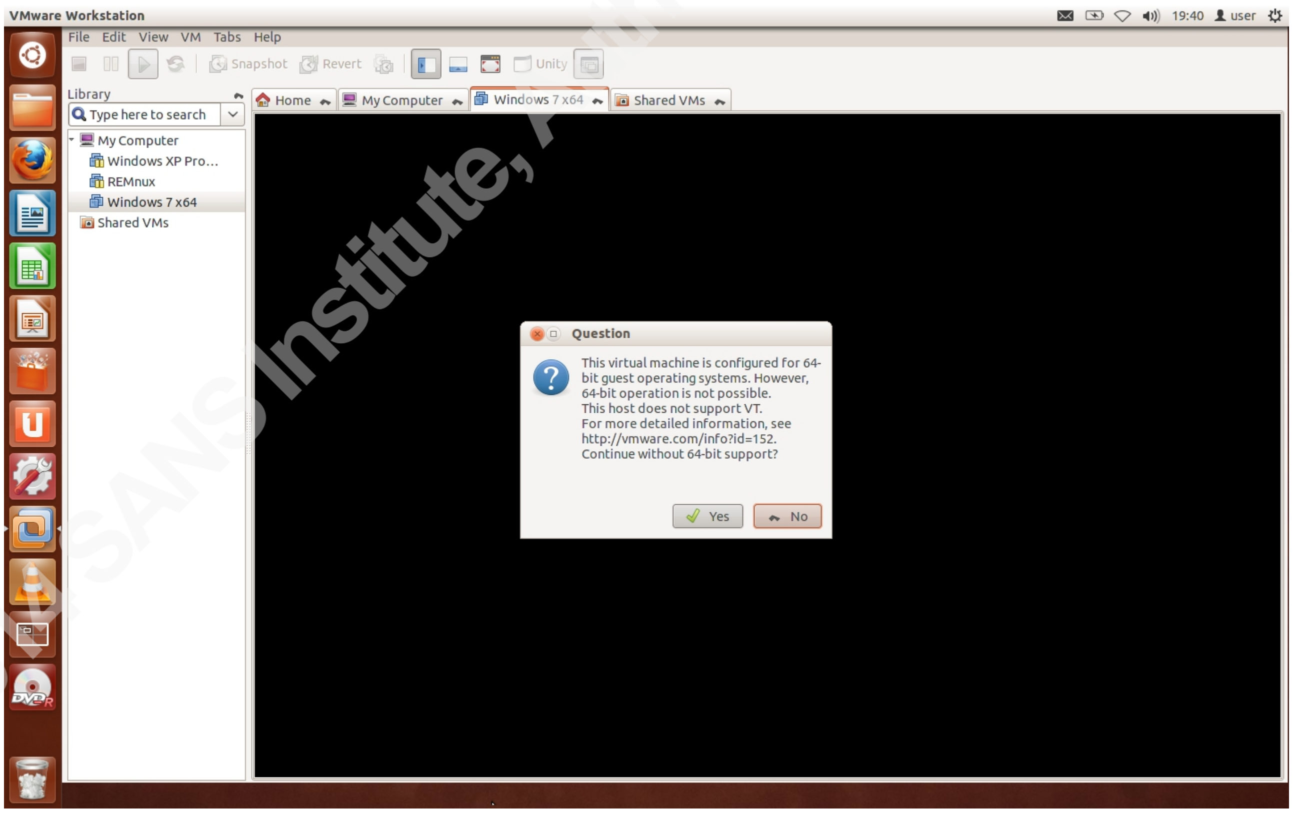
2.2 Setup of the Windows 7 x64 environment

At this point the setup of the Windows 7 x64 environment was started. Initially VMware was used to install the operating system to as this was the technique used in the FOR610 course and would allow quick roll back and the ability to save system states to aid with testing.

Unfortunately, very quickly an issue arose with the installation of a x64 operating system on VMware.

A dialog box appeared as soon as the operating system installer was launched with the following message:

“This virtual machine is configured for 64-bit guest operating systems. However, 64-bit operation is not possible. This host does not support VT. For more detailed information, see http://vmware.com/info?id=152. Continue without 64-bit support?

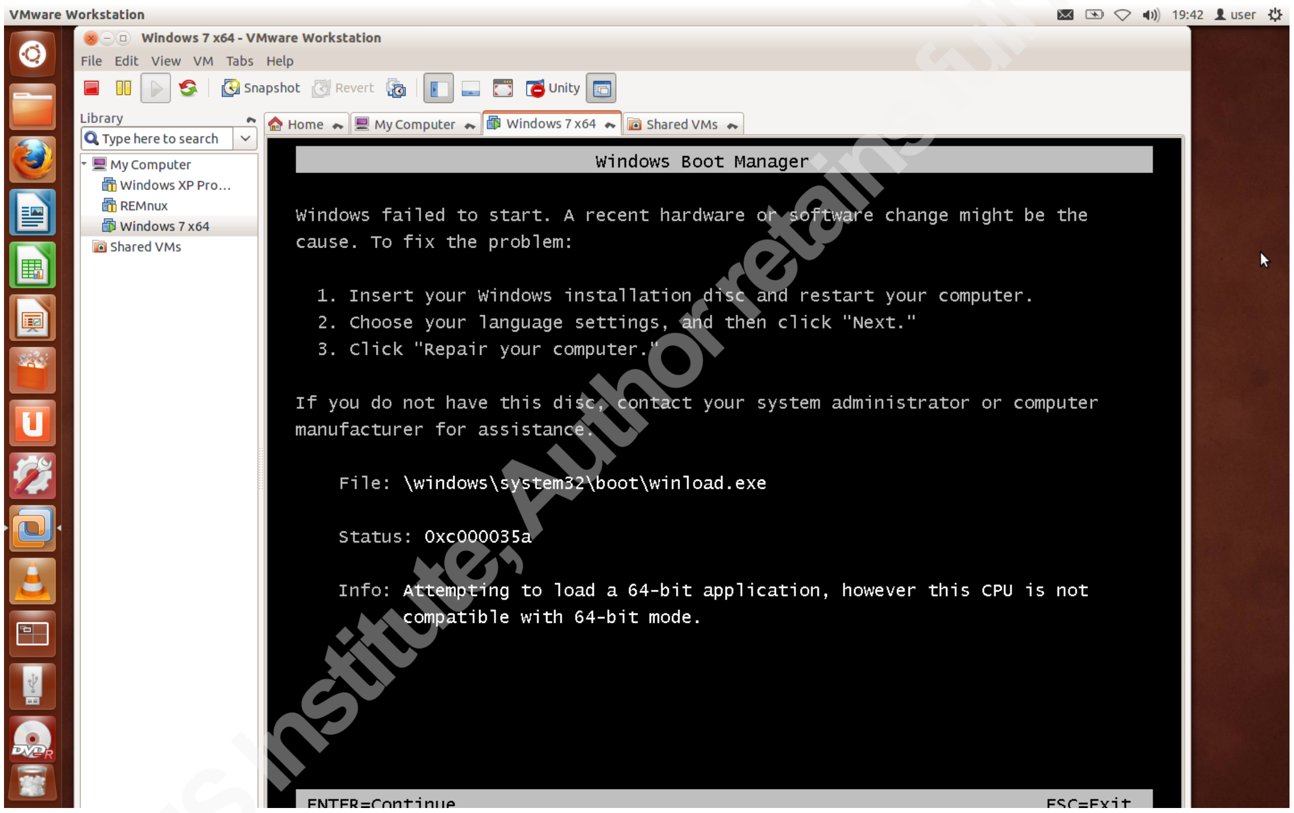


This virtual machine is configured for 64-bit guest operating systems. However, 64-bit operation is not possible.
This host does not support VT.
For more detailed information, see http://vmware.com/info?id=152.
Continue without 64-bit support?

On proceeding a message was presented which was part of the Windows 7 x64 installation process and gave the error message shown in the screenshot below.

As follows:

"Info: Attempting to load a 64-bit application, however this CPU is not compatible with 64-bit mode"



```
Windows Boot Manager

Windows failed to start. A recent hardware or software change might be the
cause. To fix the problem:

  1. Insert your Windows installation disc and restart your computer.
  2. Choose your language settings, and then click "Next."
  3. Click "Repair your computer."

If you do not have this disc, contact your system administrator or computer
manufacturer for assistance.

    File: \windows\system32\boot\winload.exe

    Status: 0xc000035a

    Info: Attempting to load a 64-bit application, however this CPU is not
          compatible with 64-bit mode.

ENTER=Continue                                                      ESC=Exit
```

It appears that the hardware of the machine which was being used to do this testing was causing issues while loading the "winload.exe" program which formed part of the installation process. The advice provided by VMware was reviewed but it was not possible to get it working. At this point there was a choice of changing to a different machine or installing the operating system directly rather than using a virtual environment.

At this point a partition was created and Windows 7 was installed directly to the machine to prevent any further VM related issues.

Adam Kramer, adamkramer@hotmail.com

3. Individual tool testing

A selection of the core tools from the malware analysts toolkit as taught in FOR610 were tested using the benign sample and their compatibility with "Windows 7" is summarized below and detailed in the following pages.

Tool	Functionality	Identified issues
BinText	Fully functional	None
Process Monitor	Fully functional	None
Process Hacker	Fully functional	None
Autoruns	Fully functional	None
PEiD	Fully functional	None
Capture BAT	Partially functional	Network traffic only
Regshot	Fully functional	None
Lord PE	Appears functional *	Possible issue with automated header repair
CHimpREC	Testing failed	Loading & functionality issues – see report
Ollydbg v2	Appears functional *	None
IDA Pro	Appears functional *	None

Please note:

Where a tool is marked Appears functional * this is a recognition of the complex nature of code analysis, including the unreliable "trial and error" nature of process dumping & header repair in the analysis of a packed executable.

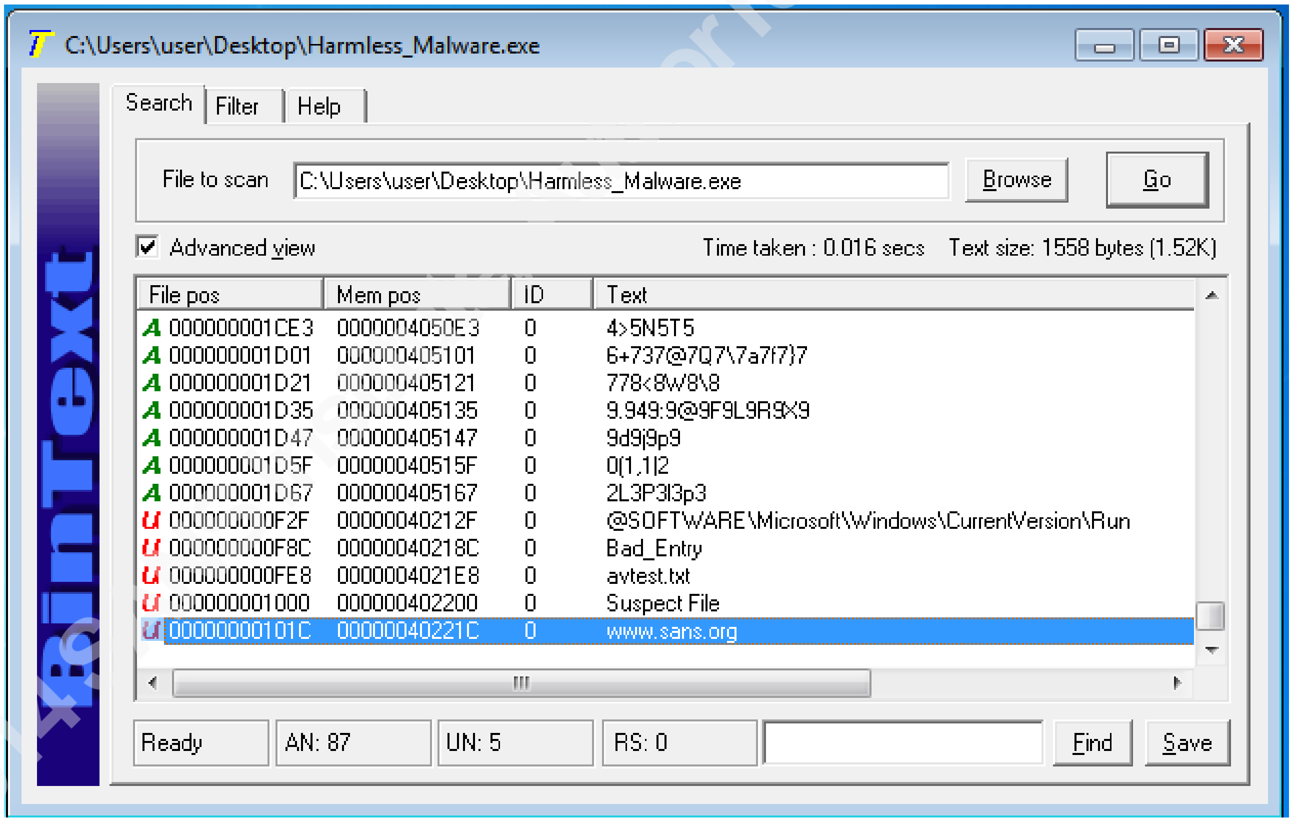
The individual report pages detail the testing that was completed and results achieved.

3.1 BinText – Fully functional

(BinText, 2013)

This tool was run against the unpacked executable and as can be seen in the screenshot below was able to identify various strings from static variables that we would expect to see including:

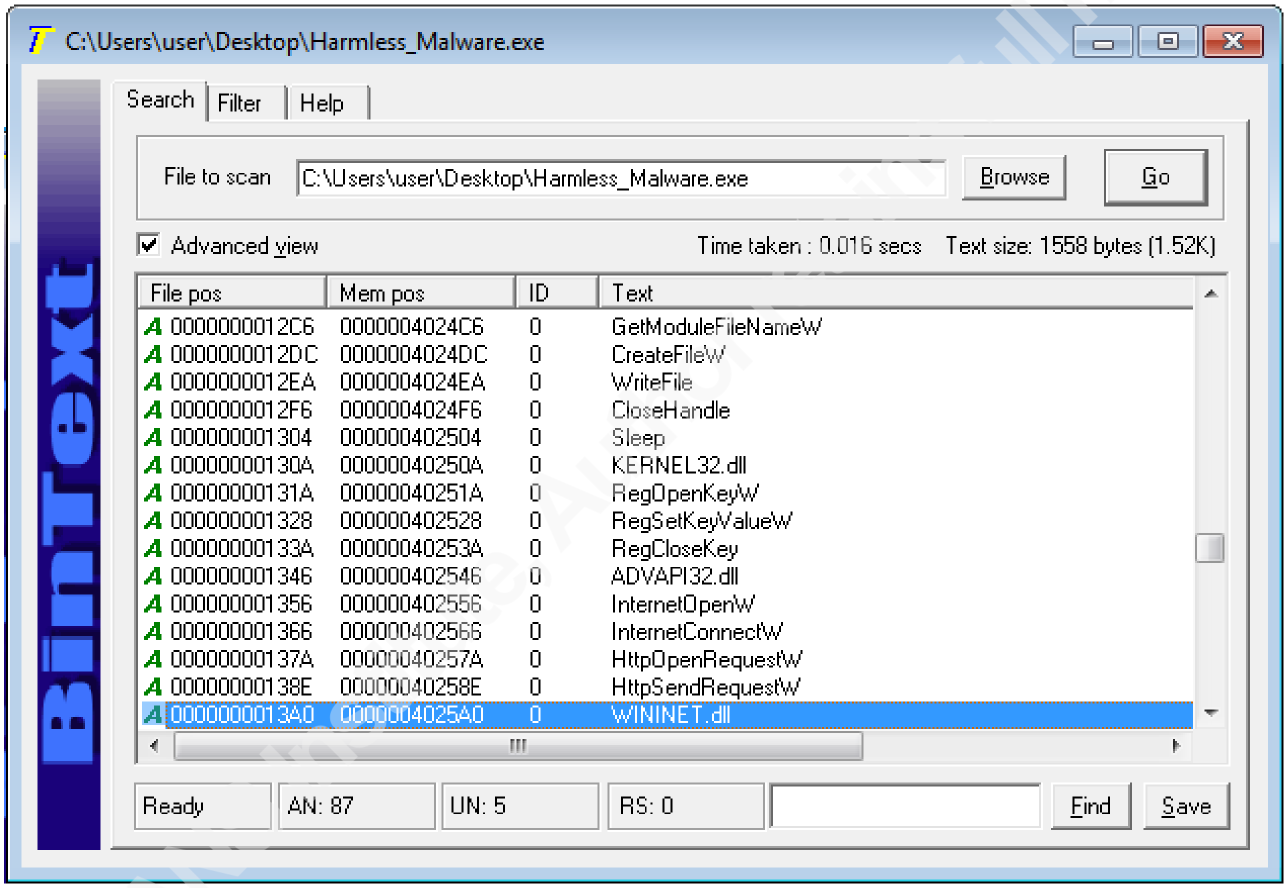
- SOFTWARE\Microsoft\Windows\CurrentVersion\Run
- Bad_Entry
- avtest.txt
- Suspect File
- www.sans.org


C:\Users\user\Desktop\Harmless_Malware.exe
BinText
Search
Filter
Help
File to scan
C:\Users\user\Desktop\Harmless_Malware.exe
Browse
Go
Advanced view
Time taken : 0.016 secs
Text size: 1558 bytes (1.52K)
Ready
AN: 87
UN: 5
RS: 0
Find
Save

```
   File pos          Mem pos           ID   Text
A  000000001CE3      0000004050E3      0    4>5N5T5
A  000000001D01      000000405101      0    6+737@7Q7\7a7f7}7
A  000000001D21      000000405121      0    778<8W8\8
A  000000001D35      000000405135      0    9.949:9@9F9L9R9X9
A  000000001D47      000000405147      0    9d9j9p9
A  000000001D5F      00000040515F      0    0(1,1|2
A  000000001D67      000000405167      0    2L3P3l3p3
U  000000000F2F      00000040212F      0    @SOFTWARE\Microsoft\Windows\CurrentVersion\Run
U  000000000F8C      00000040218C      0    Bad_Entry
U  000000000FE8      0000004021E8      0    avtest.txt
U  000000001000      000000402200      0    Suspect File
U  00000000101C      00000040221C      0    www.sans.org
```

Adam Kramer, adamkramer@hotmail.com

Further results from the same executable reveal API calls which were used in the program. We can see the calls made, in fact everything listed in the screenshot below would have been called or linked in to the program somehow. If we were using this tool in the primary stages of an analysis we would be able to identify some of the functionality of the program without much work.


C:\Users\user\Desktop\Harmless_Malware.exe
BinText
Search
Filter
Help
File to scan
C:\Users\user\Desktop\Harmless_Malware.exe
Browse
Go
Advanced view

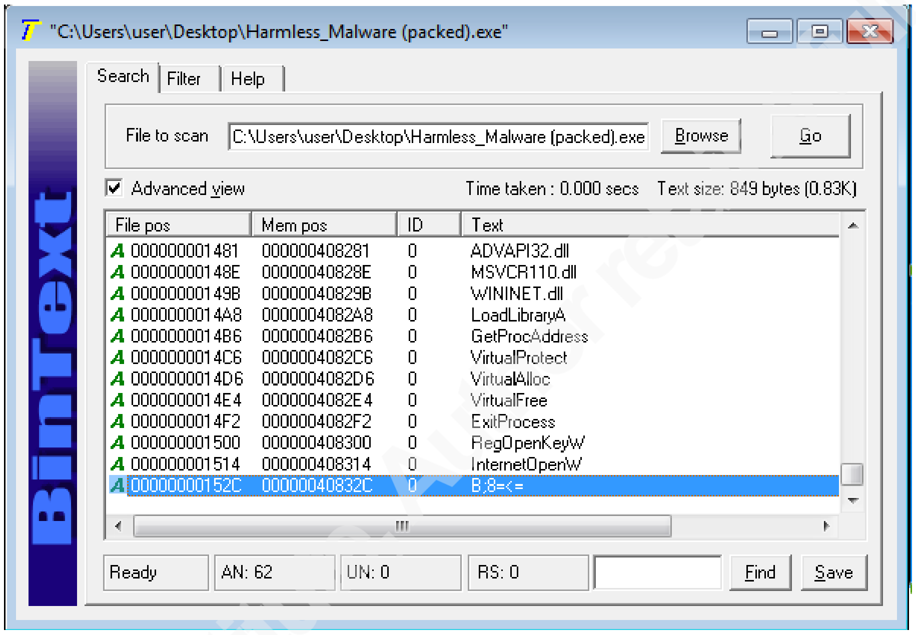
Time taken : 0.016 secs Text size: 1558 bytes (1.52K)

File pos	Mem pos	ID	Text
A 0000000012C6	0000004024C6	0	GetModuleFileNameW
A 0000000012DC	0000004024DC	0	CreateFileW
A 0000000012EA	0000004024EA	0	WriteFile
A 0000000012F6	0000004024F6	0	CloseHandle
A 000000001304	000000402504	0	Sleep
A 00000000130A	00000040250A	0	KERNEL32.dll
A 00000000131A	00000040251A	0	RegOpenKeyW
A 000000001328	000000402528	0	RegSetKeyValueW
A 00000000133A	00000040253A	0	RegCloseKey
A 000000001346	000000402546	0	ADVAPI32.dll
A 000000001356	000000402556	0	InternetOpenW
A 000000001366	000000402566	0	InternetConnectW
A 00000000137A	00000040257A	0	HttpOpenRequestW
A 00000000138E	00000040258E	0	HttpSendRequestW
A 0000000013A0	0000004025A0	0	WININET.dll

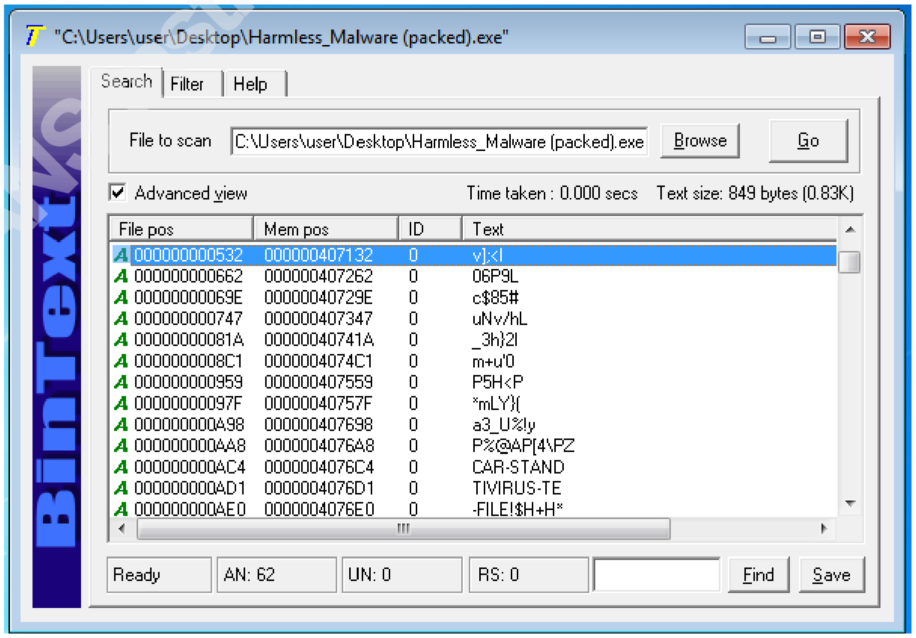
Ready AN: 87 UN: 5 RS: 0

Now we can move on to the packed executable and as we can see from the two screenshots below there are some indications of the functionality of the packed program (in fact there are more clear text API calls than would typically be expected to seen from a packed executable).

Either way the tool was performing as expected and outputting correctly.


"C:\Users\user\Desktop\Harmless_Malware (packed).exe"
Search
Filter
Help
File to scan
C:\Users\user\Desktop\Harmless_Malware (packed).exe
Browse
Go
Advanced view
BinText

```
Time taken : 0.000 secs    Text size: 849 bytes (0.83K)

File pos        Mem pos         ID  Text
A 00000000148 1  000000408281    0   ADVAPI32.dll
A 00000000148E   00000040828E    0   MSVCR110.dll
A 00000000149B   00000040829B    0   WININET.dll
A 0000000014A8   0000004082A8    0   LoadLibraryA
A 0000000014B6   0000004082B6    0   GetProcAddress
A 0000000014C6   0000004082C6    0   VirtualProtect
A 0000000014D6   0000004082D6    0   VirtualAlloc
A 0000000014E4   0000004082E4    0   VirtualFree
A 0000000014F2   0000004082F2    0   ExitProcess
A 000000001500   000000408300    0   RegOpenKeyW
A 000000001514   000000408314    0   InternetOpenW
A 00000000152C   00000040832C    0   B;8=<=

Ready   AN: 62   UN: 0   RS: 0
```


"C:\Users\user\Desktop\Harmless_Malware (packed).exe"
Search
Filter
Help
File to scan
C:\Users\user\Desktop\Harmless_Malware (packed).exe
Browse
Go
Advanced view
BinText

```
Time taken : 0.000 secs    Text size: 849 bytes (0.83K)

File pos        Mem pos         ID  Text
A 000000000532   000000407132    0   v];<I
A 000000000662   000000407262    0   06P9L
A 00000000069E   00000040729E    0   c$85#
A 000000000747   000000407347    0   uNv/hL
A 00000000081A   00000040741A    0   _3h}2I
A 0000000008C1   0000004074C1    0   m+u'0
A 000000000959   000000407559    0   P5H<P
A 00000000097F   00000040757F    0   *mLY}(
A 000000000A98   000000407698    0   a3_U%!y
A 000000000AA8   0000004076A8    0   P%@AP[4\PZ
A 000000000AC4   0000004076C4    0   CAR-STAND
A 000000000AD1   0000004076D1    0   TIVIRUS-TE
A 000000000AE0   0000004076E0    0   -FILE!$H+H*

Ready   AN: 62   UN: 0   RS: 0
```

3.2 Process Monitor – Fully functional

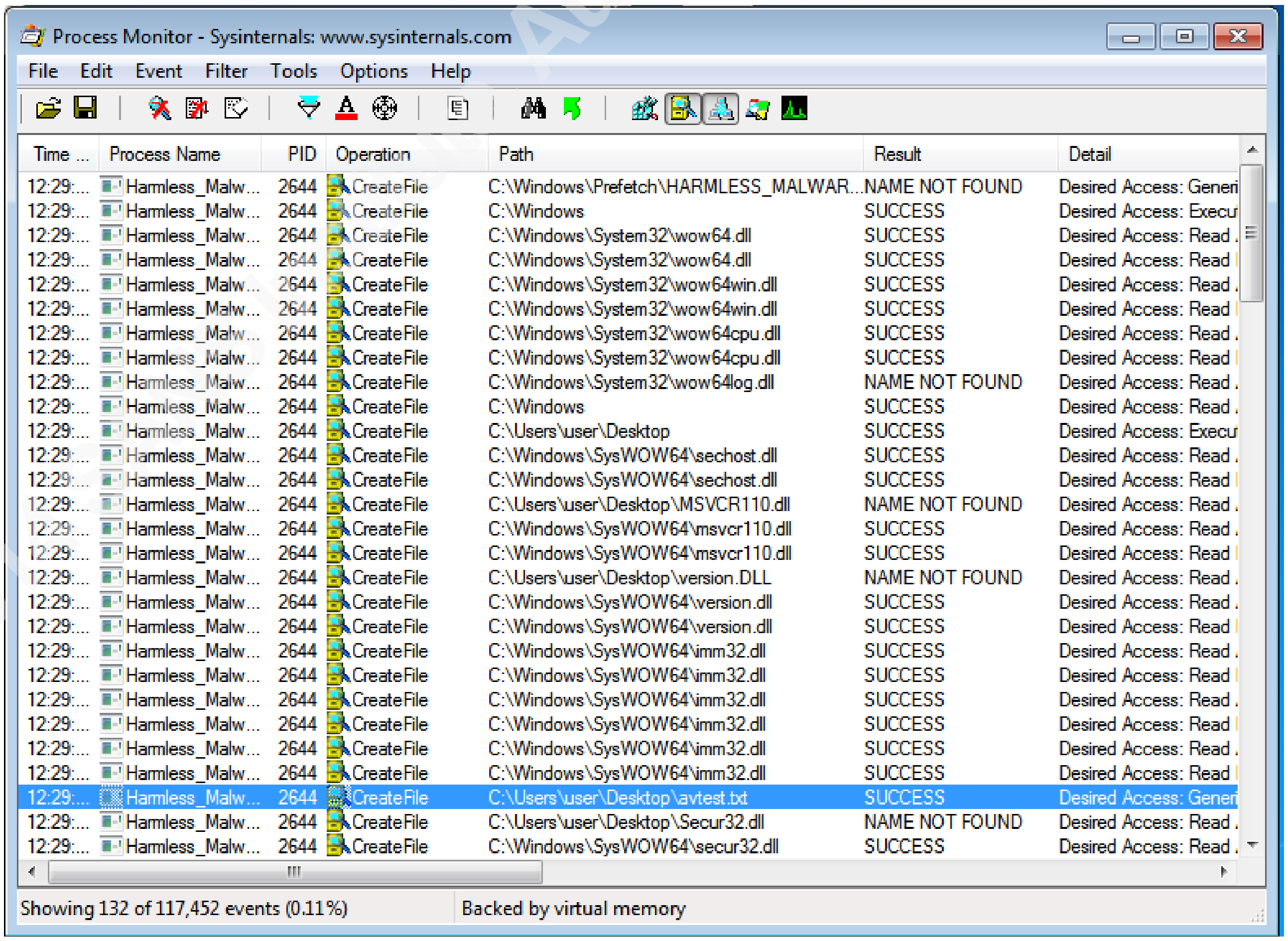
(Process Monitor, 2013)

Process monitor is designed to capture all the Windows API functions that have been called and log them in an interface that can be filtered and saved.

The screenshot below shows a small segment of the output which resulted from filtering to only show the CreateFile operations from our process.

As ever, there are vast numbers of Windows API calls which are being done in the background, but of note a CreateFile operation was highlighted which is linking in with the avtest.txt file.

As well as correctly capturing this call, Process Monitor managed to capture all the relevant calls made during the program including the full details and results.


Process Monitor - Sysinternals: www.sysinternals.com
File Edit Event Filter Tools Options Help

Time ...	Process Name	PID	Operation	Path	Result	Detail
12:29:...	Harmless_Malw...	2644	CreateFile	C:\Windows\Prefetch\HARMLESS_MALWAR...	NAME NOT FOUND	Desired Access: Generi
12:29:...	Harmless_Malw...	2644	CreateFile	C:\Windows	SUCCESS	Desired Access: Execu
12:29:...	Harmless_Malw...	2644	CreateFile	C:\Windows\System32\wow64.dll	SUCCESS	Desired Access: Read .
12:29:...	Harmless_Malw...	2644	CreateFile	C:\Windows\System32\wow64.dll	SUCCESS	Desired Access: Read I
12:29:...	Harmless_Malw...	2644	CreateFile	C:\Windows\System32\wow64win.dll	SUCCESS	Desired Access: Read .
12:29:...	Harmless_Malw...	2644	CreateFile	C:\Windows\System32\wow64win.dll	SUCCESS	Desired Access: Read I
12:29:...	Harmless_Malw...	2644	CreateFile	C:\Windows\System32\wow64cpu.dll	SUCCESS	Desired Access: Read .
12:29:...	Harmless_Malw...	2644	CreateFile	C:\Windows\System32\wow64cpu.dll	SUCCESS	Desired Access: Read I
12:29:...	Harmless_Malw...	2644	CreateFile	C:\Windows\System32\wow64log.dll	NAME NOT FOUND	Desired Access: Read .
12:29:...	Harmless_Malw...	2644	CreateFile	C:\Windows	SUCCESS	Desired Access: Read .
12:29:...	Harmless_Malw...	2644	CreateFile	C:\Users\user\Desktop	SUCCESS	Desired Access: Execu
12:29:...	Harmless_Malw...	2644	CreateFile	C:\Windows\SysWOW64\sechost.dll	SUCCESS	Desired Access: Read .
12:29:...	Harmless_Malw...	2644	CreateFile	C:\Windows\SysWOW64\sechost.dll	SUCCESS	Desired Access: Read I
12:29:...	Harmless_Malw...	2644	CreateFile	C:\Users\user\Desktop\MSVCR110.dll	NAME NOT FOUND	Desired Access: Read .
12:29:...	Harmless_Malw...	2644	CreateFile	C:\Windows\SysWOW64\msvcr110.dll	SUCCESS	Desired Access: Read .
12:29:...	Harmless_Malw...	2644	CreateFile	C:\Windows\SysWOW64\msvcr110.dll	SUCCESS	Desired Access: Read I
12:29:...	Harmless_Malw...	2644	CreateFile	C:\Users\user\Desktop\version.DLL	NAME NOT FOUND	Desired Access: Read .
12:29:...	Harmless_Malw...	2644	CreateFile	C:\Windows\SysWOW64\version.dll	SUCCESS	Desired Access: Read .
12:29:...	Harmless_Malw...	2644	CreateFile	C:\Windows\SysWOW64\version.dll	SUCCESS	Desired Access: Read I
12:29:...	Harmless_Malw...	2644	CreateFile	C:\Windows\SysWOW64\imm32.dll	SUCCESS	Desired Access: Read .
12:29:...	Harmless_Malw...	2644	CreateFile	C:\Windows\SysWOW64\imm32.dll	SUCCESS	Desired Access: Read I
12:29:...	Harmless_Malw...	2644	CreateFile	C:\Windows\SysWOW64\imm32.dll	SUCCESS	Desired Access: Read .
12:29:...	Harmless_Malw...	2644	CreateFile	C:\Windows\SysWOW64\imm32.dll	SUCCESS	Desired Access: Read I
12:29:...	Harmless_Malw...	2644	CreateFile	C:\Windows\SysWOW64\imm32.dll	SUCCESS	Desired Access: Read .
12:29:...	Harmless_Malw...	2644	CreateFile	C:\Windows\SysWOW64\imm32.dll	SUCCESS	Desired Access: Read I
12:29:...	Harmless_Malw...	2644	CreateFile	C:\Users\user\Desktop\avtest.txt	SUCCESS	Desired Access: Generi
12:29:...	Harmless_Malw...	2644	CreateFile	C:\Users\user\Desktop\Secur32.dll	NAME NOT FOUND	Desired Access: Read .
12:29:...	Harmless_Malw...	2644	CreateFile	C:\Windows\SysWOW64\secur32.dll	SUCCESS	Desired Access: Read .

Showing 132 of 117,452 events (0.11%) | Backed by virtual memory

Adam Kramer, adamkramer@hotmail.com

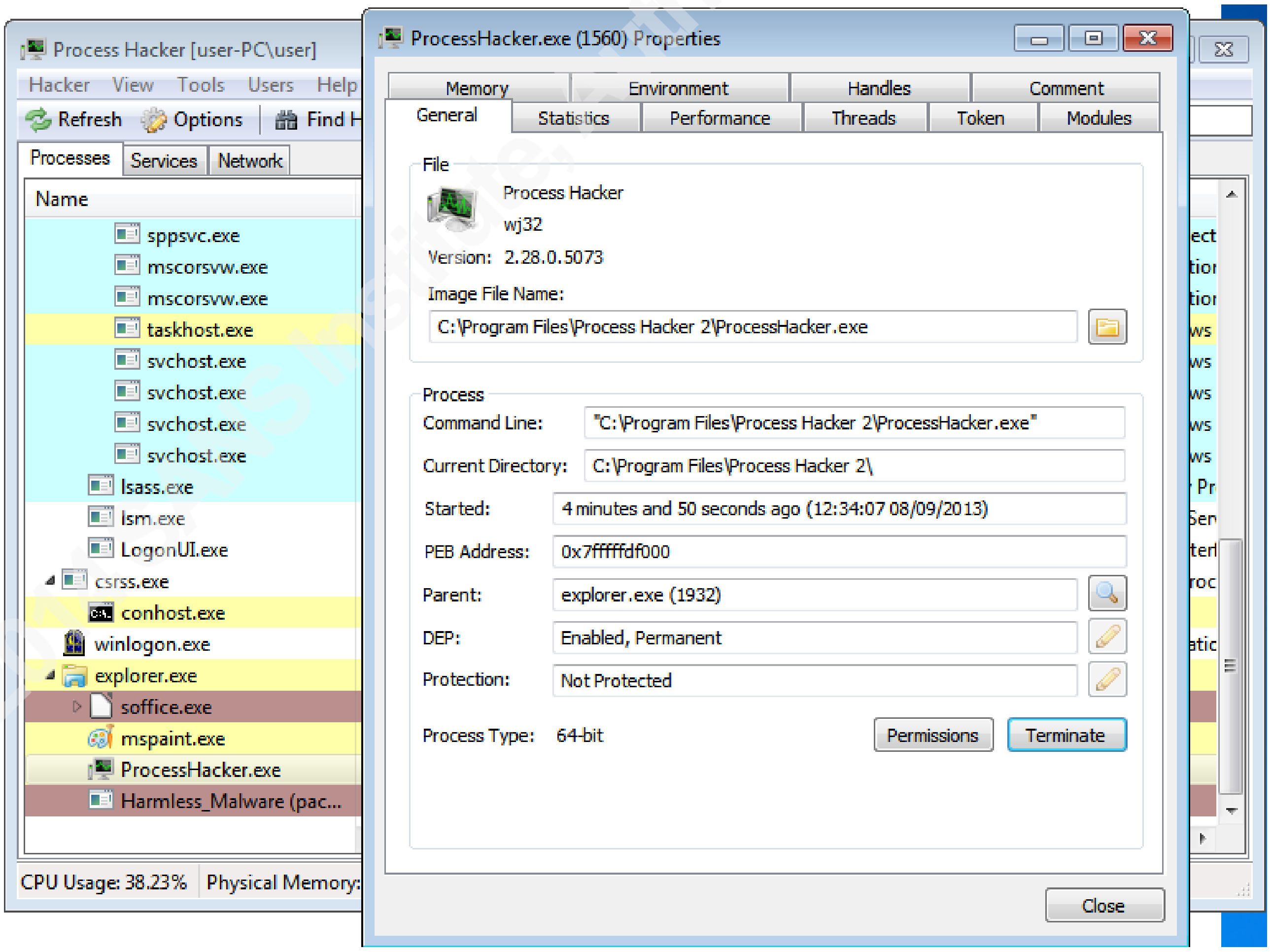
3.3 Process Hacker – Fully functional

(Process Hacker, 2013)

Process Hacker is a significantly more powerful task manager than the one built into Windows. It is able to correctly identify and detail both x86 and x64 bit processes and the tests into the functionality and detailing came back showing that it was operating as would be expected.

As shown in the screenshot below, our process is listed in the background. The properties window has been opened for process hacker itself and as we can see it is able to successfully show the overview details about the process.

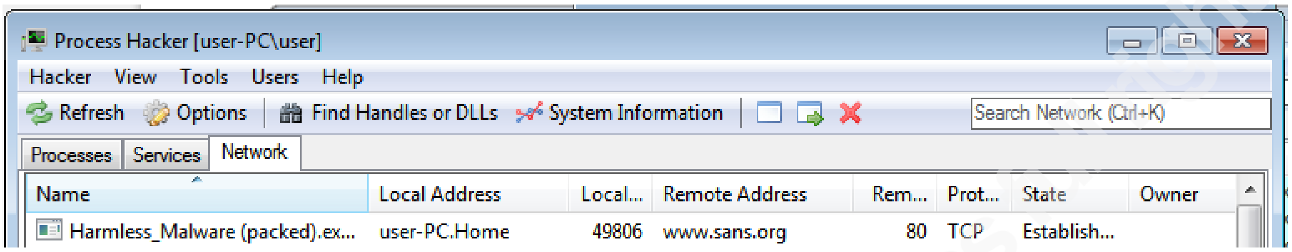
The powerful functionality such as the ability to terminate a process directly was tested and was also a success.



```
Process Hacker
wj32
Version:            2.28.0.5073
Image File Name:    C:\Program Files\Process Hacker 2\ProcessHacker.exe

Command Line:       "C:\Program Files\Process Hacker 2\ProcessHacker.exe"
Current Directory:  C:\Program Files\Process Hacker 2\
Started:            4 minutes and 50 seconds ago (12:34:07 08/09/2013)
PEB Address:        0x7fffffdf000
Parent:             explorer.exe (1932)
DEP:                Enabled, Permanent
Protection:         Not Protected
Process Type:       64-bit
```

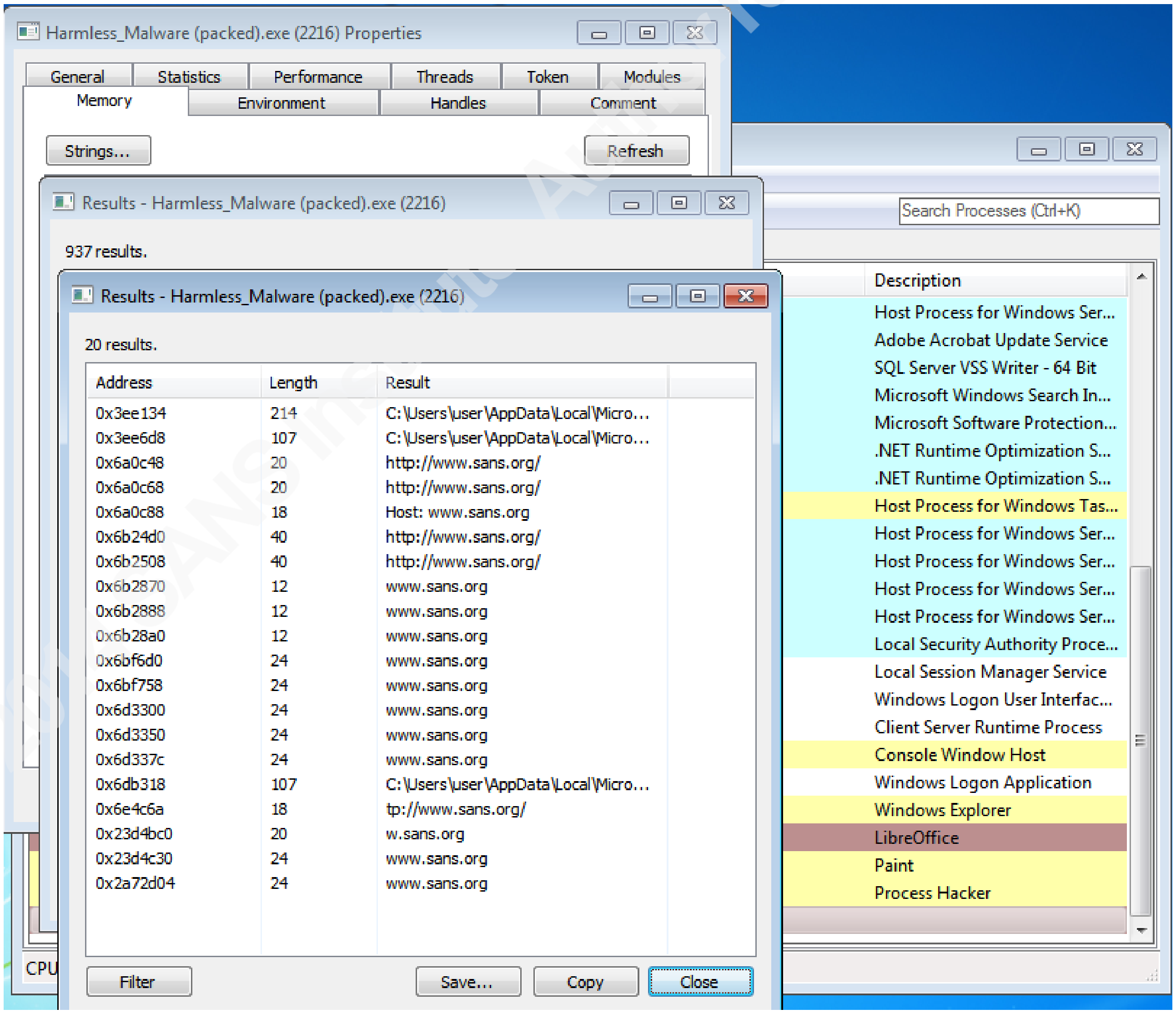
Adam Kramer, adamkramer@hotmail.com

Below is a screenshot of the network tab which show that it was able to successfully identify and display the connection that our program has made to www.sans.org and details the ports and current state.


Process Hacker [user-PC\user]
Hacker View Tools Users Help
Refresh Options Find Handles or DLLs System Information
Search Network (Ctrl+K)
Processes Services Network

Name	Local Address	Local...	Remote Address	Rem...	Prot...	State	Owner
Harmless_Malware (packed).ex...	user-PC.Home	49806	www.sans.org	80	TCP	Establish...	

Again, further functionality has been tested showing that it is able to successfully access the memory and extract strings on request.


Harmless_Malware (packed).exe (2216) Properties
General Statistics Performance Threads Token Modules
Memory Environment Handles Comment
Strings... Refresh
Results - Harmless_Malware (packed).exe (2216)
937 results.
Results - Harmless_Malware (packed).exe (2216)
20 results.
Filter Save... Copy Close
Search Processes (Ctrl+K)
CPU

Address	Length	Result
0x3ee134	214	C:\Users\user\AppData\Local\Micro...
0x3ee6d8	107	C:\Users\user\AppData\Local\Micro...
0x6a0c48	20	http://www.sans.org/
0x6a0c68	20	http://www.sans.org/
0x6a0c88	18	Host: www.sans.org
0x6b24d0	40	http://www.sans.org/
0x6b2508	40	http://www.sans.org/
0x6b2870	12	www.sans.org
0x6b2888	12	www.sans.org
0x6b28a0	12	www.sans.org
0x6bf6d0	24	www.sans.org
0x6bf758	24	www.sans.org
0x6d3300	24	www.sans.org
0x6d3350	24	www.sans.org
0x6d337c	24	www.sans.org
0x6db318	107	C:\Users\user\AppData\Local\Micro...
0x6e4c6a	18	tp://www.sans.org/
0x23d4bc0	20	w.sans.org
0x23d4c30	24	www.sans.org
0x2a72d04	24	www.sans.org

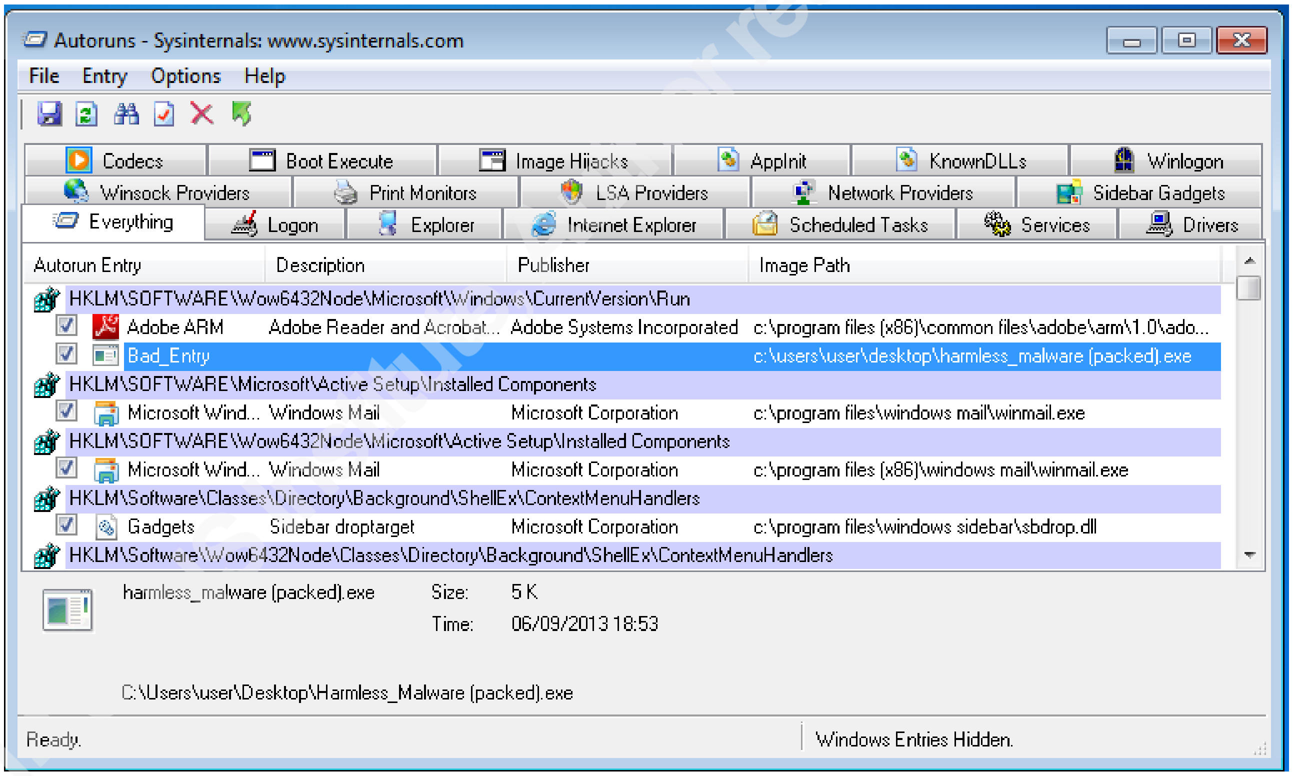
Description
Host Process for Windows Ser...
Adobe Acrobat Update Service
SQL Server VSS Writer - 64 Bit
Microsoft Windows Search In...
Microsoft Software Protection...
.NET Runtime Optimization S...
.NET Runtime Optimization S...
Host Process for Windows Tas...
Host Process for Windows Ser...
Host Process for Windows Ser...
Host Process for Windows Ser...
Host Process for Windows Ser...
Local Security Authority Proce...
Local Session Manager Service
Windows Logon User Interfac...
Client Server Runtime Process
Console Window Host
Windows Logon Application
Windows Explorer
LibreOffice
Paint
Process Hacker

3.4 Autoruns – Fully functional

(Autoruns, 2013)

We can see below that 'Autoruns' has been able to successfully identify and list the auto run entry that we added into the registry. This has been highlighted below, and we are able to use this software to modify the auto-run functionality.

As this entry has been created by a x86 program, the registry key enter was added under the Wow6432Node. This software is able to successfully identify and list that fact.



Autorun Entry	Description	Publisher	Image Path
HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Run			
Adobe ARM	Adobe Reader and Acrobat...	Adobe Systems Incorporated	c:\program files (x86)\common files\adobe\arm\1.0\ado...
Bad_Entry			c:\users\user\desktop\harmless_malware (packed).exe
HKLM\SOFTWARE\Microsoft\Active Setup\Installed Components			
Microsoft Wind...	Windows Mail	Microsoft Corporation	c:\program files\windows mail\winmail.exe
HKLM\SOFTWARE\Wow6432Node\Microsoft\Active Setup\Installed Components			
Microsoft Wind...	Windows Mail	Microsoft Corporation	c:\program files (x86)\windows mail\winmail.exe
HKLM\Software\Classes\Directory\Background\ShellEx\ContextMenuHandlers			
Gadgets	Sidebar droptarget	Microsoft Corporation	c:\program files\windows sidebar\sbdrop.dll
HKLM\Software\Wow6432Node\Classes\Directory\Background\ShellEx\ContextMenuHandlers			

harmless_malware (packed).exe Size: 5 K
Time: 06/09/2013 18:53

C:\Users\user\Desktop\Harmless_Malware (packed).exe

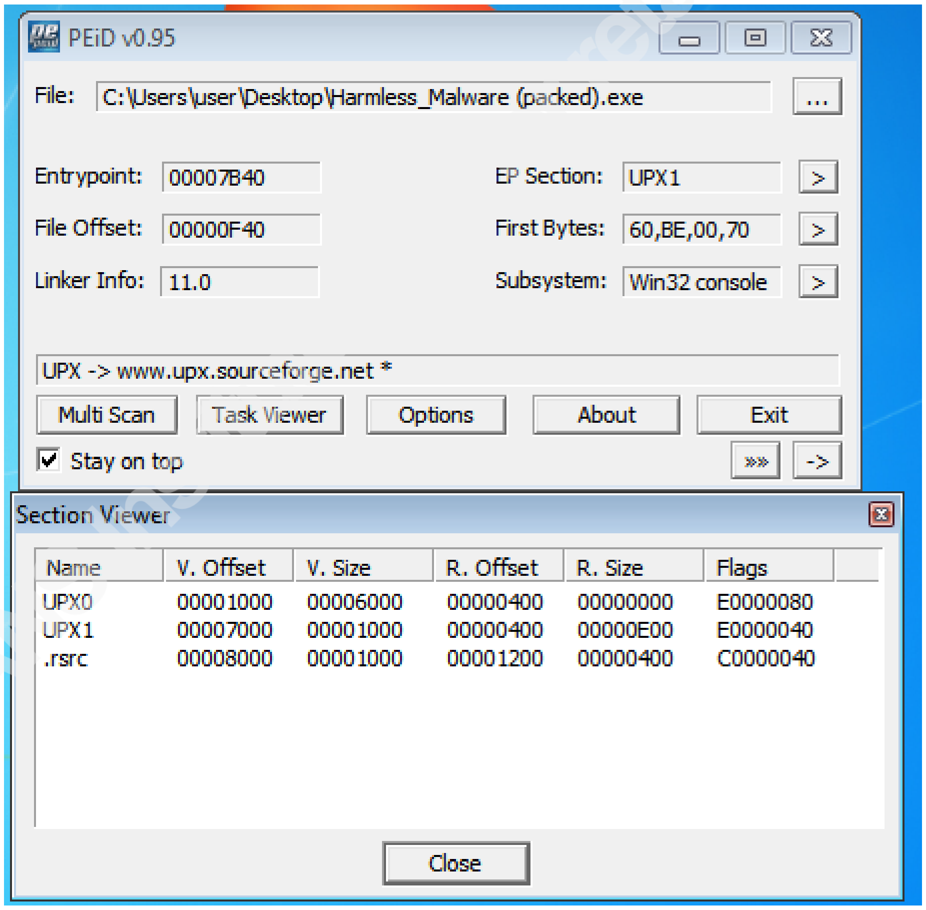
Ready. Windows Entries Hidden.

3.5 PEiD – Fully functional

(PEiD, 2013)

The packed executable was loaded into PEiD to see whether it would be able to identify the packer.

As can be seen, it has been able to identify the packer we have used as UPX and gives details about the sections within the file.



File: C:\Users\user\Desktop\Harmless_Malware (packed).exe

Entrypoint: 00007B40 EP Section: UPX1

File Offset: 00000F40 First Bytes: 60,BE,00,70

Linker Info: 11.0 Subsystem: Win32 console

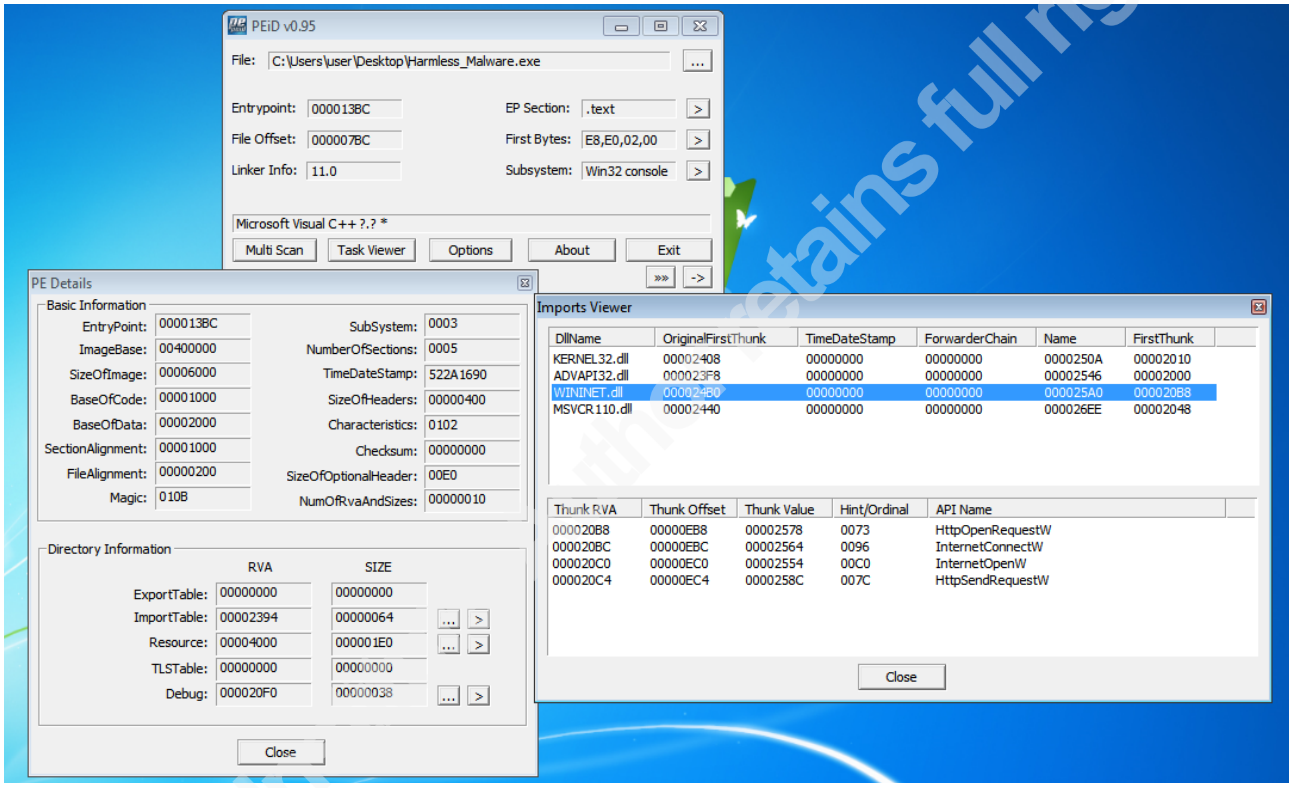
UPX -> www.upx.sourceforge.net *

☑ Stay on top

Section Viewer

Name	V. Offset	V. Size	R. Offset	R. Size	Flags
UPX0	00001000	00006000	00000400	00000000	E0000080
UPX1	00007000	00001000	00000400	00000E00	E0000040
.rsrc	00008000	00001000	00001200	00000400	C0000040

Furthermore, digging into the functionality which details PE information including the import table and other headers.



File:	C:\Users\user\Desktop\Harmless_Malware.exe
Entrypoint:	000013BC
File Offset:	000007BC
Linker Info:	11.0
EP Section:	.text
First Bytes:	E8,E0,02,00
Subsystem:	Win32 console

Microsoft Visual C++ ?.? *

Basic Information

EntryPoint:	000013BC
ImageBase:	00400000
SizeOfImage:	00006000
BaseOfCode:	00001000
BaseOfData:	00002000
SectionAlignment:	00001000
FileAlignment:	00000200
Magic:	010B
SubSystem:	0003
NumberOfSections:	0005
TimeDateStamp:	522A1690
SizeOfHeaders:	00000400
Characteristics:	0102
Checksum:	00000000
SizeOfOptionalHeader:	00E0
NumOfRvaAndSizes:	00000010

Directory Information

	RVA	SIZE
ExportTable:	00000000	00000000
ImportTable:	00002394	00000064
Resource:	00004000	000001E0
TLSTable:	00000000	00000000
Debug:	000020F0	00000038

DllName	OriginalFirstThunk	TimeDateStamp	ForwarderChain	Name	FirstThunk
KERNEL32.dll	00002408	00000000	00000000	0000250A	00002010
ADVAPI32.dll	000023F8	00000000	00000000	00002546	00002000
WININET.dll	000024B0	00000000	00000000	000025A0	000020B8
MSVCR110.dll	00002440	00000000	00000000	000026EE	00002048

Thunk RVA	Thunk Offset	Thunk Value	Hint/Ordinal	API Name
000020B8	00000EB8	00002578	0073	HttpOpenRequestW
000020BC	00000EBC	00002564	0096	InternetConnectW
000020C0	00000EC0	00002554	00C0	InternetOpenW
000020C4	00000EC4	0000258C	007C	HttpSendRequestW

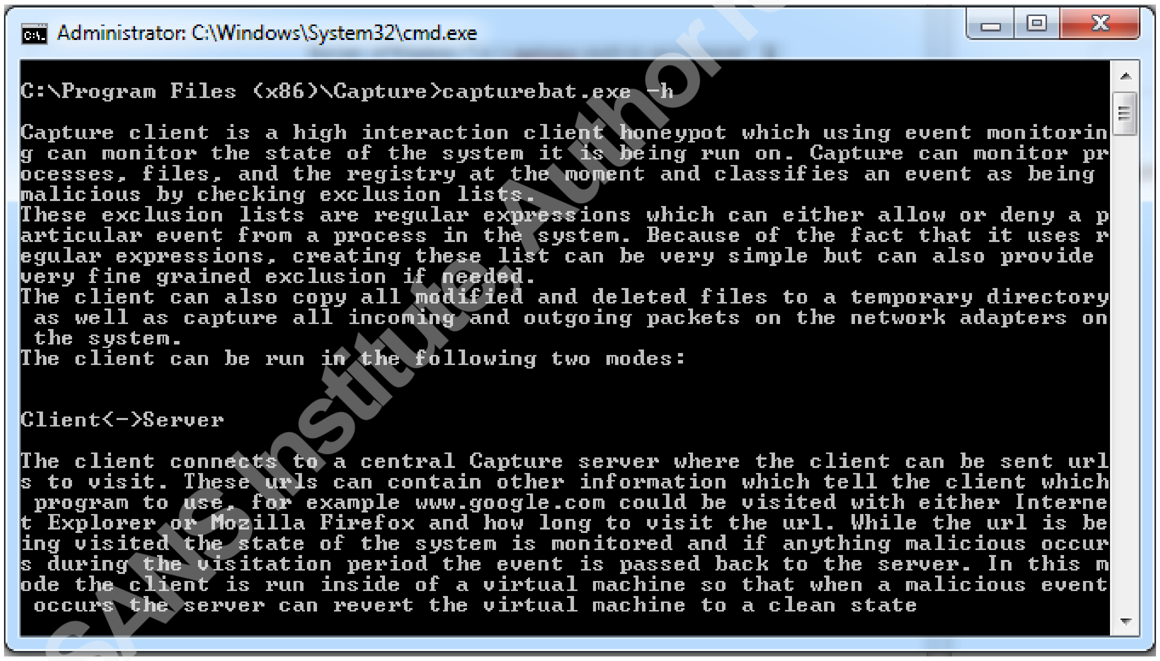
In the IAT view, APIs that have been used in the test executable can be easily recognized, in this screenshot the APIs used in the internet connection are seen.

3.6 Capture BAT – Partial functionality

(Capture BAT, 2013)

CaptureBAT is designed to monitor the system and attempts to identify suspicious activity. It has the functionality to monitor processes, files, registry and network traffic.

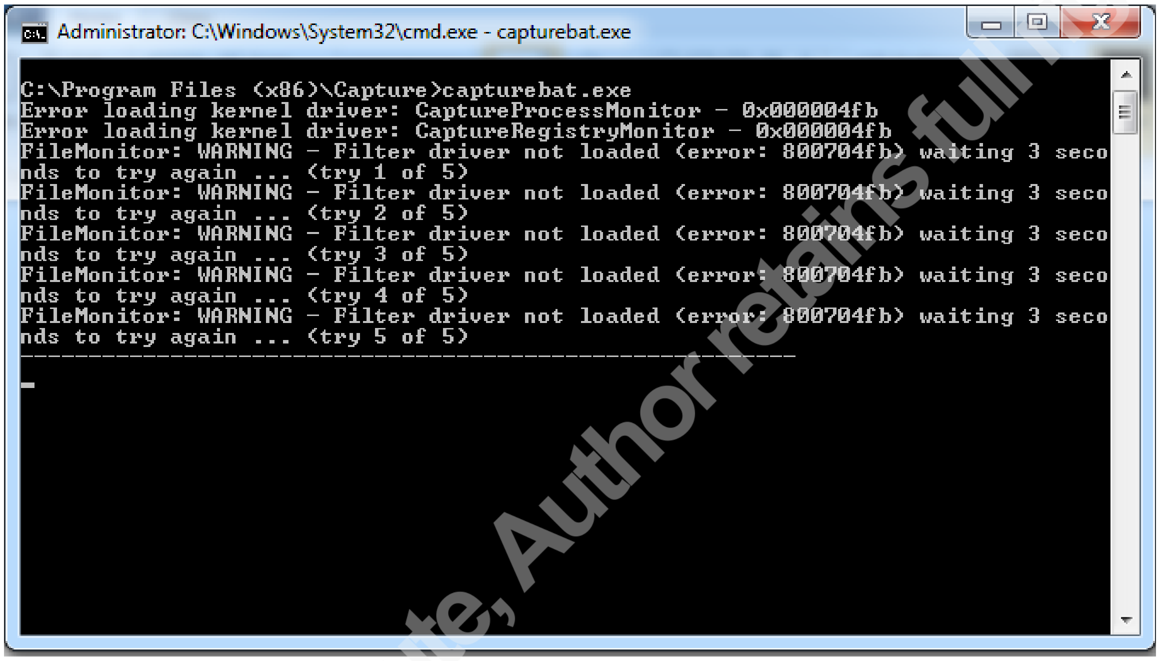
By running CaptureBAT with the -h argument, the help screen is displayed. This provides a detailed explanation of the full functionality of the program.



```
C:\Program Files (x86)\Capture>capturebat.exe -h

Capture client is a high interaction client honeypot which using event monitorin
g can monitor the state of the system it is being run on. Capture can monitor pr
ocesses, files, and the registry at the moment and classifies an event as being
malicious by checking exclusion lists.
These exclusion lists are regular expressions which can either allow or deny a p
articular event from a process in the system. Because of the fact that it uses r
egular expressions, creating these list can be very simple but can also provide
very fine grained exclusion if needed.
The client can also copy all modified and deleted files to a temporary directory
 as well as capture all incoming and outgoing packets on the network adapters on
 the system.
The client can be run in the following two modes:


Client<->Server

The client connects to a central Capture server where the client can be sent url
s to visit. These urls can contain other information which tell the client which
 program to use, for example www.google.com could be visited with either Interne
t Explorer or Mozilla Firefox and how long to visit the url. While the url is be
ing visited the state of the system is monitored and if anything malicious occur
s during the visitation period the event is passed back to the server. In this m
ode the client is run inside of a virtual machine so that when a malicious event
 occurs the server can revert the virtual machine to a clean state
```

Unfortunately although the program will execute and display what it was designed to do, it doesn't work fully under Windows 7 x64.

If we execute the program without any parameters we get the following output:



```
C:\Program Files (x86)\Capture>capturebat.exe
Error loading kernel driver: CaptureProcessMonitor - 0x000004fb
Error loading kernel driver: CaptureRegistryMonitor - 0x000004fb
FileMonitor: WARNING - Filter driver not loaded (error: 800704fb) waiting 3 seco
nds to try again ... (try 1 of 5)
FileMonitor: WARNING - Filter driver not loaded (error: 800704fb) waiting 3 seco
nds to try again ... (try 2 of 5)
FileMonitor: WARNING - Filter driver not loaded (error: 800704fb) waiting 3 seco
nds to try again ... (try 3 of 5)
FileMonitor: WARNING - Filter driver not loaded (error: 800704fb) waiting 3 seco
nds to try again ... (try 4 of 5)
FileMonitor: WARNING - Filter driver not loaded (error: 800704fb) waiting 3 seco
nds to try again ... (try 5 of 5)
---------------------------------------------------------
_
```

As can be seen the user is presented with error 0x4FB which according to the Microsoft 'System Error Codes' (Microsoft – Dev Center (B), 2013)

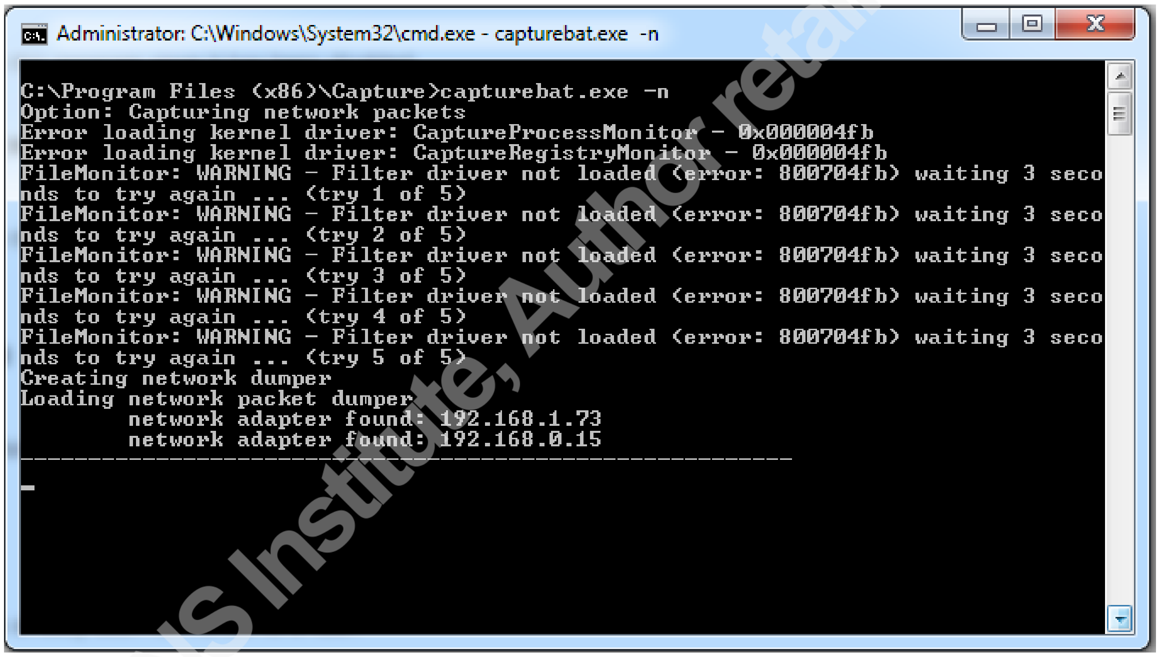
> ERROR_DRIVER_BLOCKED
> 1275 (0x4FB)
>
> This driver has been blocked from loading.

Windows 7 is not happy with the driver and has not allowed it to load. The kernel driver provides the core functionality of the program and accordingly without it, the execution fails.

However, there is limited functionality that does work under Windows 7. If the program is run with the -n argument, according to the help screen (-h) it will do the following:

-n Capture all incoming and outgoing network packets from the network adapters on the system and store them in .pcap files in the log directory

This element of the functionality appears to work correctly and loads after the errors thrown from the kernel driver load failure.



Administrator: C:\Windows\System32\cmd.exe - capturebat.exe -n

```
C:\Program Files (x86)\Capture>capturebat.exe -n
Option: Capturing network packets
Error loading kernel driver: CaptureProcessMonitor - 0x000004fb
Error loading kernel driver: CaptureRegistryMonitor - 0x000004fb
FileMonitor: WARNING - Filter driver not loaded (error: 800704fb) waiting 3 seco
nds to try again ... (try 1 of 5)
FileMonitor: WARNING - Filter driver not loaded (error: 800704fb) waiting 3 seco
nds to try again ... (try 2 of 5)
FileMonitor: WARNING - Filter driver not loaded (error: 800704fb) waiting 3 seco
nds to try again ... (try 3 of 5)
FileMonitor: WARNING - Filter driver not loaded (error: 800704fb) waiting 3 seco
nds to try again ... (try 4 of 5)
FileMonitor: WARNING - Filter driver not loaded (error: 800704fb) waiting 3 seco
nds to try again ... (try 5 of 5)
Creating network dumper
Loading network packet dumper
        network adapter found: 192.168.1.73
        network adapter found: 192.168.0.15
---------------------------------------------------------
_
```

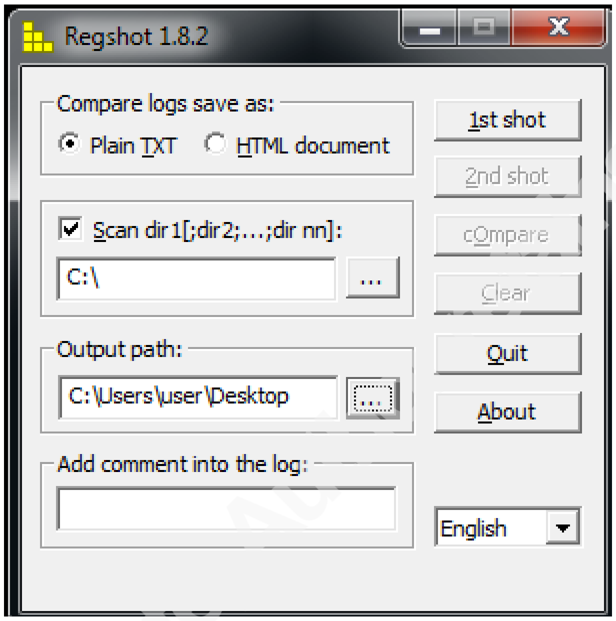
The PCAP files are successfully created and if loaded into Wireshark show what appears to be a successful packet capture.

Adam Kramer, adamkramer@hotmail.com

3.7 Regshot – Fully functional

(Regshot, 2013)

Regshot is designed to let the user take a snapshot of the file system and registry, execute a program, take a second snapshot and then compare the two.



Compare logs save as: [✓ Plain TXT] HTML document

[✓] Scan dir1[;dir2;...;dir nn]: C:\

Output path: C:\Users\user\Desktop

Add comment into the log:

English

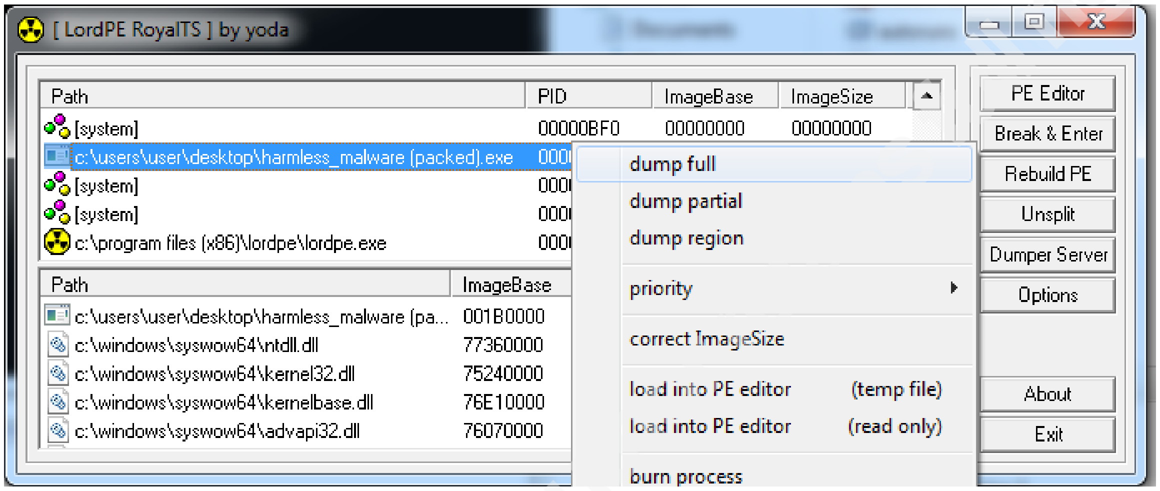
When tested by executing 'Harmless_Malware.exe' between the two snapshots the expected file system and registry changes are successfully captured and this program appears to be fully functional under the Windows 7 x64 environment.

3.8 LordPE – Appears functional

(LordPE, 2013)

LordPE is designed to dump running processes from memory and edit the PE headers.

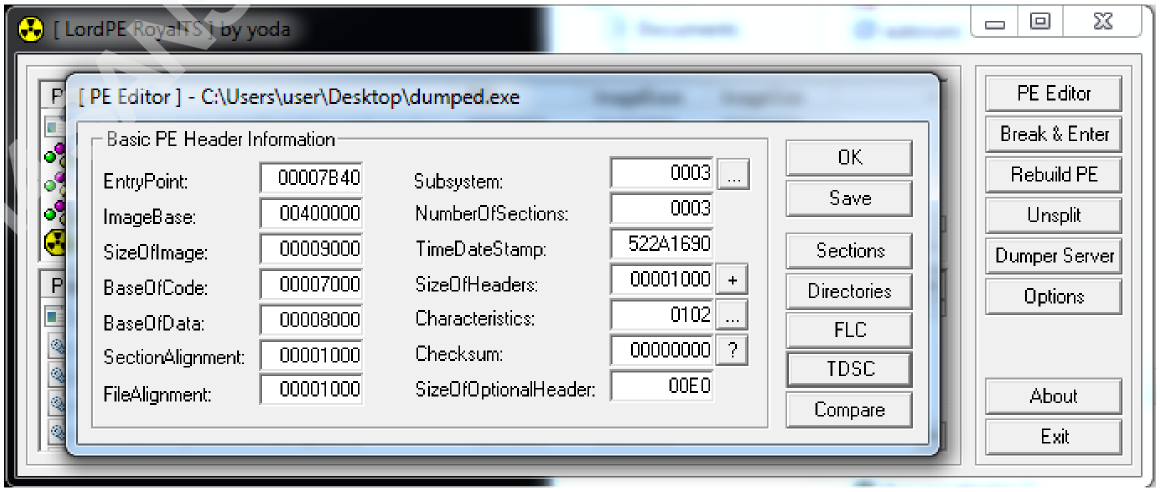
As can be seen below LordPE correctly identifies the process and associated DLLs.



```
Path                                                       PID       ImageBase  ImageSize
[system]                                                   00000BF0  00000000   00000000
c:\users\user\desktop\harmless_malware (packed).exe        000
[system]                                                   000
[system]                                                   000
c:\program files (x86)\lordpe\lordpe.exe                   000

Path                                                  ImageBase
c:\users\user\desktop\harmless_malware (pa...         001B0000
c:\windows\syswow64\ntdll.dll                         77360000
c:\windows\syswow64\kernel32.dll                      75240000
c:\windows\syswow64\kernelbase.dll                    76E10000
c:\windows\syswow64\advapi32.dll                      76070000
```

The process can be 'dumped' from memory into a file and when the resultant executable file is run through BinText to view embedded strings a similar result to the unpacked executable is seen.

Unsurprisingly the dumped executable will not run, likely due to incorrect headers. We can view and edit the headers but unless you are able to manually repair the file we may not be able to fix it. There are possible limitations in LordPE's ability to repair PE headers in Windows 7.



Basic PE Header Information

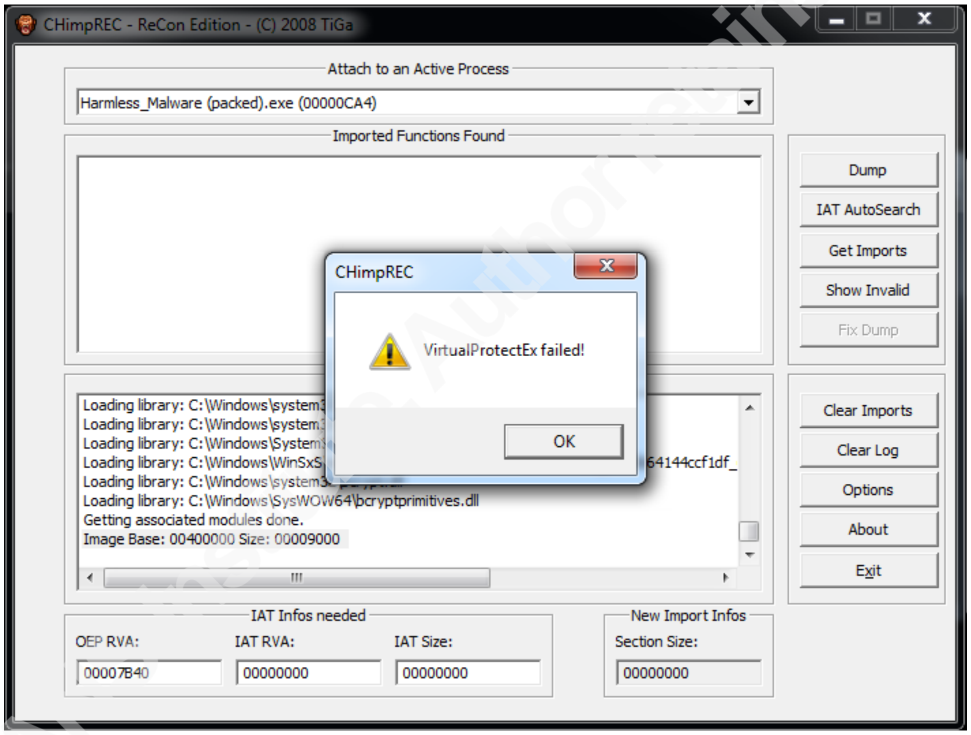
EntryPoint:	00007B40	Subsystem:	0003
ImageBase:	00400000	NumberOfSections:	0003
SizeOfImage:	00009000	TimeDateStamp:	522A1690
BaseOfCode:	00007000	SizeOfHeaders:	00001000
BaseOfData:	00008000	Characteristics:	0102
SectionAlignment:	00001000	Checksum:	00000000
FileAlignment:	00001000	SizeOfOptionalHeader:	00E0

3.9 CHimpREC – Testing failed

(CHimpREC, 2013)

CHimpREC is an alternative to LordPE with the functionality to dump a process from memory to a file and then attempt to fix the headers so the process can be executed.

It comes in two flavors, CHimpREC and CHimpREC 64. During tests of CHimpREC on Windows 7 x64 the test process was identified and able to be selected however when the 'Dump' button was selected the error message “VirtualProtectEx failed!” arose. This occurred whether the process was executed as administrator or not.



Harmless_Malware (packed).exe (00000CA4)

VirtualProtectEx failed!

```
Loading library: C:\Windows\system3
Loading library: C:\Windows\system3
Loading library: C:\Windows\System3
Loading library: C:\Windows\WinSxS\                                    64144ccf1df_
Loading library: C:\Windows\system3
Loading library: C:\Windows\SysWOW64\bcryptprimitives.dll
Getting associated modules done.
Image Base: 00400000 Size: 00009000
```

OEP RVA: 00007B40
IAT RVA: 00000000
IAT Size: 00000000
Section Size: 00000000

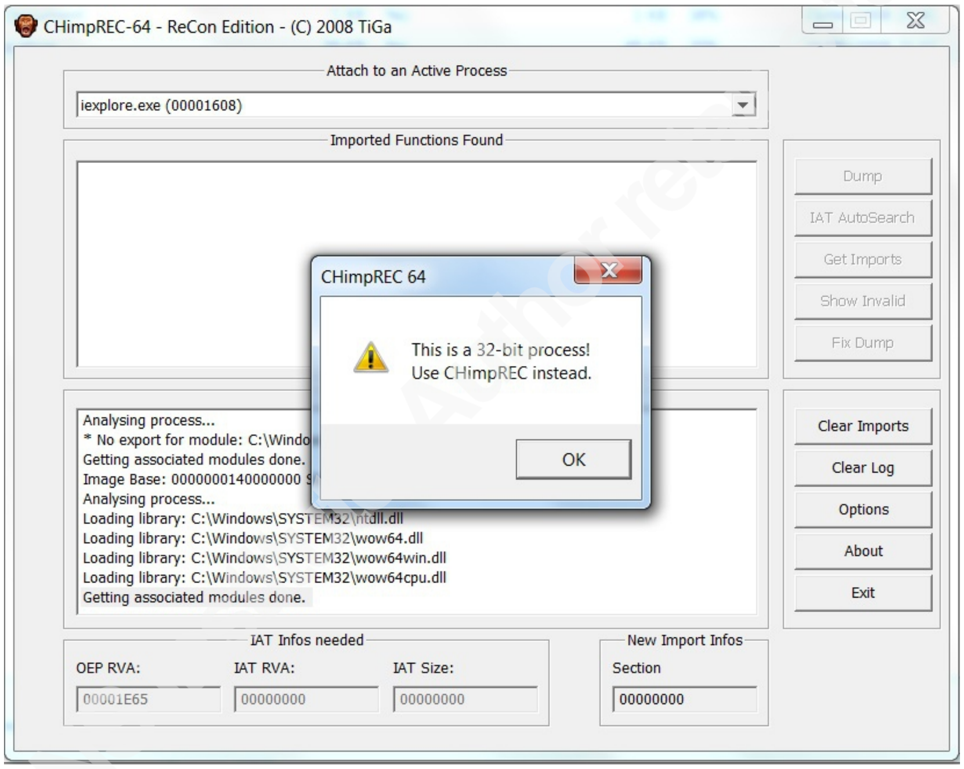
Accordingly as this was being executed on a 64 bit operating system attempts were made to run the 'CHimpREC 64' version, however unfortunately the following error occurred prior to any window appearing:

“The application has failed to start because its side-by-side configuration is incorrect. Please see the application event log or use the command-line sxstrace.exe tool for more detail.”

This is likely due to the installed version of the required runtime libraries however after several attempts at at various runtime library versions there was no successful execution.

As the side-by-side configuration issue was likely due to the run time libraries installed on that particular machine, the program was run on another system running Windows 7 x64.

In this case, the program ran and loaded, however when a 32 bit process was selected an error message appeared stating "This is a 32-bit process! Use CHimpREC instead." This is shown in the screenshot below.



This is a 32-bit process!
Use CHimpREC instead.

```
Analysing process...
* No export for module: C:\Windo
Getting associated modules done.
Image Base: 0000000140000000 S
Analysing process...
Loading library: C:\Windows\SYSTEM32\ntdll.dll
Loading library: C:\Windows\SYSTEM32\wow64.dll
Loading library: C:\Windows\SYSTEM32\wow64win.dll
Loading library: C:\Windows\SYSTEM32\wow64cpu.dll
Getting associated modules done.
```

OEP RVA: 00001E65 | IAT RVA: 00000000 | IAT Size: 00000000 | Section: 00000000

On the same system, the 32 bit version of CHimpREC was run and the same program selected.

Unfortunately again in this case the same issue arose and the "VirtualProtectEx failed!" error appeared when a dump was attempted.

Accordingly during the testing this program did not function correctly.

3.10 Ollydbg – Appears functional

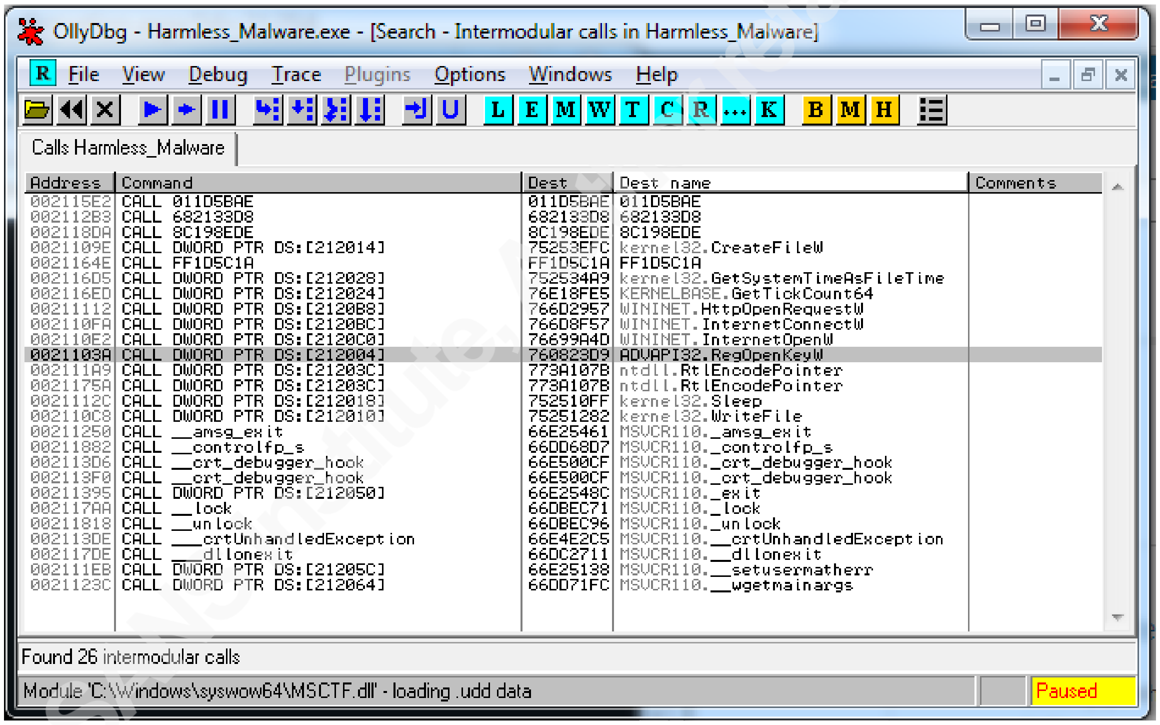
(Ollydbg, 2013)

In the analysis of executable files there are an unlimited number of issues that could be encountered depending on code sequences, hardware and debug settings. Accordingly a number of focused tests were completed to identify whether the core functionality is present.

These tests are based on Ollydbg v2 as v1 causes a number of errors on launch.

Ollydbg v2 was launched and the test program was loaded in – Successful

The intermodular call list was loaded and all expected APIs were identified – Successful

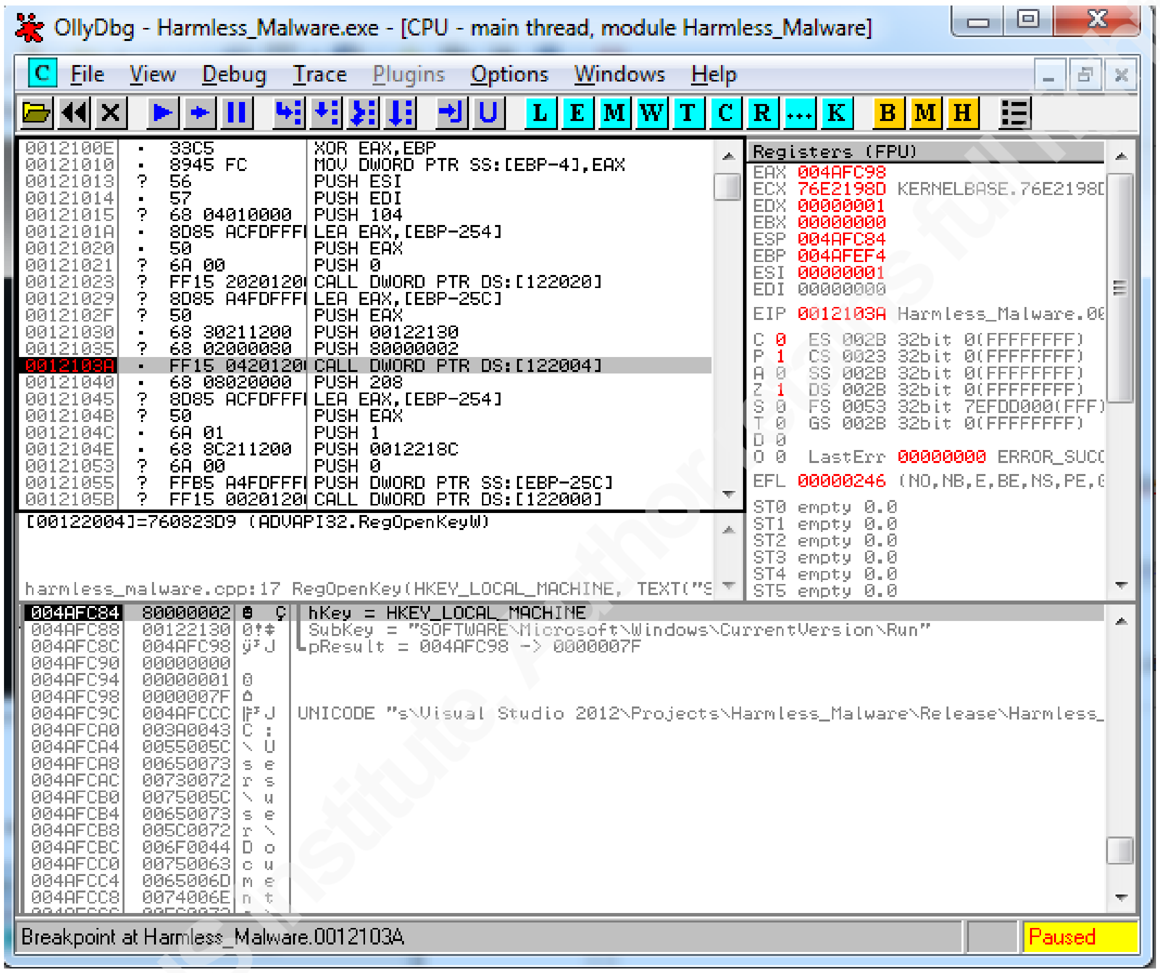

OllyDbg - Harmless_Malware.exe - [Search - Intermodular calls in Harmless_Malware]
File View Debug Trace Plugins Options Windows Help
Calls Harmless_Malware

```
Address   Command                          Dest      Dest name                               Comments
002115E2  CALL 011D5BAE                    011D5BAE  011D5BAE
002112B3  CALL 682133D8                    682133D8  682133D8
002118DA  CALL 8C198EDE                    8C198EDE  8C198EDE
0021109E  CALL DWORD PTR DS:[212014]       75253EFC  kernel32.CreateFileW
0021164E  CALL FF1D5C1A                    FF1D5C1A  FF1D5C1A
002116D5  CALL DWORD PTR DS:[212028]       752534A9  kernel32.GetSystemTimeAsFileTime
002116ED  CALL DWORD PTR DS:[212024]       76E18FE5  KERNELBASE.GetTickCount64
00211112  CALL DWORD PTR DS:[2120B8]       766D2957  WININET.HttpOpenRequestW
002110FA  CALL DWORD PTR DS:[2120BC]       766D8F57  WININET.InternetConnectW
002110E2  CALL DWORD PTR DS:[2120C0]       76699A4D  WININET.InternetOpenW
0021103A  CALL DWORD PTR DS:[212004]       760823D9  ADVAPI32.RegOpenKeyW
002111A9  CALL DWORD PTR DS:[21203C]       773A107B  ntdll.RtlEncodePointer
0021175A  CALL DWORD PTR DS:[21203C]       773A107B  ntdll.RtlEncodePointer
0021112C  CALL DWORD PTR DS:[212018]       752510FF  kernel32.Sleep
002110C8  CALL DWORD PTR DS:[212010]       75251282  kernel32.WriteFile
00211250  CALL __amsg_exit                 66E25461  MSVCR110._amsg_exit
00211882  CALL __controlfp_s               66DD68D7  MSVCR110._controlfp_s
002113D6  CALL __crt_debugger_hook         66E500CF  MSVCR110._crt_debugger_hook
002113F0  CALL __crt_debugger_hook         66E500CF  MSVCR110._crt_debugger_hook
00211395  CALL DWORD PTR DS:[212050]       66E2548C  MSVCR110._exit
002117AA  CALL __lock                      66DBEC71  MSVCR110._lock
00211818  CALL __unlock                    66DBEC96  MSVCR110._unlock
002113DE  CALL ___crtUnhandledException    66E4E2C5  MSVCR110.__crtUnhandledException
002117DE  CALL ___dllonexit                66DC2711  MSVCR110.__dllonexit
002111EB  CALL DWORD PTR DS:[21205C]       66E25138  MSVCR110.__setusermatherr
0021123C  CALL DWORD PTR DS:[212064]       66DD71FC  MSVCR110.__wgetmainargs
```

Found 26 intermodular calls

Module 'C:\Windows\syswow64\MSCTF.dll' - loading .udd data   Paused

A breakpoint was added in the part of code that interacts with the registry and on break the screenshot shows in clear text the registry key to be added – Successful


OllyDbg - Harmless_Malware.exe - [CPU - main thread, module Harmless_Malware]
C File View Debug Trace Plugins Options Windows Help
L E M W T C R ... K B M H

```
0012100E  .   33C5            XOR EAX,EBP
00121010  .   8945 FC         MOV DWORD PTR SS:[EBP-4],EAX
00121013  ?   56              PUSH ESI
00121014  .   57              PUSH EDI
00121015  ?   68 04010000     PUSH 104
0012101A  .   8D85 ACFDFFFI   LEA EAX,[EBP-254]
00121020  .   50              PUSH EAX
00121021  ?   6A 00           PUSH 0
00121023  ?   FF15 2020120    CALL DWORD PTR DS:[122020]
00121029  ?   8D85 A4FDFFFI   LEA EAX,[EBP-25C]
0012102F  ?   50              PUSH EAX
00121030  .   68 30211200     PUSH 00122130
00121035  ?   68 02000080     PUSH 80000002
0012103A  .   FF15 0420120    CALL DWORD PTR DS:[122004]
00121040  .   68 08020000     PUSH 208
00121045  ?   8D85 ACFDFFFI   LEA EAX,[EBP-254]
0012104B  ?   50              PUSH EAX
0012104C  .   6A 01           PUSH 1
0012104E  .   68 8C211200     PUSH 0012218C
00121053  ?   6A 00           PUSH 0
00121055  ?   FFB5 A4FDFFFI   PUSH DWORD PTR SS:[EBP-25C]
0012105B  ?   FF15 0020120    CALL DWORD PTR DS:[122000]
[00122004]=760823D9 (ADVAPI32.RegOpenKeyW)

harmless_malware.cpp:17 RegOpenKey(HKEY_LOCAL_MACHINE, TEXT("S
```

```
Registers (FPU)
EAX 004AFC98
ECX 76E2198D KERNELBASE.76E2198D
EDX 00000001
EBX 00000000
ESP 004AFC84
EBP 004AFEF4
ESI 00000001
EDI 00000000

EIP 0012103A Harmless_Malware.00

C 0  ES 002B 32bit 0(FFFFFFFF)
P 1  CS 0023 32bit 0(FFFFFFFF)
A 0  SS 002B 32bit 0(FFFFFFFF)
Z 1  DS 002B 32bit 0(FFFFFFFF)
S 0  FS 0053 32bit 7EFDD000(FFF)
T 0  GS 002B 32bit 0(FFFFFFFF)
D 0
O 0  LastErr 00000000 ERROR_SUCC
EFL 00000246 (NO,NB,E,BE,NS,PE,G

ST0 empty 0.0
ST1 empty 0.0
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
ST5 empty 0.0
```

```
004AFC84  80000002  ȅ  Ç  hKey = HKEY_LOCAL_MACHINE
004AFC88  00122130  0!‡   SubKey = "SOFTWARE\Microsoft\Windows\CurrentVersion\Run"
004AFC8C  004AFC98  ÿ³J   pResult = 004AFC98 -> 0000007F
004AFC90  00000000
004AFC94  00000001  0
004AFC98  0000007F  ⌂
004AFC9C  004AFCCC  ╠³J      UNICODE "s\Visual Studio 2012\Projects\Harmless_Malware\Release\Harmless_
004AFCA0  003A0043  C :
004AFCA4  0055005C  \ U
004AFCA8  00650073  s e
004AFCAC  00730072  r s
004AFCB0  0075005C  \ u
004AFCB4  00650073  s e
004AFCB8  005C0072  r \
004AFCBC  006F0044  D o
004AFCC0  00750063  c u
004AFCC4  0065006D  m e
004AFCC8  0074006E  n t
```

Breakpoint at Harmless_Malware.0012103A   Paused

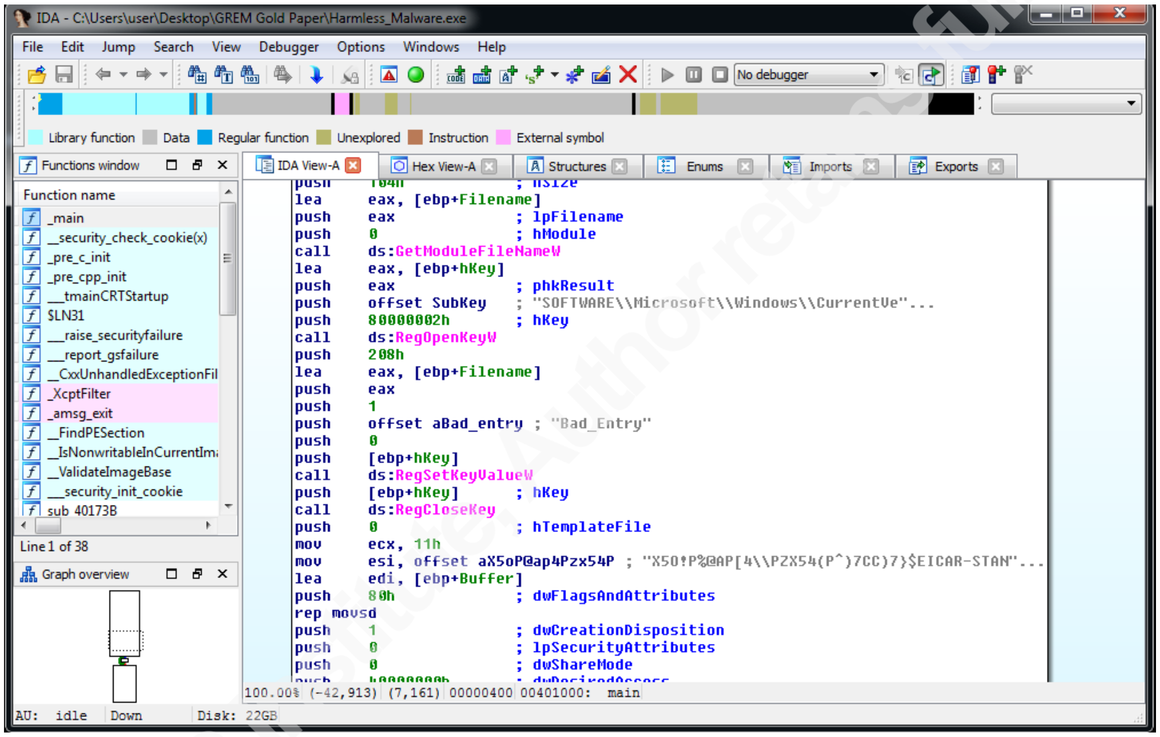
The basic tests have passed, the user is able to load, step through, set breakpoints and analyze various areas of code with access to the memory dump, registers and stack.

There are a number of plugins available that are compatible with Ollydbg v2 including one designed to dump running processes from memory with the ability to set various elements as was available with a similar version 1 plugin.

3.11 IDA Pro – Appears functional

(IDA Pro, 2013)

The test executable was successfully loaded into IDA Pro and as can be seen from the screenshot below, a successful disassemble results in the registry key, test file output and expected API calls being available for visual inspection.



```
_main
__security_check_cookie(x)
_pre_c_init
_pre_cpp_init
__tmainCRTStartup
$LN31
__raise_securityfailure
__report_gsfailure
__CxxUnhandledExceptionFil
_XcptFilter
_amsg_exit
__FindPESection
__IsNonwritableInCurrentIm
__ValidateImageBase
__security_init_cookie
sub_40173B
```

```
push    104h              ; nSize
lea     eax, [ebp+Filename]
push    eax               ; lpFilename
push    0                 ; hModule
call    ds:GetModuleFileNameW
lea     eax, [ebp+hKey]
push    eax               ; phkResult
push    offset SubKey     ; "SOFTWARE\\Microsoft\\Windows\\CurrentVe"...
push    80000002h         ; hKey
call    ds:RegOpenKeyW
push    208h
lea     eax, [ebp+Filename]
push    eax
push    1
push    offset aBad_entry ; "Bad_Entry"
push    0
push    [ebp+hKey]
call    ds:RegSetKeyValueW
push    [ebp+hKey]        ; hKey
call    ds:RegCloseKey
push    0                 ; hTemplateFile
mov     ecx, 11h
mov     esi, offset aX5oP@ap4Pzx54P ; "X5O!P%@AP[4\\PZX54(P^)7CC)7}$EICAR-STAN"...
lea     edi, [ebp+Buffer]
push    80h               ; dwFlagsAndAttributes
rep movsd
push    1                 ; dwCreationDisposition
push    0                 ; lpSecurityAttributes
push    0                 ; dwShareMode
push    40000000h         ; dwDesiredAccess
```

```
100.00% (-42,913) (7,161) 00000400 00401000: main
AU:  idle   Down    Disk: 22GB
```

As with Ollydbg, there is a vast amount of functionality available in IDA Pro and accordingly a significant amount of further testing on various programs would be required to establish full compatibility.

Adam Kramer, adamkramer@hotmail.com

4. Conclusion

It appears from the tests conducted that the majority of programs are able to run successfully on Windows 7 (x64) when the malicious executable being run has been compiled on x86 architecture as might be expected with an author intending for the widest possible distribution.

The programs which are used for behavioral analysis are the most successful, likely because they are independent of the malicious program being analyzed and are linking directly with the Windows API.

Code analysis being the more complex side of reverse engineering is, by it's very nature, more likely to cause unpredictable results, especially in the area of PE header repair such that it can be successfully run on the analysis system.

However having reviewed Ollydbg v2 as a debugger and IDA Pro as a disassembler there is nothing to suggest that a successful code analysis could not take place.

There may be other tools which can be used to dump processes from memory although LordPE appears to have been able to do this successfully, and there are plugins available for Ollydbg v2 to assist with this.

The author believes that there is sufficient functionality available in the tested tools for a thorough analysis to take place in the Windows 7 x64 environment and for the analyst to build a complete tool set and be able to operate successfully.

5. References

SANS Institute (2013). FOR610: Reverse-Engineering Malware Analysis Tools and Techniques - Laptop requirements.
Retrieved from: http://www.sans.org/course/reverse-engineering-malware-malware-analysis-tools-techniques#section_with_details_laptop_description

Microsoft (2013). Windows lifecycle fact sheet.
Retrieved from: http://windows.microsoft.com/en-US/windows/products/lifecycle

Tech Republic (2013). Latest poll results: What percentage of your enterprise is running Windows XP?
Retrieved from: http://www.techrepublic.com/blog/windows-and-office/latest-poll-results-what-percentage-of-your-enterprise-is-running-windows-xp/

EICAR (2003). EICAR anti-malware test file.
http://www.eicar.org/86-0-Intended-use.html

MSDN (2013). MSDN library.
http://msdn.microsoft.com/library

Microsoft – Dev Center (A) (2013) – Running 32-bit applications.
http://msdn.microsoft.com/en-us/library/windows/desktop/aa384249(v=vs.85).aspx

UPX (2013). Ultimate Packer for eXecutables.
Available from: http://upx.sourceforge.net

BinText (2013). BinText - Finds Ascii, Unicode and Resource strings in a file.
Available from: http://www.mcafee.com/uk/downloads/free-tools/bintext.aspx

Process Monitor (2013). Windows sysinternals - Process Monitor.
Available from: http://technet.microsoft.com/en-gb/sysinternals/bb896645.aspx

Process Hacker (2013). Process Hacker - A free, powerful, multi-purpose tool that helps you monitor system resources, debug software and detect malware.
Available from: http://processhacker.sourceforge.net/

Autoruns (2013). Autoruns for Windows.
Available from: http://technet.microsoft.com/en-gb/sysinternals/bb963902.aspx

PEiD (2013).
[Official site no longer active - Available from various untrusted sources]

Capture BAT (2013). [No longer available from official site - Available from: http://zeltser.com/reverse-malware/capturebat.html]

Microsoft – Dev Center (B) (2013). System Error Codes (1000 – 1299). Retrieved from: http://msdn.microsoft.com/en-us/library/windows/desktop/ms681383(v=vs.85).aspx#ERROR_DRIVER_BLOCKED

Regshot (2013). Regshot - open-source (LGPL) registry compare utility
Available from: http://sourceforge.net/projects/regshot/

LordPE (2013).
[Official site no longer active - Available from various untrusted sources]

CHimpREC (2013).
[Official site no longer active - Available from various untrusted sources]

Ollydbg (2013). OllyDbg is a 32-bit assembler level analyzing debugger for Microsoft® Windows®
Available from: http://www.ollydbg.de/

IDA Pro (2013). IDA is the Interactive DisAssembler.
Available from: https://www.hex-rays.com

Appendix A – Source code of Harmless_Malware.exe

COPYRIGHT NOTICE

PROGRAM SOURCE – Harmless_Malware.cpp

```
#include "stdafx.h"

int _tmain(int argc, _TCHAR* argv[])
{

        /*   Test Sequence: REGISTRY WRITE
        *************************************
        1. Identify the path to the executable
        2. Add a registry key for an autorun
        *************************************/
        TCHAR current_filepath[MAX_PATH];
        HKEY registry_key;

        GetModuleFileName(NULL, current_filepath, MAX_PATH);

        RegOpenKey(HKEY_LOCAL_MACHINE,
        TEXT("SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run"),
        &registry_key);

        RegSetKeyValue(registry_key, 0, TEXT("Bad_Entry"), REG_SZ,
        current_filepath, sizeof(current_filepath));

        RegCloseKey(registry_key);
        /*   Test Sequence: FILE WRITE
        ***************************************
        1. Set a variable to the EICAR test string
        2. Open & write to the file (avtest.txt)
        ***************************************/
        HANDLE output_file;
        DWORD bytes_written = 0;
        char data_to_dump[] = "X5O!P%@AP[4\\PZX54(P^)7CC)7}$EICAR-
        STANDARD-ANTIVIRUS-TEST-FILE!$H+H*";

        output_file = CreateFile(TEXT("avtest.txt"), GENERIC_WRITE, 0, NULL,
                CREATE_NEW, FILE_ATTRIBUTE_NORMAL, NULL);

        WriteFile(output_file, data_to_dump, strlen(data_to_dump),
        &bytes_written, NULL);

        CloseHandle(output_file);

        /*   Test Sequence: INTERNET CONNECTION
        ***************************************
        1. Open internet connection to www.sans.org
        2. Request the main page
        Note: We are not interpreting the response
        ***************************************/
        HINTERNET internet_session = InternetOpen(TEXT("Suspect File"),
        INTERNET_OPEN_TYPE_PRECONFIG, NULL, NULL, 0);

        HINTERNET internet_connection = InternetConnect(internet_session,
        TEXT("www.sans.org"), INTERNET_DEFAULT_HTTP_PORT, NULL, NULL,
        INTERNET_SERVICE_HTTP, 0, 1);

        HINTERNET internet_request = HttpOpenRequest(internet_connection,    NULL,
                TEXT("/"),      NULL, NULL, NULL, 0, 0);

        HttpSendRequest(internet_request, NULL, 0, NULL, 0);


        /*   Test Sequence: 10 MINUTE SLEEP
        ***************************************
        1. Sleep for 10 mins allowing time to attach
        ***************************************/
        Sleep(600000);

        return 0;
}
```

Adam Kramer, adamkramer@hotmail.com

Last Updated: April 8th, 2014

Upcoming Training

SANS Security West 2014	San Diego, CA	May 08, 2014 - May 17, 2014	Live Event
Mentor Session - FOR 610	Columbia, MD	May 21, 2014 - Jul 23, 2014	Mentor
Digital Forensics & Incident Response Summit	Austin, TX	Jun 03, 2014 - Jun 10, 2014	Live Event
Community SANS Ottawa	Ottawa, ON	Jun 16, 2014 - Jun 21, 2014	Community SANS
SANSFIRE 2014	Baltimore, MD	Jun 21, 2014 - Jun 30, 2014	Live Event
SANS vLive - FOR610: Reverse-Engineering Malware: Malware Analysis Tools and Techniques	FOR610 - 201407,	Jul 14, 2014 - Aug 20, 2014	vLive
SANS Virginia Beach 2014	Virginia Beach, VA	Aug 18, 2014 - Aug 29, 2014	Live Event
SANS Baltimore 2014	Baltimore, MD	Sep 22, 2014 - Sep 27, 2014	Live Event
SANS DFIR Prague 2014	Prague, Czech Republic	Sep 29, 2014 - Oct 11, 2014	Live Event
SANS vLive - FOR610: Reverse-Engineering Malware: Malware Analysis Tools and Techniques	FOR610 - 201410,	Oct 13, 2014 - Nov 19, 2014	vLive
Community SANS Paris @ HSC - FOR610 (in French)	Paris, France	Nov 24, 2014 - Nov 28, 2014	Community SANS
SANS OnDemand	Online	Anytime	Self Paced
SANS SelfStudy	Books & MP3s Only	Anytime	Self Paced