MORGAN STANLEY

Physical Memory Standard Operating Procedures

HBGary Memory Forensic Tools

Phil Wallisch

5/11/2010

This document details the procedures that Morgan Stanley CERT will perform to acquire and analyze physical memory from target systems. Fastdump Professional and Responder Professional by HBGary are described and use case examples are provided.

Table of Contents

1. Executive Summary

Memory forensics allows MSCERT to become more effective and agile regarding the acquisition of actionable intelligence. Traditional disk forensic approaches to investigations are slow and non-scalable. Large amounts of data must be acquired, transferred, and then analyzed. Memory forensics reveal what the true running state of a target system is at the time of acquisition. Hidden processes and other system activities are made available to an analyst by analyzing a smaller set of data than disk forensics.

This document details Morgan Stanley's (MS) Standard Operating Procedures (SOPs) for acquiring and analyzing physical memory using the HBGary forensic toolset. Fastdump Professional and Responder Professional usage are detailed through a case study methodology.

2. Memory Acquisition

2.1. Background

HBGary Fastdump Professional (FDPro) is the approved tool for memory acquisition. When a system has been identified as requiring further investigation, FDPro should be deployed. Systems may be identified through a variety of means such as IDS alerts, AV logs, Proxy alerts, help desk ticket, or other enterprise security mechanisms.

2.2. FDPro Features

FDPro has a number of features that allow for varying levels of forensic integrity, speed, compression, and thoroughness. The combination of features depends on the circumstances of the incident.

2.2.1. Physical Memory Acquisition

Physical memory acquisition is the core component of FDPro. It performs a "dd" style memory dump which is non-proprietary in nature. Third-party tools can process this memory dump assuming they support the underlying operation systems.

2.2.2. Pagefile Acquisition

Modern Microsoft Windows operating systems use the physical disk to extend memory capacity. Information that is dynamically stored in on the disk is contained in the "pagefile.sys". In order capture a system's entire memory space the pagefile.sys must be acquired and analyzed. This feature is useful when investigating user activities due to the pagefile's ability to store volatile data for extended periods of time such as visited URLs.

2.2.3. Process Probing

The Windows operating system does not load an entire executable into memory when it is launched. The code that is required to run the function of interest is entered in memory. The remaining code is on the filesystem until called. The process probe feature forces code from disk into memory as well as data that had been paged out to pagefile.sys. Process probe allows an analyst to view more strings and code in memory where it has been deobfuscated. This is a powerful feature when dealing with packed or obfuscated programs. An analyst can probe all processes or only non-system processes which is a "smart probe".

It should be noted that process probing is disruptive to the target system regarding forensic quality of the image. The probe alters the state of the system but this is acceptable under most circumstances. If a memory image is required for litigation purposes the process probe should only be used after a more forensically sound approach to memory acquisition has been completed.

2.2.4. Compression

FDPro supports compression of memory dumps. Physical memory compresses very well due to the abundant null bytes sequences. It is useful to compress acquisitions when transferring across a network connection with limited bandwidth. It should be noted that compression is only supported on acquisitions with pagefile.sys (known as .hpak format).

2.2.5. Strict Acquisition

FDPro acquires memory in 1024KB increments. This makes acquisitions of large images fast. The 1024KB size requires a matching sized buffer in memory. Although buffers are created out of unallocated memory, this can potentially overwrite data that exists in the buffer space. The “strict” acquisition mode forces FDPro to use 4KB increments. This reduces the risk that valuable data is overwritten but increased the time required to acquire memory.

2.3. FDPro Execution

The FDPro executable is deployed on a network drive which is accessible to target systems, for example the “U:” drive. FDPro does not need to be copied locally to the target. The memory dump should be dropped locally to the target system however. Network based acquisitions are not supported at this time due to lack of reliable transfers. The image will be taken locally and transferred once completed. The executable is launched using psexec over the network from the analyst workstation to the target system.

2.4. Case Studies

2.4.1. Anti-Virus Alert

Background: The MSCERT team receives a ticket stating that a system with hostname “JSMITH2600” has generated an Anti-Virus alert. The alert indicates that a “keygen” program has been accessed and blocked. A decision must be made to either clean the system of the keygen program or resinstall the entire system due to a malware infection. Keygen programs are notorious for carrying Trojan horse programs that allow attackers to steal corporate or personal information from victims.

Response: The MSCERT team launches FDPro to acquire the physical memory of the JSMITH2600 system and probe all processes (Figure 1). This provides the team with a manageable size of data to determine the state of the target system’s integrity. If malware is identified, the process probe will provide additional intelligence required to remediate the system and scan the enterprise for other victims.

Figure 1

```
C:\>psexec \\JSMITH2600 -u pcadmin -p <target host PCAdmin password> -c u:\fdpro
c:\windows\JSMITH2600.bin -probe all
```

The memory image must then be retrieved for analysis. The team can now map a network share to the target system and retrieve the memory dump. The memory dump is placed on the analyst workstation in the c:\memory_images folder (Figure 2).

Figure 2

```
C:\ >net use * \\JSMITH2600\c$ /user:pcadmin
C:\>robocopy \\JSMITH2600\c$\windows\JSMITH2600.bin c:\memory_images
```

The target host is then cleaned of all memory acquisition files. Finally the network share is deleted (Figure 3).

Figure 3

```
C:\>del  \\JSMITH2600\c$\windows\JSMITH2600.bin
C:\> net use \\JSMITH2600\c$ /del
```

2.4.2. Electronic Discovery

Background: An external entity informs Morgan Stanley that an IP address originating from their net block has been observed participating in the transfer of illegal content. MSCERT associates the activity with an internal system named “DCHEN2600”. Risk management has determined that law enforcement will be involved and that litigation will be pursued.

Response: The MSCERT team launches FDPro from a local USB drive to minimize the impact on the disk. The USB drive has a “\tools” directory where fdpro.exe resides and an “\images” directory where memory images are stored. It is connected by a local IT resource. The USB drive is recognized as the “E:” drive on the target system and is formatted in NTFS to support large files. The goal is to acquire the complete virtual memory of the target system in the most forensically sound manner.

Once the USB drive has been connected the memory acquisition is conducted using remote access via psexec (Figure 4). The complete virtual memory space is acquired through the use of the “.hpak” modifier and 4KB increments are used due to the “-strict” option.

Figure 4

```
psexec \\DCHEN2600 -u pcadmin -p <target host PCAdmin password> -c e:\tools\fdpro.exe
e:\images\DCHEN2600.hpak -strict
```

The target system does not require cleaning of memory image remnants due to the use of the USB drive.

3. Memory Analysis

3.1. Background

Memory analysis is a vital component of modern digital forensics. Disk drives are increasing in size thus creating vast data sets to parse and increased acquisition times. Memory analysis allows the MSCERT to quickly identify what is running on a suspect system. Malware and other unwanted software is easily identified through Responder and Digital DNA (explained in the following sections). Many attackers are leveraging process injection and in-memory-only attacks. This allows them to evade anti-virus and traditional disk based forensics.

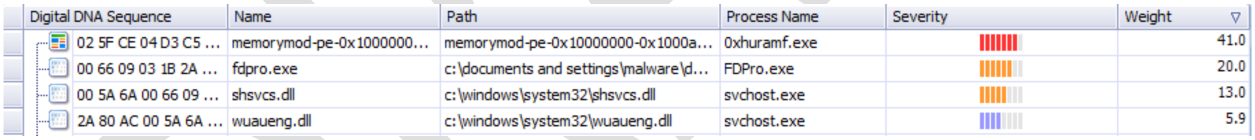
HBGary Responder Professional is the approved tool for conducting memory analysis. Once physical memory has been acquired using the approved method the analyst must then extract case relevant data.

3.2. Responder Pro Features

3.2.1. Digital DNA

Digital DNA is an HBGary proprietary feature that allows an analyst to rapidly identify memory modules that require further investigation. A memory module is a piece of executable code that runs on a system and performs a function. For example, when the process svchost.exe is running it has multiple modules running within its context. The svchost.exe itself is a memory module as are the supporting libraries such as shsvcs.dll.

Figure 5

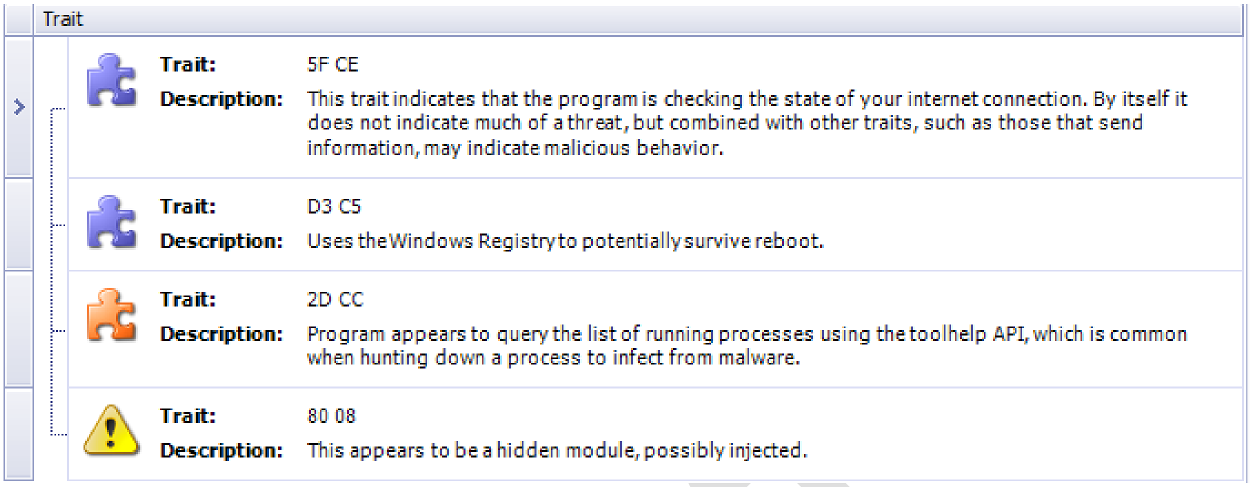


Digital DNA Sequence	Name	Path	Process Name	Severity	Weight ▽
02 5F CE 04 D3 C5 …	memorymod-pe-0x1000000…	memorymod-pe-0x10000000-0x1000a…	0xhuramf.exe	🟥	41.0
00 66 09 03 1B 2A …	fdpro.exe	c:\documents and settings\malware\d…	FDPro.exe	🟧	20.0
00 5A 6A 00 66 09 …	shsvcs.dll	c:\windows\system32\shsvcs.dll	svchost.exe	🟧	13.0
2A 80 AC 00 5A 6A …	wuaueng.dll	c:\windows\system32\wuaueng.dll	svchost.exe	🟦	5.9

Digital DNA ranks memory modules based on weight. The weight is derived from an aggregation of individual traits scores. Figure 5 demonstrates the color scheme associated with a module weight. Red is the most suspicious, orange is less suspicious, and blue is the least suspicious.

A trait is a capability of a memory module. If the module uses TCP/IP, a low score is assigned to that trait whereas if the module shows signs of obfuscation that trait would receive a high score. The combined traits define the final weight of the module. Figure 6 displays shows example traits.

Figure 6



Trait

Trait: 5F CE
Description: This trait indicates that the program is checking the state of your internet connection. By itself it does not indicate much of a threat, but combined with other traits, such as those that send information, may indicate malicious behavior.

Trait: D3 C5
Description: Uses the Windows Registry to potentially survive reboot.

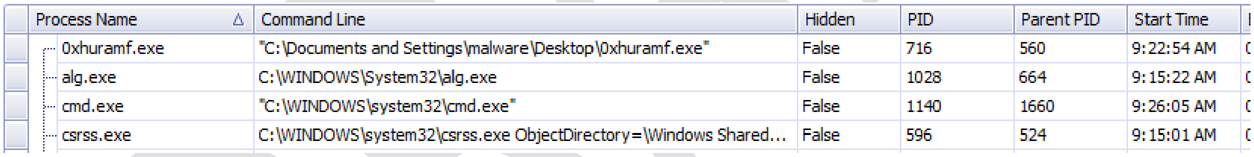
Trait: 2D CC
Description: Program appears to query the list of running processes using the toolhelp API, which is common when hunting down a process to infect from malware.

Trait: 80 08
Description: This appears to be a hidden module, possibly injected.

3.2.2. Process Listing

Responder Pro has the ability to get an accurate listing of running processes on a suspect system through its use of off-line memory analysis. Processes can be hidden from users and analysts through rootkit techniques that often defeat real-time analysis of a system. Responder finds hidden and non-hidden process and displays them as demonstrated in Figure 7. Responder also details what command-line arguments were used and what the parent process is.

Figure 7

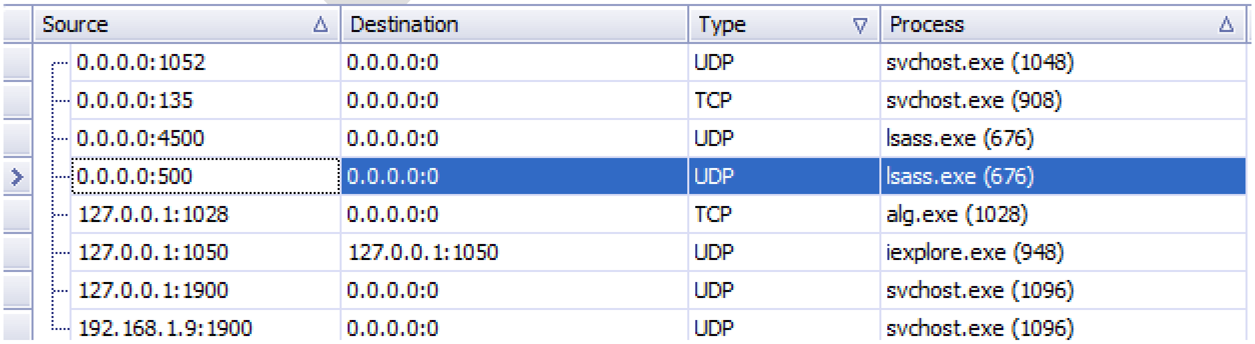


Process Name △	Command Line	Hidden	PID	Parent PID	Start Time	I
0xhuramf.exe	"C:\Documents and Settings\malware\Desktop\0xhuramf.exe"	False	716	560	9:22:54 AM	(
alg.exe	C:\WINDOWS\System32\alg.exe	False	1028	664	9:15:22 AM	(
cmd.exe	"C:\WINDOWS\system32\cmd.exe"	False	1140	1660	9:26:05 AM	(
csrss.exe	C:\WINDOWS\system32\csrss.exe ObjectDirectory=\Windows Shared...	False	596	524	9:15:01 AM	(

3.2.3. Network Sockets

Responder Pro can list all listening and established network connections on a suspect system. Rootkit technology can hide network related indicators similar to how processes may be hidden. Figure 8 shows network sockets on a system.

Figure 8

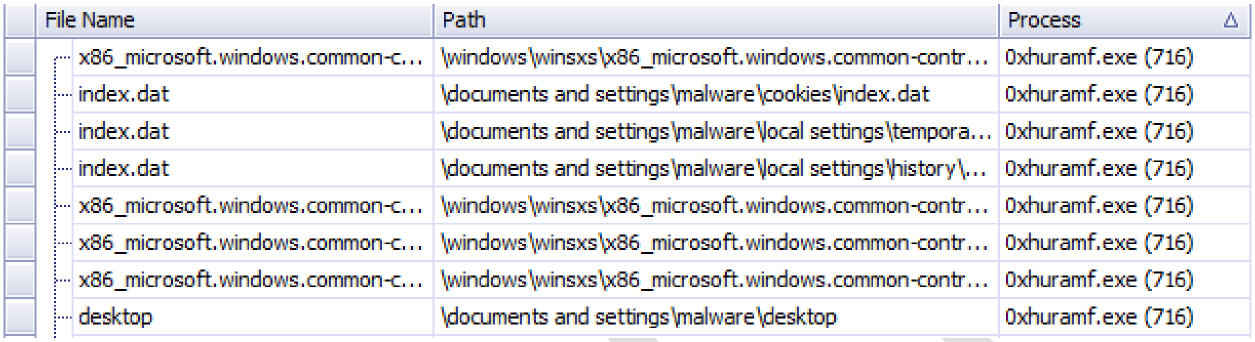


Source △	Destination	Type ▽	Process △
0.0.0.0:1052	0.0.0.0:0	UDP	svchost.exe (1048)
0.0.0.0:135	0.0.0.0:0	TCP	svchost.exe (908)
0.0.0.0:4500	0.0.0.0:0	UDP	lsass.exe (676)
0.0.0.0:500	0.0.0.0:0	UDP	lsass.exe (676)
127.0.0.1:1028	0.0.0.0:0	TCP	alg.exe (1028)
127.0.0.1:1050	127.0.0.1:1050	UDP	iexplore.exe (948)
127.0.0.1:1900	0.0.0.0:0	UDP	svchost.exe (1096)
192.168.1.9:1900	0.0.0.0:0	UDP	svchost.exe (1096)

3.2.4. File Handles

An analyst can view all files that a target process has open at the time of memory acquisition. File handles can provide evidence that a suspect process is performing an action that affects the disk drive such as logging keystrokes to a file. Figure 9 demonstrates how a suspect process can be identified as using HTTP as indicated by the open file handles to index.dat.

Figure 9

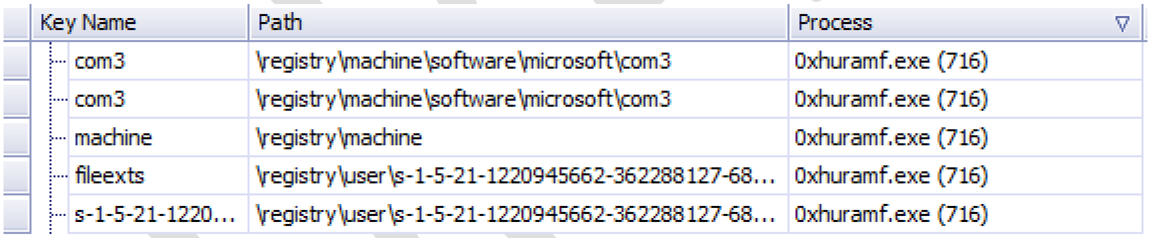


File Name	Path	Process
x86_microsoft.windows.common-c...	\windows\winsxs\x86_microsoft.windows.common-contr...	0xhuramf.exe (716)
index.dat	\documents and settings\malware\cookies\index.dat	0xhuramf.exe (716)
index.dat	\documents and settings\malware\local settings\tempora...	0xhuramf.exe (716)
index.dat	\documents and settings\malware\local settings\history\...	0xhuramf.exe (716)
x86_microsoft.windows.common-c...	\windows\winsxs\x86_microsoft.windows.common-contr...	0xhuramf.exe (716)
x86_microsoft.windows.common-c...	\windows\winsxs\x86_microsoft.windows.common-contr...	0xhuramf.exe (716)
x86_microsoft.windows.common-c...	\windows\winsxs\x86_microsoft.windows.common-contr...	0xhuramf.exe (716)
desktop	\documents and settings\malware\desktop	0xhuramf.exe (716)

3.2.5. Registry Keys

Responder Pro displays all registry keys that a target process has open at the time of memory acquisition. Registry keys can give clues about a process's capabilities such as persistence across reboots. Figure 10 shows an example of open registry keys by process ID 716.

Figure 10

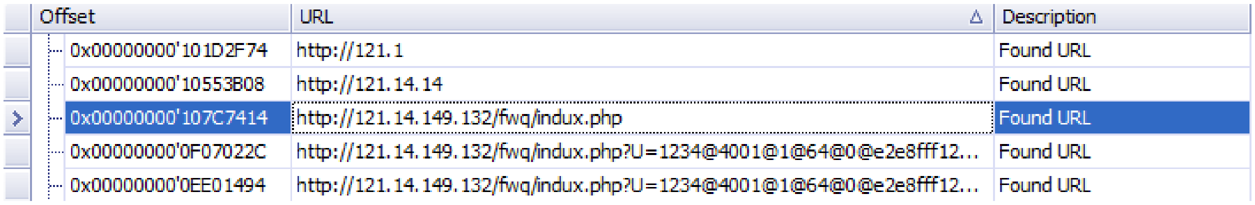


Key Name	Path	Process
com3	\registry\machine\software\microsoft\com3	0xhuramf.exe (716)
com3	\registry\machine\software\microsoft\com3	0xhuramf.exe (716)
machine	\registry\machine	0xhuramf.exe (716)
fileexts	\registry\user\s-1-5-21-1220945662-362288127-68...	0xhuramf.exe (716)
s-1-5-21-1220...	\registry\user\s-1-5-21-1220945662-362288127-68...	0xhuramf.exe (716)

3.2.6. Internet History

Responder Pro discovers all URLs that exist in memory. This is independent of browser usage. For example, if malware makes a direct API call to download a next stage downloader via HTTP then a URL will exist in memory but not in the index.dat file. This is a significant advantage over traditional disk based forensic approaches. Figure 11 provides an example of the Internet History view.

Figure 11

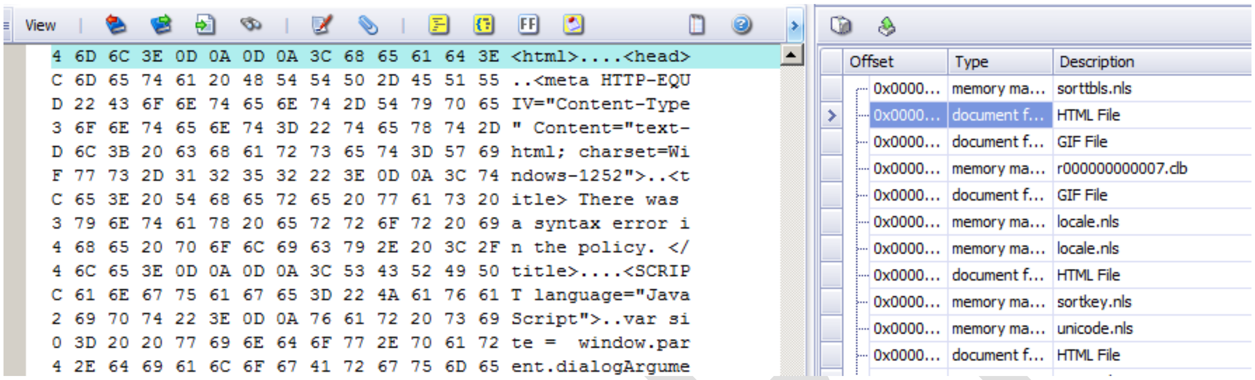


Offset	URL	Description
0x00000000'101D2F74	http://121.1	Found URL
0x00000000'10553B08	http://121.14.14	Found URL
0x00000000'107C7414	http://121.14.149.132/fwq/indux.php	Found URL
0x00000000'0F07022C	http://121.14.149.132/fwq/indux.php?U=1234@4001@1@64@0@e2e8fff12...	Found URL
0x00000000'0EE01494	http://121.14.149.132/fwq/indux.php?U=1234@4001@1@64@0@e2e8fff12...	Found URL

3.2.7. File Fragments

Responder Pro has a “Documents and Messages” section that lists discovered file fragments. File types such as HTML can be recovered and examined. This information can provide answers to infection vector questions such as drive-by downloads via hidden javascript. Figure 12 shows a recovered HTML page in the left pane and a listing of files in the right pane.

Figure 12



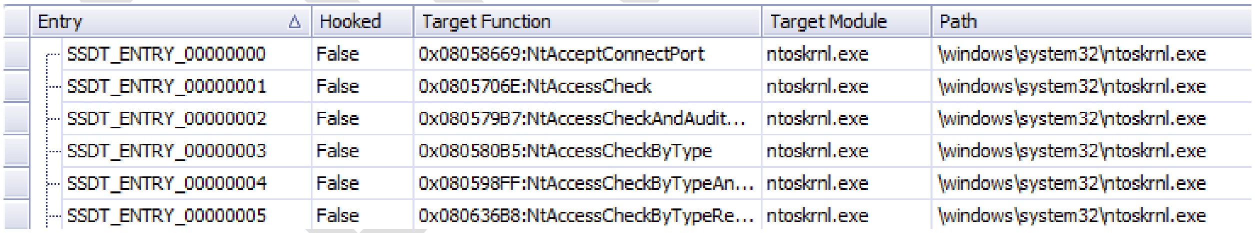
```
4 6D 6C 3E 0D 0A 0D 0A 3C 68 65 61 64 3E <html>....<head>
C 6D 65 74 61 20 48 54 54 50 2D 45 51 55 ..<meta HTTP-EQU
D 22 43 6F 6E 74 65 6E 74 2D 54 79 70 65 IV="Content-Type
3 6F 6E 74 65 6E 74 3D 22 74 65 78 74 2D " Content="text-
D 6C 3B 20 63 68 61 72 73 65 74 3D 57 69 html; charset=Wi
F 77 73 2D 31 32 35 32 22 3E 0D 0A 3C 74 ndows-1252">..<t
C 65 3E 20 54 68 65 72 65 20 77 61 73 20 itle> There was
3 79 6E 74 61 78 20 65 72 72 6F 72 20 69 a syntax error i
4 68 65 20 70 6F 6C 69 63 79 2E 20 3C 2F n the policy. </
4 6C 65 3E 0D 0A 0D 0A 3C 53 43 52 49 50 title>....<SCRIP
C 61 6E 67 75 61 67 65 3D 22 4A 61 76 61 T language="Java
2 69 70 74 22 3E 0D 0A 76 61 72 20 73 69 Script">..var si
0 3D 20 20 77 69 6E 64 6F 77 2E 70 61 72 te =  window.par
4 2E 64 69 61 6C 6F 67 41 72 67 75 6D 65 ent.dialogArgume
```

Offset	Type	Description
0x0000...	memory ma...	sorttbls.nls
0x0000...	document f...	HTML File
0x0000...	document f...	GIF File
0x0000...	memory ma...	r000000000007.clb
0x0000...	document f...	GIF File
0x0000...	memory ma...	locale.nls
0x0000...	memory ma...	locale.nls
0x0000...	document f...	HTML File
0x0000...	memory ma...	sortkey.nls
0x0000...	memory ma...	unicode.nls
0x0000...	document f...	HTML File

3.2.8. System Service Descriptor Table (SSDT)

The SSDT serves an important and system wide function on a Windows system. The SSDT tells the operating system where to find key system functions in memory. Kernel level malware will often hook this table in order to hide itself and other components of the malware. Reliably finding these hooks on a running system is challenging. Off-line memory analysis via Responder Pro makes finding these highly dangerous hooks easy. Any discovered hooks are enumerated and displayed to the analyst. Responder lists the hooked function and the hooking component. Figure 13 displays an example of the SSDT view.

Figure 13

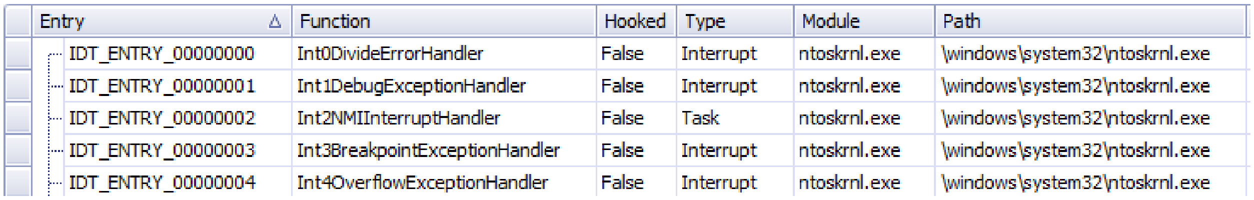


Entry	Hooked	Target Function	Target Module	Path
SSDT_ENTRY_00000000	False	0x08058669:NtAcceptConnectPort	ntoskrnl.exe	\windows\system32\ntoskrnl.exe
SSDT_ENTRY_00000001	False	0x0805706E:NtAccessCheck	ntoskrnl.exe	\windows\system32\ntoskrnl.exe
SSDT_ENTRY_00000002	False	0x080579B7:NtAccessCheckAndAudit...	ntoskrnl.exe	\windows\system32\ntoskrnl.exe
SSDT_ENTRY_00000003	False	0x080580B5:NtAccessCheckByType	ntoskrnl.exe	\windows\system32\ntoskrnl.exe
SSDT_ENTRY_00000004	False	0x080598FF:NtAccessCheckByTypeAn...	ntoskrnl.exe	\windows\system32\ntoskrnl.exe
SSDT_ENTRY_00000005	False	0x080636B8:NtAccessCheckByTypeRe...	ntoskrnl.exe	\windows\system32\ntoskrnl.exe

3.2.9. Interrupt Descriptor Table (IDT)

The IDT is a low level mechanism in the Windows operating system that handles the interrupts that are sent to the CPU. This is enticing place for a malware author to hook as it affects the entire system. For example all keystrokes can be logged by hooking the appropriate interrupt. The target function and the hooking mechanism are displayed to the analyst. Figure 14 displays an example of the IDT as displayed in Responder Pro.

Figure 14

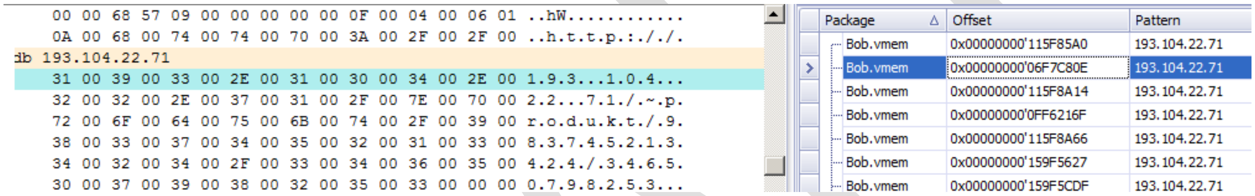


Entry	Function	Hooked	Type	Module	Path
IDT_ENTRY_00000000	Int0DivideErrorHandler	False	Interrupt	ntoskrnl.exe	\windows\system32\ntoskrnl.exe
IDT_ENTRY_00000001	Int1DebugExceptionHandler	False	Interrupt	ntoskrnl.exe	\windows\system32\ntoskrnl.exe
IDT_ENTRY_00000002	Int2NMIInterruptHandler	False	Task	ntoskrnl.exe	\windows\system32\ntoskrnl.exe
IDT_ENTRY_00000003	Int3BreakpointExceptionHandler	False	Interrupt	ntoskrnl.exe	\windows\system32\ntoskrnl.exe
IDT_ENTRY_00000004	Int4OverflowExceptionHandler	False	Interrupt	ntoskrnl.exe	\windows\system32\ntoskrnl.exe

3.2.10. Pattern Matches

Responder Pro provides the analyst with an automated way of searching a memory image for predetermined strings. The analyst creates a carriage return delimited file of ASCII strings prior to importing a memory image. When Responder Pro imports the memory image it locates all instances of each string in memory and places the results in the "Pattern Matches" folder. In Figure 15 the pattern matches are displayed in the right pane. The location in memory is displayed in the left pane when the pattern match is double-clicked.

Figure 15



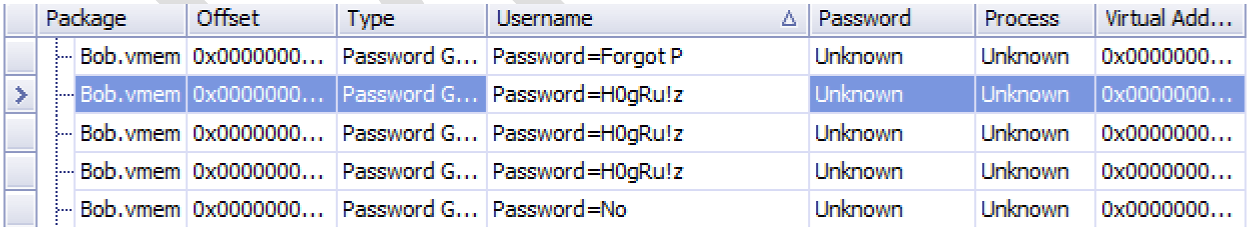
```
    00 00 68 57 09 00 00 00 00 00 0F 00 04 00 06 01 ..hW............
    0A 00 68 00 74 00 74 00 70 00 3A 00 2F 00 2F 00 ..h.t.t.p.:././.
db 193.104.22.71
    31 00 39 00 33 00 2E 00 31 00 30 00 34 00 2E 00 1.9.3...1.0.4...
    32 00 32 00 2E 00 37 00 31 00 2F 00 7E 00 70 00 2.2...7.1./.~.p.
    72 00 6F 00 64 00 75 00 6B 00 74 00 2F 00 39 00 r.o.d.u.k.t./.9.
    38 00 33 00 37 00 34 00 35 00 32 00 31 00 33 00 8.3.7.4.5.2.1.3.
    34 00 32 00 34 00 2F 00 33 00 34 00 36 00 35 00 4.2.4./.3.4.6.5.
    30 00 37 00 39 00 38 00 32 00 35 00 33 00 00 00 0.7.9.8.2.5.3...
```

Package	Offset	Pattern
Bob.vmem	0x00000000'115F85A0	193.104.22.71
Bob.vmem	0x00000000'06F7C80E	193.104.22.71
Bob.vmem	0x00000000'115F8A14	193.104.22.71
Bob.vmem	0x00000000'0FF6216F	193.104.22.71
Bob.vmem	0x00000000'115F8A66	193.104.22.71
Bob.vmem	0x00000000'159F5627	193.104.22.71
Bob.vmem	0x00000000'159F5CDF	193.104.22.71

3.2.11. Keys and Passwords

Responder performs a pattern match for common key and password strings across an imported memory snapshot. This feature searches for strings such as "Password=" so only blatantly obvious passwords will be discovered with this method. Figure 16 displays a listing of recovered passwords from a memory image.

Figure 16

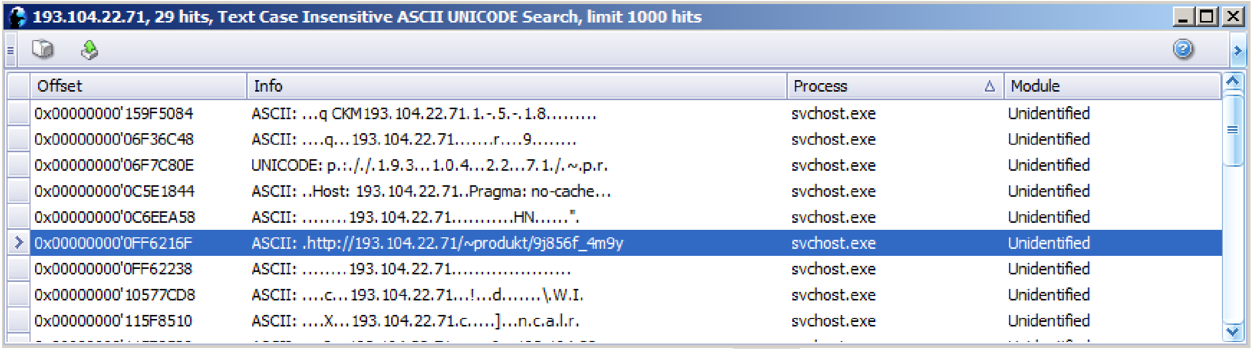


Package	Offset	Type	Username	Password	Process	Virtual Add...
Bob.vmem	0x0000000...	Password G...	Password=Forgot P	Unknown	Unknown	0x0000000...
Bob.vmem	0x0000000...	Password G...	Password=H0gRu!z	Unknown	Unknown	0x0000000...
Bob.vmem	0x0000000...	Password G...	Password=H0gRu!z	Unknown	Unknown	0x0000000...
Bob.vmem	0x0000000...	Password G...	Password=H0gRu!z	Unknown	Unknown	0x0000000...
Bob.vmem	0x0000000...	Password G...	Password=No	Unknown	Unknown	0x0000000...

3.2.12. String Searching

An analyst can manually search a memory image for ASCII and Unicode strings. This feature allows an analyst to associate a string with a process and potentially a module within that process. For example if a system is known to be communicating with an external IP address, the IP address can be searched for across all of virtual memory. The IP address "193.104.22.71" was searched for and displayed in Figure 16. The IP address was associated with the svchost.exe process.

Figure 16



193.104.22.71, 29 hits, Text Case Insensitive ASCII UNICODE Search, limit 1000 hits

Offset	Info	Process	Module
0x00000000'159F5084	ASCII: ...q CKM193.104.22.71.1.-.5.-.1.8.........	svchost.exe	Unidentified
0x00000000'06F36C48	ASCII: ....q...193.104.22.71.......r....9........	svchost.exe	Unidentified
0x00000000'06F7C80E	UNICODE: p.:./././.1.9.3...1.0.4...2.2...7.1./.~.p.r.	svchost.exe	Unidentified
0x00000000'0C5E1844	ASCII: ..Host: 193.104.22.71..Pragma: no-cache...	svchost.exe	Unidentified
0x00000000'0C6EEA58	ASCII: ....193.104.22.71..........HN......".	svchost.exe	Unidentified
0x00000000'0FF6216F	ASCII: .http://193.104.22.71/~produkt/9j856f_4m9y	svchost.exe	Unidentified
0x00000000'0FF62238	ASCII: ....193.104.22.71....................	svchost.exe	Unidentified
0x00000000'10577CD8	ASCII: ....c...193.104.22.71...!...d.......\.W.I.	svchost.exe	Unidentified
0x00000000'115F8510	ASCII: ....X...193.104.22.71.c.....]...n.c.a.l.r.	svchost.exe	Unidentified

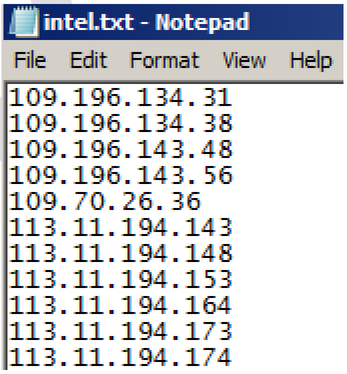
3.3. Case Studies

3.3.1. Help Desk Alert

Background: MSCERT receives a ticket from the desktop support team. A user "Bob" reported opening a link to a PDF document embedded in an email from a coworker. The user also noticed fraudulent transactions on their bank statement shortly after the incident. MSCERT is tasked with determining if the system has been compromised and to gather any actionable intelligence that can be added to the enterprise security infrastructure.

Response: MSCERT begins the investigation with a volatile memory acquisition as described in section 2.4.1 of this document. Once the memory snapshot is acquired and placed on the evidence drive the investigation begins.

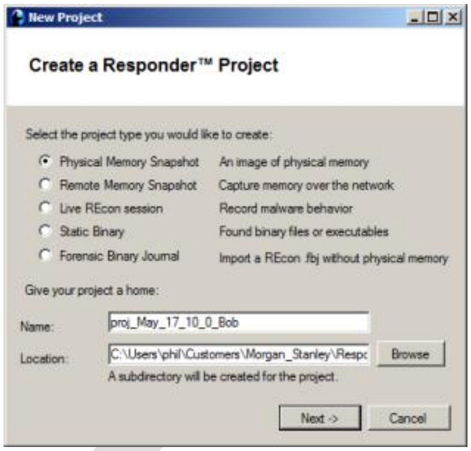
1) Create a CR delimited text file with known case data and open source intelligence
   a) Obtain ZeuS block list from https://zeustracker.abuse.ch/blocklist.php
   b) Place data in a text file called "intel.txt"



intel.txt - Notepad

```
109.196.134.31
109.196.134.38
109.196.143.48
109.196.143.56
109.70.26.36
113.11.194.143
113.11.194.148
113.11.194.153
113.11.194.164
113.11.194.173
113.11.194.174
```

2) Create a New Project
   a) Start Responder Pro
   b) Select "File→New"
   c) Select "Physical Memory Snapshot"
   d) Name the project uniquely
   e) Save the project to the evidence drive
   f) Enter the appropriate case tracking data



New Project

Create a Responder™ Project

Select the project type you would like to create:

● Physical Memory Snapshot An image of physical memory
○ Remote Memory Snapshot Capture memory over the network
○ Live REcon session Record malware behavior
○ Static Binary Found binary files or executables
○ Forensic Binary Journal Import a REcon .fbj without physical memory

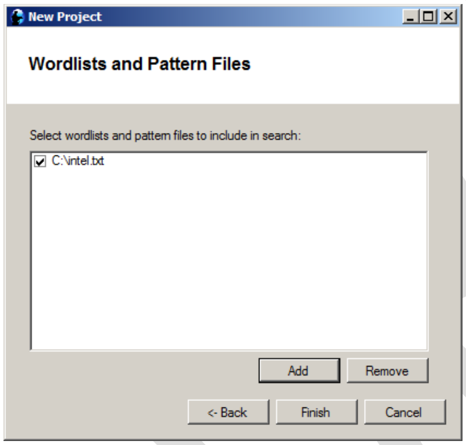
Give your project a home:

Name: proj_May_17_10_0_Bob

Location: C:\Users\phil\Customers\Morgan_Stanley\Respc

A subdirectory will be created for the project.

3) Select the intel.txt by using the checkbox



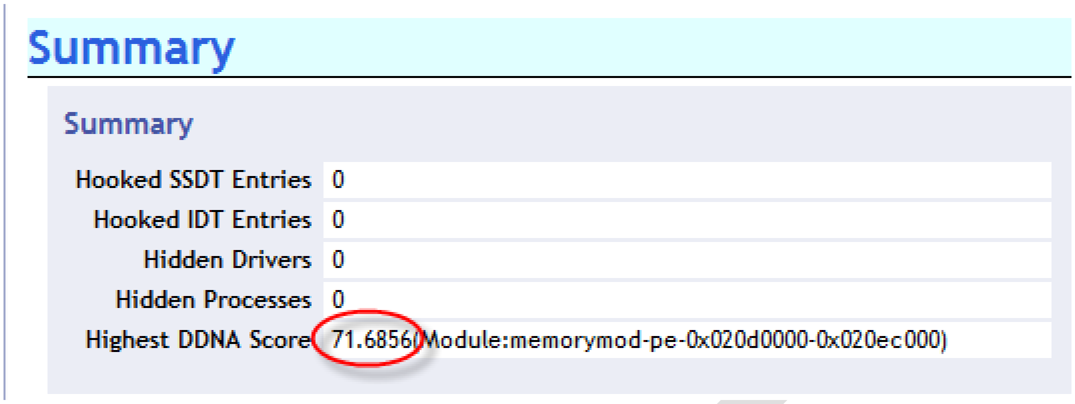
New Project

Wordlists and Pattern Files

Select wordlists and pattern files to include in search:

☑ C:\intel.txt

4) Examine the "Report" tab in the right pane
   a) Observe the high DDNA score in the Summary section for the oddly named module
   b) Notice there are no SSDT hooks, IDT hooks, hidden drivers/processes



Summary

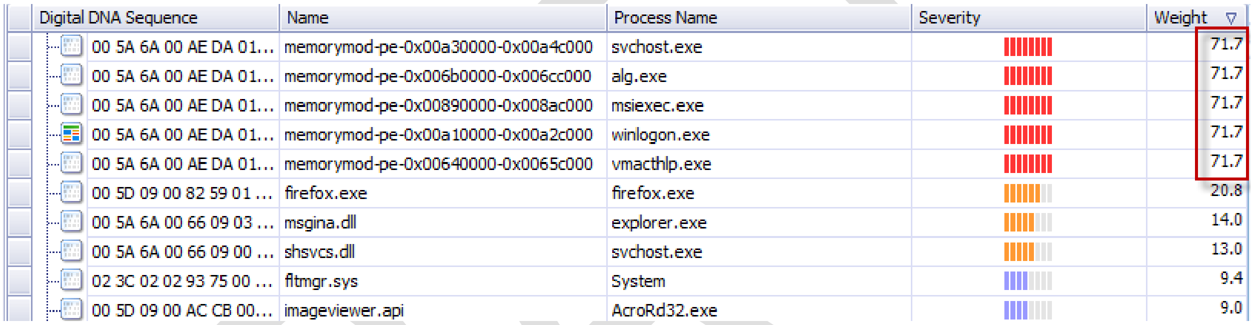
Summary

Hooked SSDT Entries	0
Hooked IDT Entries	0
Hidden Drivers	0
Hidden Processes	0
Highest DDNA Score	71.6856(Module:memorymod-pe-0x020d0000-0x020ec000)

5) Examine the "Digital DNA" tab

   a) Follow up on findings in step 4

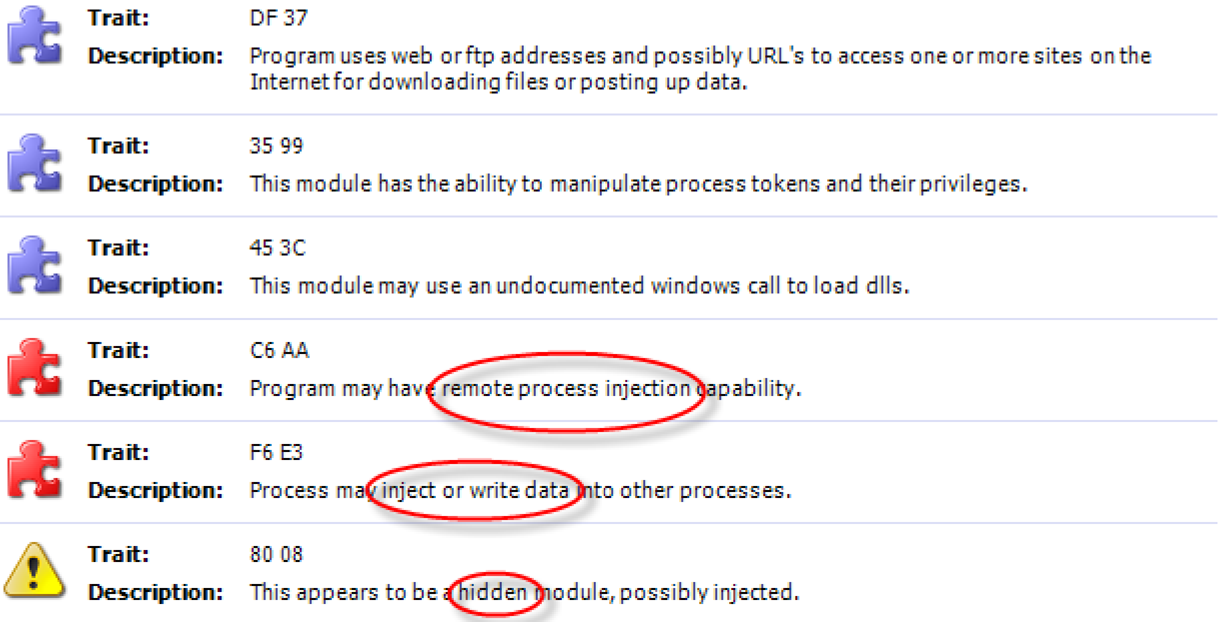
   b) Observe the numerous modules with the same score



Digital DNA Sequence	Name	Process Name	Severity	Weight ▽
00 5A 6A 00 AE DA 01…	memorymod-pe-0x00a30000-0x00a4c000	svchost.exe	🟥🟥🟥🟥🟥	71.7
00 5A 6A 00 AE DA 01…	memorymod-pe-0x006b0000-0x006cc000	alg.exe	🟥🟥🟥🟥🟥	71.7
00 5A 6A 00 AE DA 01…	memorymod-pe-0x00890000-0x008ac000	msiexec.exe	🟥🟥🟥🟥🟥	71.7
00 5A 6A 00 AE DA 01…	memorymod-pe-0x00a10000-0x00a2c000	winlogon.exe	🟥🟥🟥🟥🟥	71.7
00 5A 6A 00 AE DA 01…	memorymod-pe-0x00640000-0x0065c000	vmacthlp.exe	🟥🟥🟥🟥🟥	71.7
00 5D 09 00 82 59 01 …	firefox.exe	firefox.exe	🟧🟧🟧	20.8
00 5A 6A 00 66 09 03 …	msgina.dll	explorer.exe	🟧🟧🟧	14.0
00 5A 6A 00 66 09 00 …	shsvcs.dll	svchost.exe	🟧🟧🟧	13.0
02 3C 02 02 93 75 00 …	fltmgr.sys	System	🟦🟦	9.4
00 5D 09 00 AC CB 00…	imageviewer.api	AcroRd32.exe	🟦🟦	9.0

   c) The oddly named module names indicate injected code. There is no path to the filesystem for this code.

   d) Right-click one of the modules with a "red" status and select "View Traits"



Trait: DF 37
Description: Program uses web or ftp addresses and possibly URL's to access one or more sites on the Internet for downloading files or posting up data.

Trait: 35 99
Description: This module has the ability to manipulate process tokens and their privileges.

Trait: 45 3C
Description: This module may use an undocumented windows call to load dlls.

Trait: C6 AA
Description: Program may have remote process injection capability.

Trait: F6 E3
Description: Process may inject or write data into other processes.

Trait: 80 08
Description: This appears to be a hidden module, possibly injected.

e) Observe the injection capabilities of the module

6) Examine Module Strings

a) Right-click on the module in the “Digtial DNA” tab and select “View Strings”

b) Sort the strings by the “Offset” column to view the strings in the order in which they were found in the module.

c) Observe network related strings

i) Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)

ii) HTTP/1.1

iii) Connection: close

iv) urlmon.dll

v) ObtainUserAgentString

vi) %u.%u.%u.%u (potential IP address)

vii) %S://%S:%S@%u.%u.%u.%u:%u/

d) Possible mutexes which are used by malware authors to determine if a system has been compromised previously

i) _H_64AD0625_

ii) __SYSTEM__64AD0625__

e) Protected storage access

i) PStoreCreateInstance

ii) pstorec.dll

iii) Protected Storage:

f) Registry Access

i) software\microsoft\internet explorer\phishingfilter

ii) \Registry\Machine\System\CurrentControlSet\Control\SecurePipeServers\

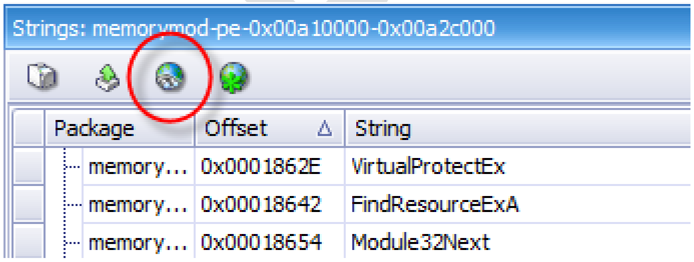
iii) software\microsoft\windows\currentversion\explorer

iv) System\CurrentControlSet\Services

v) Software\Microsoft\Windows NT\CurrentVersion\Svchost

g) Perform search of all strings for “\” character to easily identify registry paths or file paths

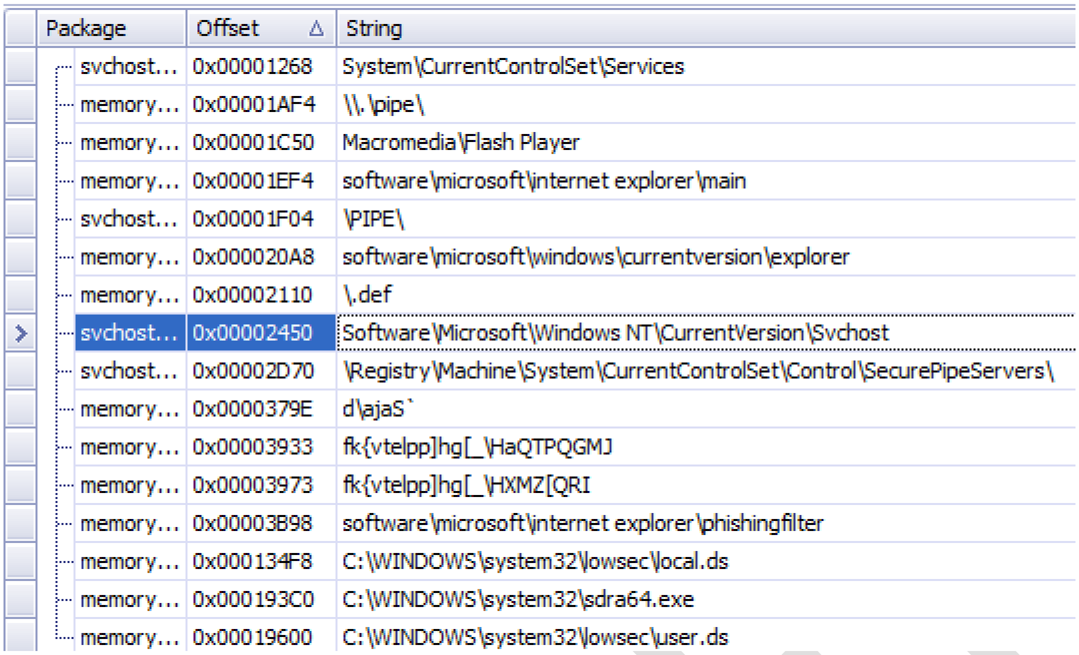
i) Click on the eyeball on globe icon



Strings: memorymod-pe-0x00a10000-0x00a2c000

Package	Offset △	String
memory...	0x0001862E	VirtualProtectEx
memory...	0x00018642	FindResourceExA
memory...	0x00018654	Module32Next

ii) Conduct a substring search for “\”



Package	Offset △	String
svchost…	0x00001268	System\CurrentControlSet\Services
memory…	0x00001AF4	\\.\pipe\
memory…	0x00001C50	Macromedia\Flash Player
memory…	0x00001EF4	software\microsoft\internet explorer\main
svchost…	0x00001F04	\PIPE\
memory…	0x000020A8	software\microsoft\windows\currentversion\explorer
memory…	0x00002110	\.def
svchost…	0x00002450	Software\Microsoft\Windows NT\CurrentVersion\Svchost
svchost…	0x00002D70	\Registry\Machine\System\CurrentControlSet\Control\SecurePipeServers\
memory…	0x0000379E	d\ajaS`
memory…	0x00003933	fk{vtelpp]hg[_\HaQTPQGMJ
memory…	0x00003973	fk{vtelpp]hg[_\HXMZ[QRI
memory…	0x00003B98	software\microsoft\internet explorer\phishingfilter
memory…	0x000134F8	C:\WINDOWS\system32\lowsec\local.ds
memory…	0x000193C0	C:\WINDOWS\system32\sdra64.exe
memory…	0x00019600	C:\WINDOWS\system32\lowsec\user.ds

h) Note the file paths

i) C:\WINDOWS\system32\lowsec\local.ds

ii) C:\WINDOWS\system32\sdra64.exe

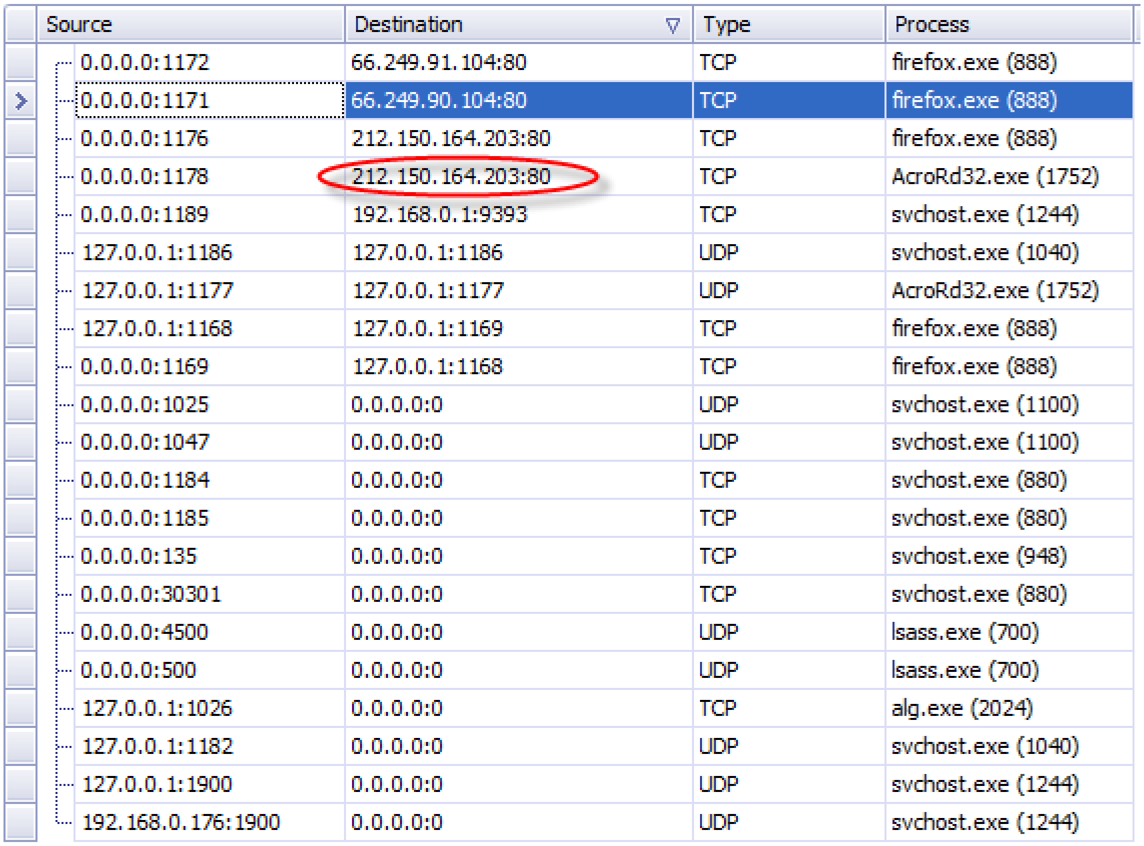
iii) C:\WINDOWS\system32\lowsec\user.ds

7) Examine network connections

a) In the “Objects” tab select “All Open Network Sockets”

b) Sort by the “Destination” column

c) Observe the suspicious network connection from the Adobe process

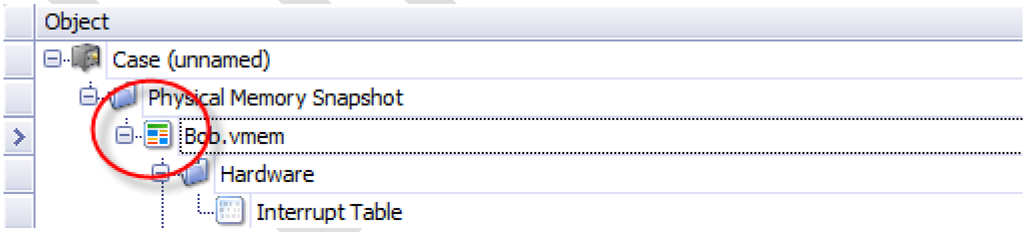


Source	Destination	Type	Process
0.0.0.0:1172	66.249.91.104:80	TCP	firefox.exe (888)
0.0.0.0:1171	66.249.90.104:80	TCP	firefox.exe (888)
0.0.0.0:1176	212.150.164.203:80	TCP	firefox.exe (888)
0.0.0.0:1178	212.150.164.203:80	TCP	AcroRd32.exe (1752)
0.0.0.0:1189	192.168.0.1:9393	TCP	svchost.exe (1244)
127.0.0.1:1186	127.0.0.1:1186	UDP	svchost.exe (1040)
127.0.0.1:1177	127.0.0.1:1177	UDP	AcroRd32.exe (1752)
127.0.0.1:1168	127.0.0.1:1169	TCP	firefox.exe (888)
0.0.0.0:1169	127.0.0.1:1168	TCP	firefox.exe (888)
0.0.0.0:1025	0.0.0.0:0	UDP	svchost.exe (1100)
0.0.0.0:1047	0.0.0.0:0	UDP	svchost.exe (1100)
0.0.0.0:1184	0.0.0.0:0	TCP	svchost.exe (880)
0.0.0.0:1185	0.0.0.0:0	TCP	svchost.exe (880)
0.0.0.0:135	0.0.0.0:0	TCP	svchost.exe (948)
0.0.0.0:30301	0.0.0.0:0	TCP	svchost.exe (880)
0.0.0.0:4500	0.0.0.0:0	UDP	lsass.exe (700)
0.0.0.0:500	0.0.0.0:0	UDP	lsass.exe (700)
127.0.0.1:1026	0.0.0.0:0	TCP	alg.exe (2024)
127.0.0.1:1182	0.0.0.0:0	UDP	svchost.exe (1040)
127.0.0.1:1900	0.0.0.0:0	UDP	svchost.exe (1244)
192.168.0.176:1900	0.0.0.0:0	UDP	svchost.exe (1244)

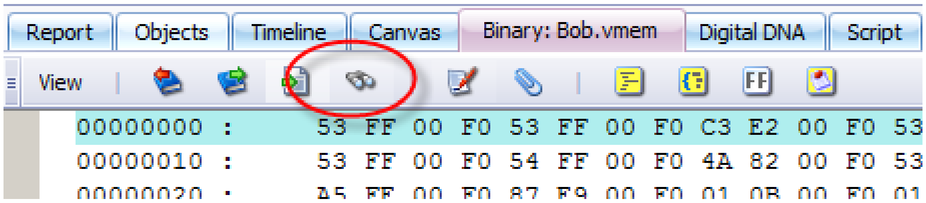
d) Also make note of the Firefox IP addresses for later investigation

8) Search for instances of the "212.150.164.203" string in memory

a) Double click the memory icon in the "Objects" tab

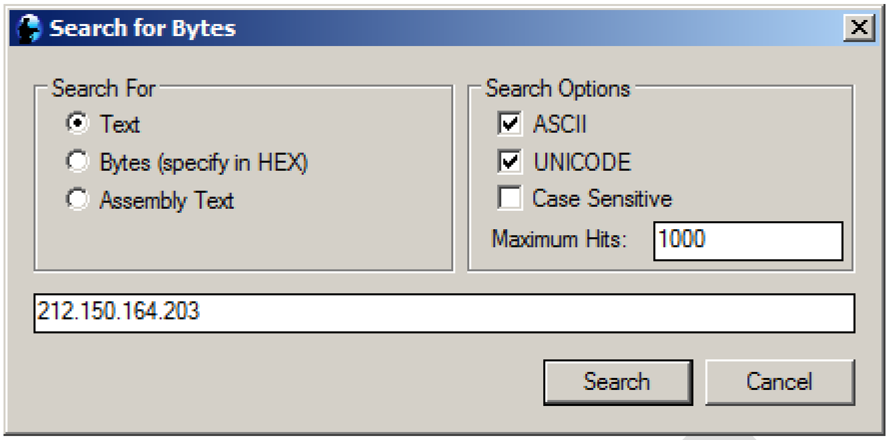


b) Use the binoculars icon to start a new search



```
00000000 :    53 FF 00 F0 53 FF 00 F0 C3 E2 00 F0 53
00000010 :    53 FF 00 F0 54 FF 00 F0 4A 82 00 F0 53
00000020 :    A5 FF 00 F0 87 F9 00 F0 01 0B 00 F0 01
```

c) Search for the "212.150.164.203" IP address in memory in both ASCII and Unicode



Search for Bytes

Search For

● Text

○ Bytes (specify in HEX)

○ Assembly Text

Search Options

☑ ASCII

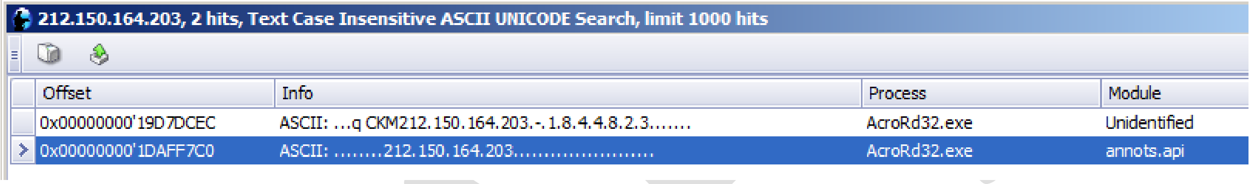
☑ UNICODE

□ Case Sensitive

Maximum Hits: 1000

212.150.164.203

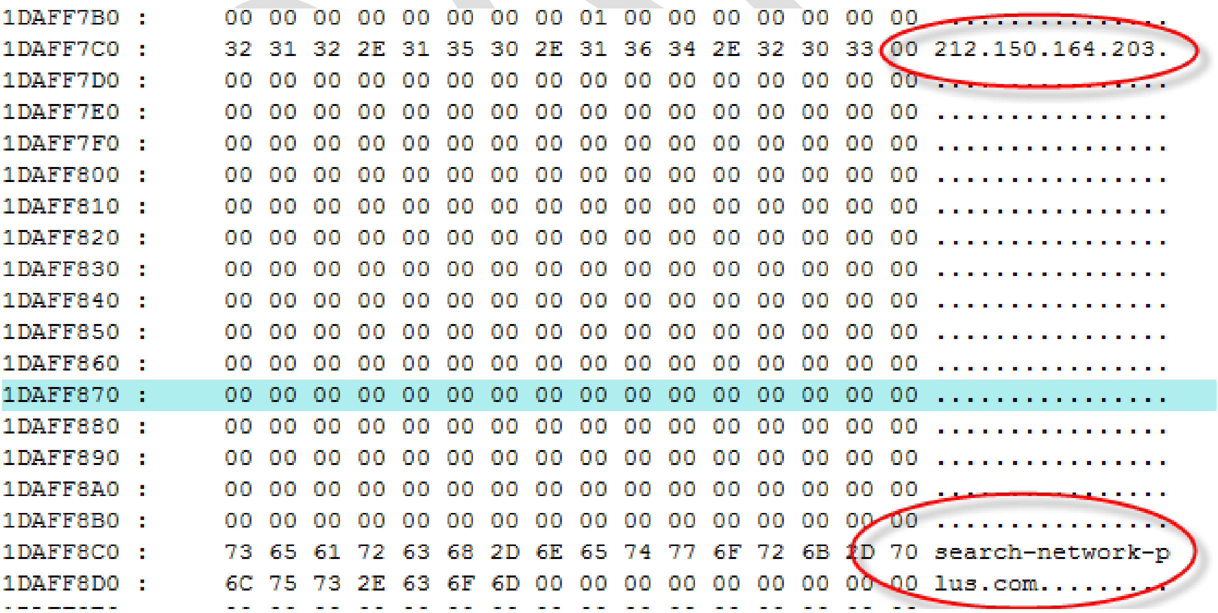
d) Double-click on the second search hit which is in the annots.api module



212.150.164.203, 2 hits, Text Case Insensitive ASCII UNICODE Search, limit 1000 hits

Offset	Info	Process	Module
0x00000000'19D7DCEC	ASCII: ...q CKM212.150.164.203.-.1.8.4.4.8.2.3.......	AcroRd32.exe	Unidentified
0x00000000'1DAFF7C0	ASCII: ....212.150.164.203......................	AcroRd32.exe	annots.api

e) The left pane displays the search hit in memory and a domain name is observed in close proximity to the IP address.



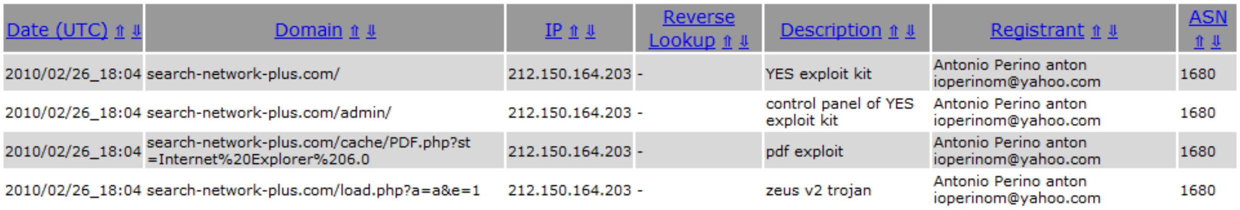
```
1DAFF7B0 :   00 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00 ................
1DAFF7C0 :   32 31 32 2E 31 35 30 2E 31 36 34 2E 32 30 33 00 212.150.164.203.
1DAFF7D0 :   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
1DAFF7E0 :   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
1DAFF7F0 :   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
1DAFF800 :   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
1DAFF810 :   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
1DAFF820 :   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
1DAFF830 :   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
1DAFF840 :   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
1DAFF850 :   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
1DAFF860 :   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
1DAFF870 :   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
1DAFF880 :   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
1DAFF890 :   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
1DAFF8A0 :   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
1DAFF8B0 :   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
1DAFF8C0 :   73 65 61 72 63 68 2D 6E 65 74 77 6F 72 6B 2D 70 search-network-p
1DAFF8D0 :   6C 75 73 2E 63 6F 6D 00 00 00 00 00 00 00 00 00 lus.com.........
```

9) Perform open source intelligence

a) Do Google search for the recovered domain name

b) Use reputable sites to extract intelligence

c) Observe the following Google hit: http://www.malwaredomainlist.com/mdl.php?search=search-network-plus.com&inactive=on



Date (UTC) ⇑ ⇓	Domain ⇑ ⇓	IP ⇑ ⇓	Reverse Lookup ⇑ ⇓	Description ⇑ ⇓	Registrant ⇑ ⇓	ASN ⇑ ⇓
2010/02/26_18:04	search-network-plus.com/	212.150.164.203	-	YES exploit kit	Antonio Perino anton ioperinom@yahoo.com	1680
2010/02/26_18:04	search-network-plus.com/admin/	212.150.164.203	-	control panel of YES exploit kit	Antonio Perino anton ioperinom@yahoo.com	1680
2010/02/26_18:04	search-network-plus.com/cache/PDF.php?st =Internet%20Explorer%206.0	212.150.164.203	-	pdf exploit	Antonio Perino anton ioperinom@yahoo.com	1680
2010/02/26_18:04	search-network-plus.com/load.php?a=a&e=1	212.150.164.203	-	zeus v2 trojan	Antonio Perino anton ioperinom@yahoo.com	1680

d) This IP and domain name combination is associated with a known YES exploit kit. It can now be theorized that this site is the exploitation vector.

e) The “PDF.php” string indicates a strong possibility of a PDF exploit along with the fact that the Adobe process has a suspicious network connection that led us here.

10) Perform additional searches in memory

a) Locate all search-network-plus.com instances in memory.

b) Double-click on hits to see the binary view of the memory region

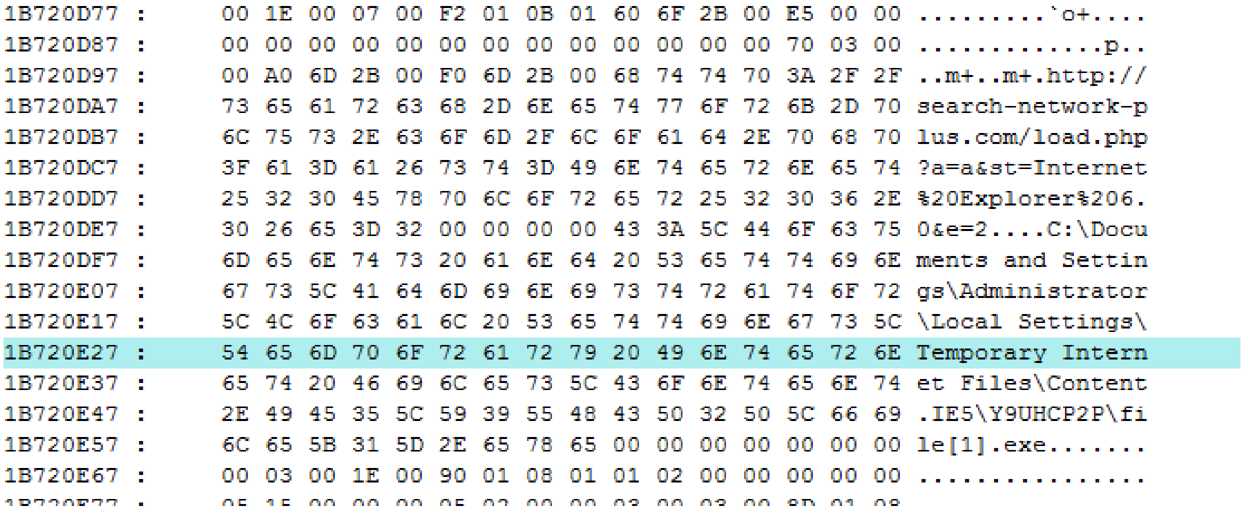
c) Look for additional URIs

i) http://search-network-plus.com/load.php?a=a&st=Internet%20Explorer%206. 0&e=2

ii) http://search-network-plus.com/cache/PDF.php?st=Internet%20Explorer%206

d) Look for any indications of downloaded malware

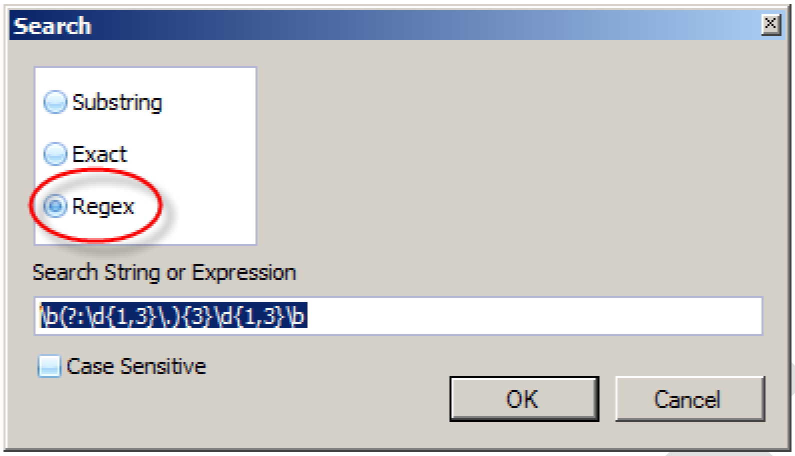
i) C:\Documents and Settings\Administrator \Local Settings\Temporary Internet Files\Content .IE5\Y9UHCP2P\fi le[1].exe



```
1B720D77 :    00 1E 00 07 00 F2 01 0B 01 60 6F 2B 00 E5 00 00 .........`o+....
1B720D87 :    00 00 00 00 00 00 00 00 00 00 00 00 00 70 03 00 .............p..
1B720D97 :    00 A0 6D 2B 00 F0 6D 2B 00 68 74 74 70 3A 2F 2F ..m+..m+.http://
1B720DA7 :    73 65 61 72 63 68 2D 6E 65 74 77 6F 72 6B 2D 70 search-network-p
1B720DB7 :    6C 75 73 2E 63 6F 6D 2F 6C 6F 61 64 2E 70 68 70 lus.com/load.php
1B720DC7 :    3F 61 3D 61 26 73 74 3D 49 6E 74 65 72 6E 65 74 ?a=a&st=Internet
1B720DD7 :    25 32 30 45 78 70 6C 6F 72 65 72 25 32 30 36 2E %20Explorer%206.
1B720DE7 :    30 26 65 3D 32 00 00 00 00 43 3A 5C 44 6F 63 75 0&e=2....C:\Docu
1B720DF7 :    6D 65 6E 74 73 20 61 6E 64 20 53 65 74 74 69 6E ments and Settin
1B720E07 :    67 73 5C 41 64 6D 69 6E 69 73 74 72 61 74 6F 72 gs\Administrator
1B720E17 :    5C 4C 6F 63 61 6C 20 53 65 74 74 69 6E 67 73 5C \Local Settings\
1B720E27 :    54 65 6D 70 6F 72 61 72 79 20 49 6E 74 65 72 6E Temporary Intern
1B720E37 :    65 74 20 46 69 6C 65 73 5C 43 6F 6E 74 65 6E 74 et Files\Content
1B720E47 :    2E 49 45 35 5C 59 39 55 48 43 50 32 50 5C 66 69 .IE5\Y9UHCP2P\fi
1B720E57 :    6C 65 5B 31 5D 2E 65 78 65 00 00 00 00 00 00 00 le[1].exe.......
1B720E67 :    00 03 00 1E 00 90 01 08 01 01 02 00 00 00 00 00 ................
1B720E77 :    05 15 00 00 00 05 02 00 00 03 00 03 00 8D 01 08
```

11) Search Internet History

a) Use the following Regex in the Internet History search panel to identify URLs accessed via an IP address: \b(?:\d{1,3}\.){3}\d{1,3}\b



○ Substring

○ Exact

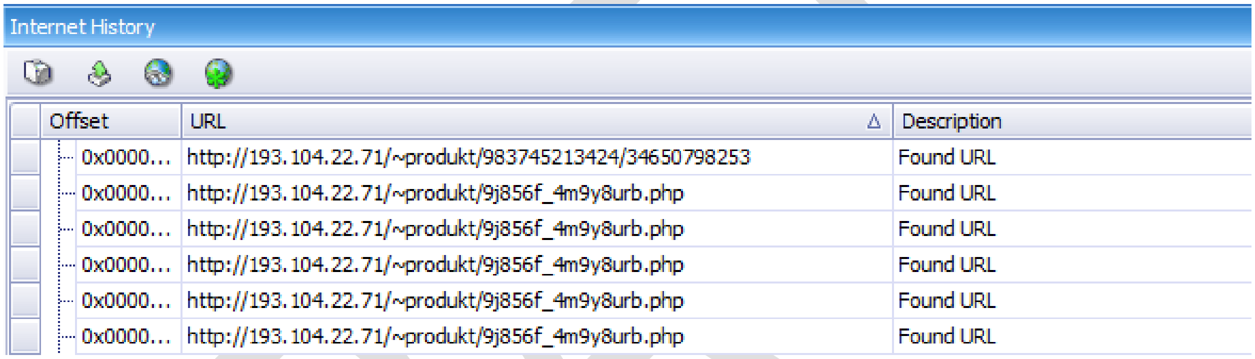
● Regex

Search String or Expression

```
\b(?:\d{1,3}\.){3}\d{1,3}\b
```

□ Case Sensitive

b) Notice the suspicious URLs which are public IP addresses



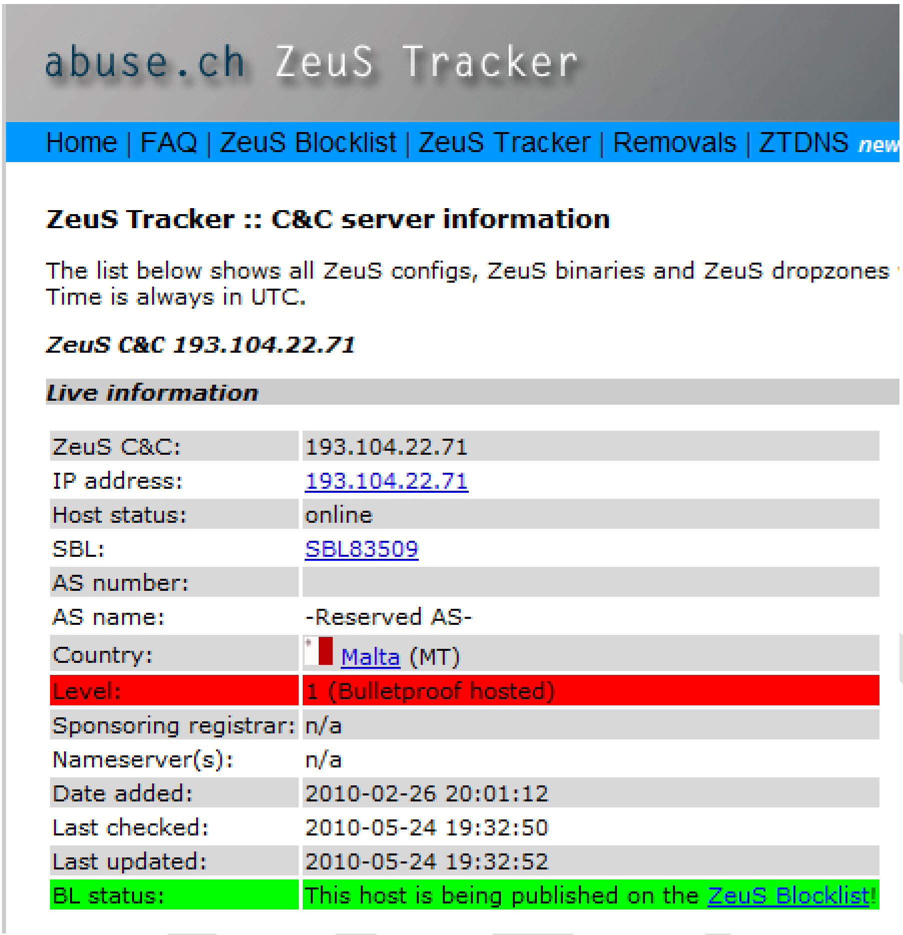
Offset	URL	Description
0x0000...	http://193.104.22.71/~produkt/983745213424/34650798253	Found URL
0x0000...	http://193.104.22.71/~produkt/9j856f_4m9y8urb.php	Found URL
0x0000...	http://193.104.22.71/~produkt/9j856f_4m9y8urb.php	Found URL
0x0000...	http://193.104.22.71/~produkt/9j856f_4m9y8urb.php	Found URL
0x0000...	http://193.104.22.71/~produkt/9j856f_4m9y8urb.php	Found URL
0x0000...	http://193.104.22.71/~produkt/9j856f_4m9y8urb.php	Found URL

i) http://193.104.22.71/~produkt/983745213424/34650798253

ii) http://193.104.22.71/~produkt/9j856f_4m9y8urb.php

c) Perform open source intelligence on this IP address

i) The first hit in Google is: https://zeustracker.abuse.ch/monitor.php?host=193.104.22.71



abuse.ch ZeuS Tracker
Home | FAQ | ZeuS Blocklist | ZeuS Tracker | Removals | ZTDNS new

ZeuS Tracker :: C&C server information

The list below shows all ZeuS configs, ZeuS binaries and ZeuS dropzones
Time is always in UTC.

ZeuS C&C 193.104.22.71

Live information

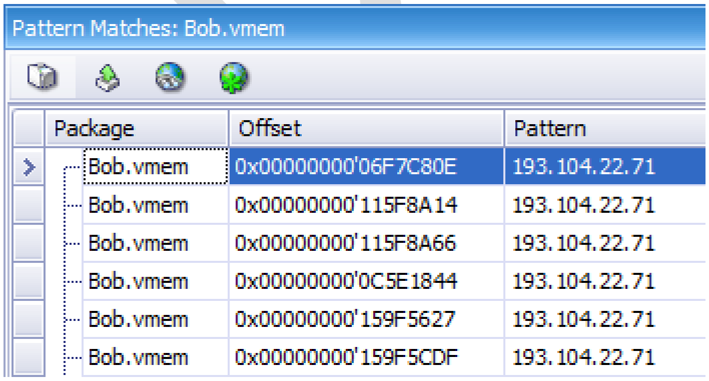
ZeuS C&C:	193.104.22.71
IP address:	193.104.22.71
Host status:	online
SBL:	SBL83509
AS number:	
AS name:	-Reserved AS-
Country:	Malta (MT)
Level:	1 (Bulletproof hosted)
Sponsoring registrar:	n/a
Nameserver(s):	n/a
Date added:	2010-02-26 20:01:12
Last checked:	2010-05-24 19:32:50
Last updated:	2010-05-24 19:32:52
BL status:	This host is being published on the ZeuS Blocklist!

ii) This IP address is associated with the ZeuS Trojan and is a command and control server

12) Leverage the “Pattern Matches” feature

a) Locate the “Pattern Matches” folder in the “Objects” tab

b) Double-click the folder to bring up the results in the right pane

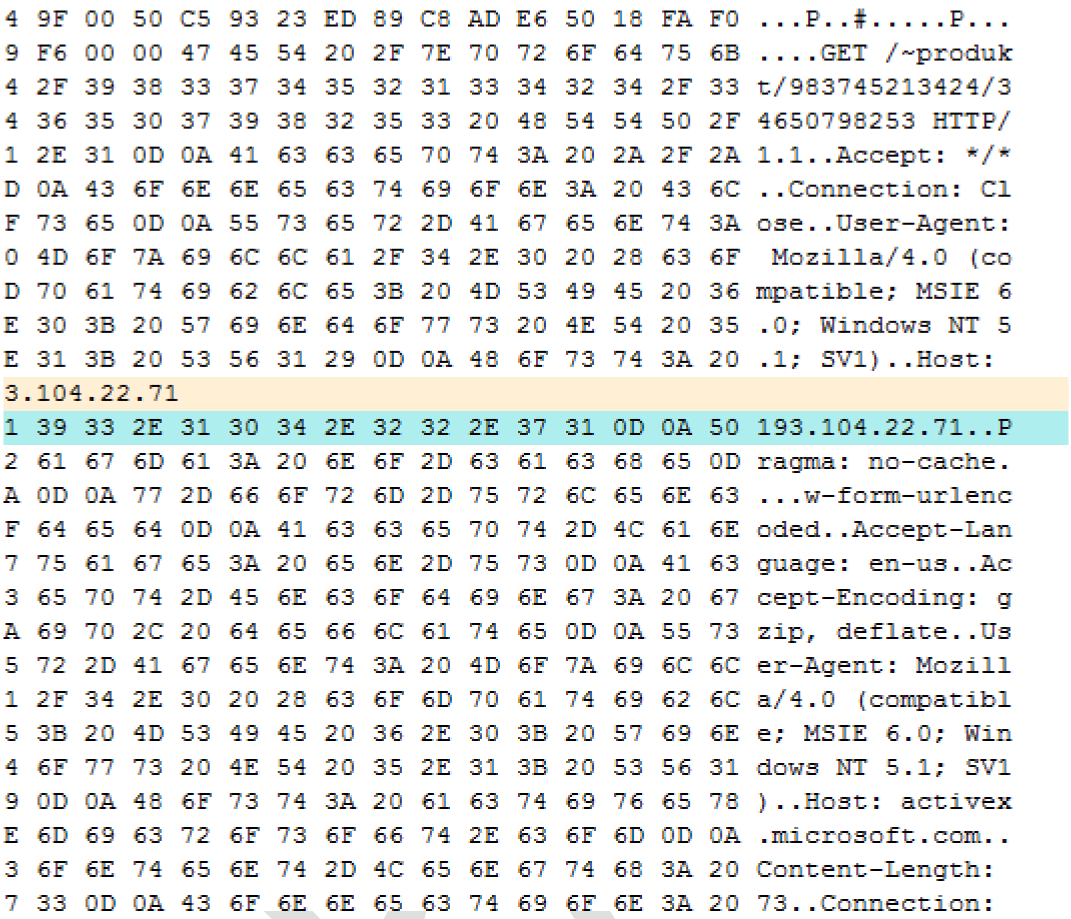


Pattern Matches: Bob.vmem

Package	Offset	Pattern
Bob.vmem	0x00000000'06F7C80E	193.104.22.71
Bob.vmem	0x00000000'115F8A14	193.104.22.71
Bob.vmem	0x00000000'115F8A66	193.104.22.71
Bob.vmem	0x00000000'0C5E1844	193.104.22.71
Bob.vmem	0x00000000'159F5627	193.104.22.71
Bob.vmem	0x00000000'159F5CDF	193.104.22.71

c) There are multiple hits for a single IP address. The address is the same one identified by the Regex search of “Internet History”.

d) Double-click the entries in the right pane to bring up the raw memory view in the left pane.



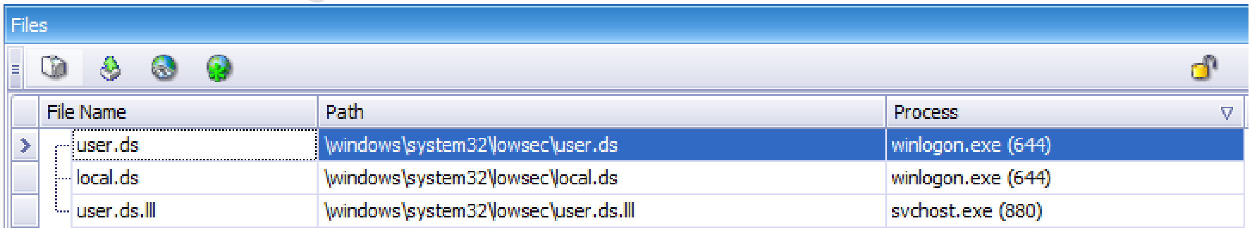
```
4 9F 00 50 C5 93 23 ED 89 C8 AD E6 50 18 FA F0 ...P..#.....P...
9 F6 00 00 47 45 54 20 2F 7E 70 72 6F 64 75 6B ....GET /~produk
4 2F 39 38 33 37 34 35 32 31 33 34 32 34 2F 33 t/983745213424/3
4 36 35 30 37 39 38 32 35 33 20 48 54 54 50 2F 4650798253 HTTP/
1 2E 31 0D 0A 41 63 63 65 70 74 3A 20 2A 2F 2A 1.1..Accept: */*
D 0A 43 6F 6E 6E 65 63 74 69 6F 6E 3A 20 43 6C ..Connection: Cl
F 73 65 0D 0A 55 73 65 72 2D 41 67 65 6E 74 3A ose..User-Agent:
0 4D 6F 7A 69 6C 6C 61 2F 34 2E 30 20 28 63 6F  Mozilla/4.0 (co
D 70 61 74 69 62 6C 65 3B 20 4D 53 49 45 20 36 mpatible; MSIE 6
E 30 3B 20 57 69 6E 64 6F 77 73 20 4E 54 20 35 .0; Windows NT 5
E 31 3B 20 53 56 31 29 0D 0A 48 6F 73 74 3A 20 .1; SV1)..Host:
3.104.22.71
1 39 33 2E 31 30 34 2E 32 32 2E 37 31 0D 0A 50 193.104.22.71..P
2 61 67 6D 61 3A 20 6E 6F 2D 63 61 63 68 65 0D ragma: no-cache.
A 0D 0A 77 2D 66 6F 72 6D 2D 75 72 6C 65 6E 63 ...w-form-urlenc
F 64 65 64 0D 0A 41 63 63 65 70 74 2D 4C 61 6E oded..Accept-Lan
7 75 61 67 65 3A 20 65 6E 2D 75 73 0D 0A 41 63 guage: en-us..Ac
3 65 70 74 2D 45 6E 63 6F 64 69 6E 67 3A 20 67 cept-Encoding: g
A 69 70 2C 20 64 65 66 6C 61 74 65 0D 0A 55 73 zip, deflate..Us
5 72 2D 41 67 65 6E 74 3A 20 4D 6F 7A 69 6C 6C er-Agent: Mozill
1 2F 34 2E 30 20 28 63 6F 6D 70 61 74 69 62 6C a/4.0 (compatibl
5 3B 20 4D 53 49 45 20 36 2E 30 3B 20 57 69 6E e; MSIE 6.0; Win
4 6F 77 73 20 4E 54 20 35 2E 31 3B 20 53 56 31 dows NT 5.1; SV1
9 0D 0A 48 6F 73 74 3A 20 61 63 74 69 76 65 78 )..Host: activex
E 6D 69 63 72 6F 73 6F 66 74 2E 63 6F 6D 0D 0A .microsoft.com..
3 6F 6E 74 65 6E 74 2D 4C 65 6E 67 74 68 3A 20 Content-Length:
7 33 0D 0A 43 6F 6E 6E 65 63 74 69 6F 6E 3A 20 73..Connection:
```

e) This example shows artifacts related to an HTTP session in memory. The user agent is visible as well as the HTTP method "GET".

13) Inspect suspect process's open file handles

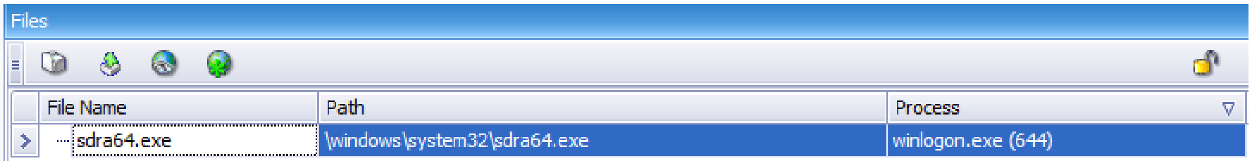
a) Attempt to locate for filesystem artifacts

b) It is known that the ZeuS Trojan uses the "lowsec" directory to store configurations and stolen credentials. Perform a search in File Handles for "lowsec"



File Name	Path	Process
user.ds	\windows\system32\lowsec\user.ds	winlogon.exe (644)
local.ds	\windows\system32\lowsec\local.ds	winlogon.exe (644)
user.ds.lll	\windows\system32\lowsec\user.ds.lll	svchost.exe (880)

c) A search can also be performed for ".exe" to look for any handles open to executable files.



Files

File Name	Path	Process
sdra64.exe	\windows\system32\sdra64.exe	winlogon.exe (644)

d) The sdra64.exe file is an artifact of the ZeuS V4 trojan.

e) The file handles should be inspected manually if dealing with an unknown infection.

Network Indicators	
IP Address	212.150.164.203
IP Address	193.104.22.71
DNS	Search-network-plus.com
Proxy	http://search-network-plus.com/load.php?a=a&st=Internet%20Explorer%206.0&e=2
Proxy	http://search-network-plus.com/cache/PDF.php?st=Internet%20Explorer%206
Filesystem Indicators	
File	C:\Documents and Settings\Administrator \Local Settings\Temporary Internet Files\Content.IE5\Y9UHCP2P\file[1].exe
File	\windows\system32\lowsec\user.ds
File	\windows\system32\lowsec\local.ds
File	\windows\system32\lowsec\local.ds.lll