

Into The Boxes

Digital Forensics and Incident Response Magazine

Welcome To Into The Boxes

January 1st, 2010
Issue 0x00



Harlan Carvey and I, Don C. Weber, have talked about starting a digital forensic and incident response (DF/IR) resource for a little over a year now. Harlan has been thinking about it for longer than that, but until recently neither of us were really in a position to start pulling together resources through our experiences and those of our peers within the security community. Nothing really has changed since we first

started talking about it, we are both still very busy in our personal and professional lives. What did change is the desire to get this resource up and running. We both believe in the potential benefit of a DF/IR magazine to the security community.

Although there are plenty of blogs out there covering DF/IR there are few resources that really pull together a collection of articles covering key issues and advancements in these fields. As Harlan mentioned in the initial post on the Into The Boxes blog, "this e-mag is NOT meant to replace anything: in fact, it's an attempt to augment what's already out there..." We are hoping that this resource will provide authors from the DF/IR industry, bloggers and non-bloggers, with a forum to present their work and discoveries to a broad audience. As this effort moves forward we are hoping that individuals within the DF/IR communities, especially those in Law Enforcement and government agencies, will find valuable information that helps them during their efforts. The sharing of information is also an important aspect of incident response and should some of the lessons learned and issues discovered here help ensure the security of businesses and government agencies then all the better.

What all of this means, however, is that we are going to need input in the form of articles, requests for research, and even resources such as system artifacts and even hardware from the community. Articles from the community is probably the easiest issue to resolve, but this requires that a few of you step up to the plate as we have done and submit your articles. Requests for research are easy as well. Harlan has mentioned to me on a number of occasions that he frequently receives requests for research into particular topics. The hard part is getting the people making these requests to supply resources to further the research. System/network artifacts, hardware, and even software are necessary to conduct detailed, effective, and reproduceable research. Whether it is evaluating the effectiveness of a DF/IR product or developing new tools such as perl or python scripts to ease investigations it takes more than just a request to get the job accomplished.

With all of that said, I would like to welcome you to the first issue of Into The Boxes. We hope you find a resource that you refer back to regularly during your DF/IR projects. We wish you the best of luck in all your future efforts and we hope that you share your experiences for the benefit of the DF/IR community. Finally, let us know your input and feedback on the Into The Boxes blog.

Enjoy,
Don C. Weber and Harlan Carvey

Featured Boxes

Windows 7 UserAssist Registry Keys - 2
Red Hat Crash Memory Forensics - 6
Hardware Quick Tip - 14
Beware The Preview Pane - 15
PCI Interview with Harlan Carvey - 18
Biographies - 20

Contributors

Didier Stevens
Jamie Levy
Don C. Weber
Harlan Carvey

Mission Statement

The mission of **Into The Boxes - Digital Forensics and Incident Response Magazine** is to provide a reliable resource regarding digital forensics and incident response topics, and issues facing the information security and computer forensic communities. The goal of **Into The Boxes** is to provide quarterly insight into technical and managerial issues in the community through content provided by professionals actively engaged in these activities. Open communications and sharing are critical components to education and advancement, and the contributors associated with **Into The Boxes** hope to provide consistent and insightful resources that will lead to open discussions and advancements within the digital forensics and incident response communities.

Windows 7 UserAssist Registry Keys

Written by Didier Stevens



Introduction

You are probably familiar with the UserAssist registry keys. If not, know that the UserAssist registry keys contain statistical data on the execution frequency of programs launched by users. The value names are ROT13 encrypted, and you can use the binary contents of the value to develop an idea what programs a user has launched via the Windows Explorer shell. If you need more details, visit my blog and search for posts with keyword "UserAssist".

With Windows 7 and Windows 2008 R2, the format of the binary data stored in the UserAssist registry values has changed. I have not found any difference between the UserAssist values in Windows 7 and Windows 2008 R2, so I will only refer to Windows 7, but know that this information applies to Windows 2008 R2, as well.

Before we dive into the details, I want to point out that beta-versions of Windows 7 and Windows 2008 R2 used a different encryption scheme for the value names. If you want to experiment with such a beta-version, or if you are in the unlikely situation that you have to conduct a forensic investigation on such a beta-version, be aware that the value names are Vigenère encrypted, and not ROT13. The encryption key is BWHQNKTEZYFSLMRGXADUJOPIVC (for more details, see <http://blog.didierstevens.com/2009/01/18/quickpost-windows-7-beta-rot13-replaced-with-vigenere-great-joke/>).

New key format

The location of the UserAssist keys with the user profile hive file has not changed, they can still be found in the registry under

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist, but with these keys:

```
{CEBFF5CD-ACE2-4F4F-9178-9926F41749EA}  
{F4E57C4B-2036-45F0-A9AB-443BCFE33D9F}
```

For the purposes of this article, all values referred to are beneath these keys, including the "Count" subkey.

As with previous versions of Windows, the information within the binary UserAssist values contains only statistical data on the applications launched by the user via Windows Explorer. Programs launched via the command-line (cmd.exe) are not counted.

UEME-strings were used to specify the type of UserAssist value, for example, UEME_RUNPATH was used to specify the launched application with its full path:

```
UEME_RUNPATH:C:\Windows\system32\cmd.exe
```

In Windows Vista, Microsoft removed a couple of UEME-strings that were used in Windows XP.

Note: here is a list of some frequently used UEME-strings in Windows XP. My UserAssist tool has an "Explain" function to explain the purpose of UEME-strings. Select a row, right-click and start the Explain... function.

- UEME_CTLSESSION: This entry is for the session ID, it doesn't hold data about executed programs
- UEME_UIQCUT: Counts the programs launched via a Quick Launch menu shortcut
- UEME_UISCUT: Counts the programs launched via a Desktop shortcut
- UEME_RUNCPL: This entry keeps data about executed control applets (.cpl)
- UEME_RUNPATH: This entry keeps data about executed programs
- UEME_RUNPIDL: This entry keeps data about executed PIDLs
- UEME_UITOOLBAR: This entry keeps data about clicks on the Windows Explorer Toolbar buttons

This trend is continued in Windows 7. In Windows 7, only 2 UEME-strings remain (you find them in shell32.dll):

UEME_CTLCUACount:ctor
UEME_CTLSESSION

But the most important change is that UserAssist registry keys for an application are no longer prefixed with a UEME-string. In Windows Vista, statistical data for cmd.exe is recorded under this UserAssist value (decoded from ROT13):

UEME_RUNPATH:C:\Windows\system32\cmd.exe

For Windows 7, statistical data for cmd.exe is recorded within this UserAssist value (decoded from ROT13):

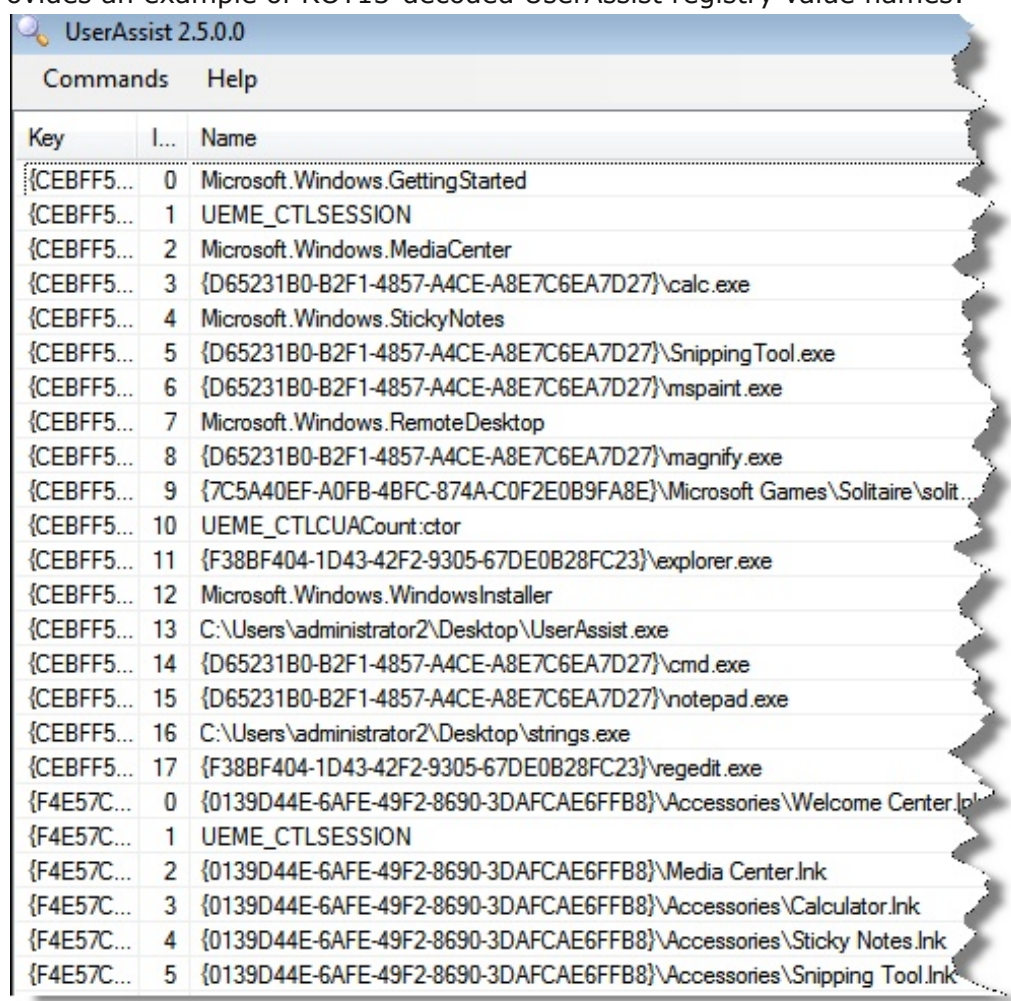
{D65231B0-B2F1-4857-A4CE-A8E7C6EA7D27}\cmd.exe

Notice the absence of the UEME_RUNPATH: prefix. Another difference is {D65231B0-B2F1-4857-A4CE-A8E7C6EA7D27}. {D65231B0-B2F1-4857-A4CE-A8E7C6EA7D27} is a well-known GUID for the SYSTEM32 folder.

Note: You lose some information, such as the ability to determine what folder the GUID refers to, if you have only the UserAssist registry keys for your analysis (for example in a .REG registry export).

{D65231B0-B2F1-4857-A4CE-A8E7C6EA7D27} tells you that the application is in the SYSTEM32 folder, but you do not know where the SYSTEM32 is located. By default, it is in C:\Windows, but it could also be in another folder, like D:\Windows, depending on the choices made when Windows 7 was installed.

Figure 1 provides an example of ROT13-decoded UserAssist registry value names:



UserAssist 2.5.0.0		
Commands		Help
Key	I...	Name
{CEBFF5...}	0	Microsoft.Windows.GettingStarted
{CEBFF5...}	1	UEME_CTLSESSION
{CEBFF5...}	2	Microsoft.Windows.MediaCenter
{CEBFF5...}	3	{D65231B0-B2F1-4857-A4CE-A8E7C6EA7D27}\calc.exe
{CEBFF5...}	4	Microsoft.Windows.StickyNotes
{CEBFF5...}	5	{D65231B0-B2F1-4857-A4CE-A8E7C6EA7D27}\SnippingTool.exe
{CEBFF5...}	6	{D65231B0-B2F1-4857-A4CE-A8E7C6EA7D27}\mspaint.exe
{CEBFF5...}	7	Microsoft.Windows.RemoteDesktop
{CEBFF5...}	8	{D65231B0-B2F1-4857-A4CE-A8E7C6EA7D27}\magnify.exe
{CEBFF5...}	9	{7C5A40EF-A0FB-4BFC-874A-C0F2E0B9FA8E}\Microsoft Games\Solitaire\solit...
{CEBFF5...}	10	UEME_CTLCUACount:ctor
{CEBFF5...}	11	{F38BF404-1D43-42F2-9305-67DE0B28FC23}\explorer.exe
{CEBFF5...}	12	Microsoft.Windows.WindowsInstaller
{CEBFF5...}	13	C:\Users\administrator2\Desktop\UserAssist.exe
{CEBFF5...}	14	{D65231B0-B2F1-4857-A4CE-A8E7C6EA7D27}\cmd.exe
{CEBFF5...}	15	{D65231B0-B2F1-4857-A4CE-A8E7C6EA7D27}\notepad.exe
{CEBFF5...}	16	C:\Users\administrator2\Desktop\strings.exe
{CEBFF5...}	17	{F38BF404-1D43-42F2-9305-67DE0B28FC23}\regedit.exe
{F4E57C...}	0	{0139D44E-6AFE-49F2-8690-3DAFCAE6FFB8}\Accessories>Welcome Center.lnk
{F4E57C...}	1	UEME_CTLSESSION
{F4E57C...}	2	{0139D44E-6AFE-49F2-8690-3DAFCAE6FFB8}\Media Center.lnk
{F4E57C...}	3	{0139D44E-6AFE-49F2-8690-3DAFCAE6FFB8}\Accessories\Calculator.lnk
{F4E57C...}	4	{0139D44E-6AFE-49F2-8690-3DAFCAE6FFB8}\Accessories\Sticky Notes.lnk
{F4E57C...}	5	{0139D44E-6AFE-49F2-8690-3DAFCAE6FFB8}\Accessories\Snipping Tool.lnk

Figure 1: Decoded UserAssist Value Names

Figure 2 provides an illustration of the binary contents of UserAssist values, after they have been parsed into their various components.

counter	Last	Last UTC	Focus counter?	Focus time?	Flags?
14	5/12/2009 18:53:39	5/12/2009 17:53:39	21	420000	0
13	5/12/2009 18:53:39	5/12/2009 17:53:39	19	380000	0
12	5/12/2009 18:53:39	5/12/2009 17:53:39	17	340000	0
11	5/12/2009 18:53:39	5/12/2009 17:53:39	15	300000	0
10	5/12/2009 18:53:39	5/12/2009 17:53:39	13	260000	0
9	5/12/2009 18:53:39	5/12/2009 17:53:39	11	220000	0
8	5/12/2009 18:53:39	5/12/2009 17:53:39	9	180000	0
7	5/12/2009 18:53:39	5/12/2009 17:53:39	7	140000	0
6	5/12/2009 18:53:39	5/12/2009 17:53:39	8	342453	0
0			0	0	0
1	5/12/2009 19:11:45	5/12/2009 18:11:45	11	245437	0
0			0	4531	0
2	6/12/2009 15:32:18	6/12/2009 14:32:18	14	224298	0
3	5/12/2009 19:39:20	5/12/2009 18:39:20	14	452111	0
2	5/12/2009 19:36:24	5/12/2009 18:36:24	5	21797	0
0			0	2750	0
1	5/12/2009 19:36:52	5/12/2009 18:36:52	1	60000	0
14	5/12/2009 18:53:39	5/12/2009 17:53:39	0	14	0
13	5/12/2009 18:53:39	5/12/2009 17:53:39	0	13	0
12	5/12/2009 18:53:39	5/12/2009 17:53:39	0	12	0
11	5/12/2009 18:53:39	5/12/2009 17:53:39	0	11	0
10	5/12/2009 18:53:39	5/12/2009 17:53:39	0	10	0

Figure 2: UserAssist Binary Value Contents, Translated

The structure of the binary data within the UserAssist key values will be presented within this article.

If you are unable to identify the program that was executed, because the UserAssist registry entry contains only a GUID (or two GUIDS), you have to lookup this GUID on the Internet or in the Windows registry to identify the associated program.

As is done in previous versions of Windows, the UserAssist registry keys for every new user will be pre-populated with some default values, as illustrated in figures 3 and 4.

Key	Index	Name
{CEBFF5...	0	Microsoft.Windows.GettingStarted
{CEBFF5...	2	Microsoft.Windows.MediaCenter
{CEBFF5...	3	{D65231B0-B2F1-4857-A4CE-A8E7C6EA7D27}\calc.exe
{CEBFF5...	4	Microsoft.Windows.StickyNotes
{CEBFF5...	5	{D65231B0-B2F1-4857-A4CE-A8E7C6EA7D27}\SnippingTool.exe
{CEBFF5...	6	{D65231B0-B2F1-4857-A4CE-A8E7C6EA7D27}\mspaint.exe
{CEBFF5...	7	Microsoft.Windows.RemoteDesktop
{CEBFF5...	8	{D65231B0-B2F1-4857-A4CE-A8E7C6EA7D27}\magnify.exe
{CEBFF5...	9	{7C5A40EF-A0FB-4BFC-874A-C0F2E0B9FA8E}\Microsoft Games\Solitaire
{CEBFF5...	1	UEME_CTLSESSION
{CEBFF5...	10	UEME_CTLCUACount:ctor
{F4E57C...	0	{0139D44E-6AFE-49F2-8690-3DAFCAE6FFB8}\Accessories\Welcome Center
{F4E57C...	2	{0139D44E-6AFE-49F2-8690-3DAFCAE6FFB8}\Media Center\Ink
{F4E57C...	3	{0139D44E-6AFE-49F2-8690-3DAFCAE6FFB8}\Accessories\Calculator\Ink
{F4E57C...	4	{0139D44E-6AFE-49F2-8690-3DAFCAE6FFB8}\Accessories\Sticky Notes
{F4E57C...	5	{0139D44E-6AFE-49F2-8690-3DAFCAE6FFB8}\Accessories\Snipping Tool
{F4E57C...	6	{0139D44E-6AFE-49F2-8690-3DAFCAE6FFB8}\Accessories\Paint\Ink
{F4E57C...	7	{0139D44E-6AFE-49F2-8690-3DAFCAE6FFB8}\Accessories\Remote Desktop
{F4E57C...	8	{A77F5D77-2E2B-44C3-A6A2-ABA601054A51}\Accessories\Accessibility\...
{F4E57C...	9	::{ED228FDF-9EA8-4870-83B1-96B02CFE0D52}\{00D8862B-6453-4957-A82...
{F4E57C...	1	UEME_CTLSESSION
{F4E57C...	10	UEME_CTLCUACount:ctor

Figure 3: UserAssist Key Default Values

C...	Last	Last UTC	Fo...	Focus t...	Flags?
14	6/12/2009 18:45:30	6/12/2009 17:45:30	21	420000	0
13	6/12/2009 18:45:30	6/12/2009 17:45:30	19	380000	0
12	6/12/2009 18:45:30	6/12/2009 17:45:30	17	340000	0
11	6/12/2009 18:45:30	6/12/2009 17:45:30	15	300000	0
10	6/12/2009 18:45:30	6/12/2009 17:45:30	13	260000	0
9	6/12/2009 18:45:30	6/12/2009 17:45:30	11	220000	0
8	6/12/2009 18:45:30	6/12/2009 17:45:30	9	180000	0
7	6/12/2009 18:45:30	6/12/2009 17:45:30	7	140000	0
6	6/12/2009 18:45:30	6/12/2009 17:45:30	5	100000	0
0			0	0	0
14	6/12/2009 18:45:30	6/12/2009 17:45:30	0	14	0
13	6/12/2009 18:45:30	6/12/2009 17:45:30	0	13	0
12	6/12/2009 18:45:30	6/12/2009 17:45:30	0	12	0
11	6/12/2009 18:45:30	6/12/2009 17:45:30	0	11	0
10	6/12/2009 18:45:30	6/12/2009 17:45:30	0	10	0
9	6/12/2009 18:45:30	6/12/2009 17:45:30	0	9	0
8	6/12/2009 18:45:30	6/12/2009 17:45:30	0	8	0
7	6/12/2009 18:45:30	6/12/2009 17:45:30	0	7	0
6	6/12/2009 18:45:30	6/12/2009 17:45:30	0	6	0
0			0	0	0

Figure 4: UserAssist Key Default Values Data

These default values should be taken into account during forensic analysis.

New data format

The most significant change of the UserAssist keys in Windows 7 is a new format for the binary value data. To date, not all the binary formats have been reverse engineered. This is a work in progress, as Microsoft has never published official documentation covering the content of UserAssist registry values. In fact, the ROT13-encoding of the keys is meant as a means for Microsoft to prevent tamper with the content of these keys by normal users. As an illustration, the ROT13 encoding prevents a user from locating the value names via an ASCII keyword search.

In older versions of Windows, the binary data for program-execution recorded in the UserAssist registry keys is 16 bytes long and had the following format:

- 32-bit integer identifying a session
- 32-bit integer counting the number of times the program was executed (offset by 5)
- 64-bit timestamp in FILE_TIME format

With Windows 7, this binary data format has changed. For program-execution entries and UEME_CTLCUACount:ctor entries, the binary data is 72 bytes long.

The binary data for both UEME_CTLCUACount:ctor entries (one for key {CEBFF5CD-ACE2-4F4F-9178-9926F41749EA} and one for key {CEBFF5CD-ACE2-4F4F-9178-9926F41749EA}) is always the same:

```
ffffff 00000000 00000000 00000000 000080bf
000080bf 000080bf 000080bf 000080bf 000080bf
000080bf 000080bf 000080bf 000080bf fffffff
00000000 00000000 00000000
```

I have never witnessed changes in this data when using Windows 7.

For program-execution entries, I have identified several variables in the binary data format.

From bytes 0 to 3, we find a 32-bit integer that is always zero (remember that this integer is equal to 0xFFFFFFFF for UEME_CTLCUACount:ctor entries).

From bytes 4 to 7 (I assign index 0 to the first byte of the 72 bytes), there is a 32-bit integer, counting the number of times the program was executed. Unlike previous versions of Windows, this counter is not offset by 5. The counter counts exactly how many times a program was launched, you do not have to subtract 5 from the counter. This counter is functionally equivalent with the execution counter in previous version of Windows.

From bytes 8 to 11, we find another 32-bit counter. This counter is usually larger than the program-execution counter, and I believe it counts the number of times an application receives focus. An application has focus when you type on the keyboard and the keystrokes go to the application. Applications receive focus when they are launched and when you select the application while switching between applications. I do not believe this counter has much forensic value, but remember that the UserAssist registry keys were not designed with forensic analysis in mind. The UserAssist registry keys collect statistical data to assist the user by presenting applications that are often utilized by the user. For example, this data is used to populate the Start Menu with frequently-used applications.

From bytes 12 to 15, we find another 32-bit value. The units used in this value are milliseconds. I have the strong impression that this value times the duration an application has focus or the time a user spends inside this application. This timer is useful as it gives you an indication how much time a user has spent directly interacting with an application. When you divide this timer with the number of times an application was started (the first counter), you get an indication of the average time a user spends interacting with the application.

From bytes 16 to 55, we find a series of 32-bit integers with the same value (0x000080BF). As I have never seen changes to this values when using Windows 7, I theorize they are entries reserved for future usage.

From bytes 56 to 59, the 32-bit integer value is always 0xFFFFFFFF. Its purpose is unknown.

From bytes 60 to 67, we have a FILETIME timestamp, a 64-bit value that records the last time an application was launched. This timestamp is the same as the last execution timestamp in previous Windows versions.

And finally, from bytes 68 to 71, we find a 32-bit integer value always equal to zero. Its purpose is

also unknown.

There is also a second binary data format that remains to be reversed. This data format is associated with UEME_CTLSESSION and is 1612 bytes long! It contains UNICODE strings like {0139D44E-6AFE-49F2-8690-3DAFCAE6FFB8}\Accessories\Welcome Center.lnk.

UserAssist tool

I have written a tool to decode and display the content of the UserAssist registry keys, called the UserAssist tool. You can find this tool via my blog. The UserAssist tool supports Windows versions 2000 through Vista. If you need a tool to analyze UserAssist registry keys from Windows 7, I have released a special version (version 2.5.0.0) of my UserAssist tool (<http://didierstevens.com/files/software/UserAssistWindows7LaunchParty.zip>).

Use it only for Windows 7 and Windows 2008 R2. When I have made more progress in reversing the binary data format, I will release a new version supporting Windows versions 2000 through 7.

Conclusion

The UserAssist registry key values contain more data in Windows 7 than in previous Windows versions, and the analysis of the data format is a work in progress. We already know that we find the same important forensic data in this new format: the counter with the number of program launches and the timestamp with the last execution of a program. It remains to be seen if the other binary data (like the application focus data) has any forensic value, but more research will have to be conducted to fully understand its behavior. If you do not completely understand the meaning of the data you collect, its forensic value is little, especially if you want to use it as evidence in a legal case.

Also, the behavior of the focus data is still not fully understood. Take a look at the UserAssist tool screenshot in figure 1 of this article. Look for the strings.exe program. Notice that all of the binary data is zero, except the focus time which is 2750 (i.e., 2.75 seconds)? How can a program have an execution counter of zero, no last execution timestamp, and still had focus for 2.75 seconds? Strings.exe is a command-line program that I launched from the command-line (cmd.exe), hence there is no UserAssist data collected for this program. But because it was the first time I used strings.exe on this Windows 7 machine, it displayed its EULA (this version of strings.exe is from Microsoft, formerly the SysInternals tools) in a dialog window and I had to accept it before it would continue to run. I estimate it took me less than 3 seconds to click this dialog away...

This example illustrates that it is important to fully understand the new binary data format, and I count on your help for this. When you use my UserAssist tool and observe data that is not consistent with what I explained in this article, please get in touch with me. It will help us with our understanding of the UserAssist registry keys.

Red Hat Crash Memory Forensics

Written by Jamie Levy



Introduction

There has been a great deal of advancement in Windows memory forensics in the last few years. There are memory dumpers (mdd, windd, winen) and there are tools extract valuable information from these memory dumps (Memoryze, PTFinder, Volatility). There is not as much obvious advancement when it comes to *nix memory however. Though some tools have been around for some time, some people may not be readily aware of them. We will take a look at one of these tools

in the Linux world.

Background

Linux systems have 2-3 locations, depending on the distribution, where memory can be readily accessed: /dev/mem and /proc/kcore and on some distributions /dev/kmem. /dev/mem is a character driver that mirrors physical memory. The /dev/kmem character driver is the same as /dev/mem except that it is for kernel memory. /dev/kmem is not available on some Linux systems. The /proc/kcore file is a special ELF format file memory alias often used to debug the kernel using gdb. /proc/kcore may or may not be the same size as RAM depending on if HIGHMEM is enabled. We will discuss /dev/mem access for the remainder of this article.

To obtain memory dumps in Linux type the following:


```
$ dd if=/dev/mem of=<external path>/<file name>
```

Where <external path> is a path outside of the system's filesystem and <file name> is the name of the resulting dump file. So if one is dumping memory to an external usb device the command might be:

```
$ dd if=/dev/mem of=/mnt/disk/mem.dd
```

Some distributions (Redhat, Fedora, Ubuntu) block access to /dev/mem by disallowing random read/write access past the first eight pages. For Redhat and Fedora systems there is a /dev/crash driver that may be used for the Redhat Crash Utility in order to allow access to RAM. This driver can also be ported to Ubuntu but still needs some tweaking in order to access vmalloc'ed (virtual) memory (Levy, 2009). It is possible to make a driver using the *drivers/char/mem.c* code, provided that the driver is named something other than /dev/mem so it will not conflict with the current /dev/mem driver.

Installation and Setup of the Redhat Crash Utility

Download the latest Redhat Crash Utility at the official website (<http://people.redhat.com/anderson/>) and extract it like so:

```
$ tar xvfz crash-4.1.1.tar.gz
```

In order to make Crash work with /dev/mem memory dumps you will have to patch it. The patch can be obtained from <http://code.google.com/p/jls-scripts/> and is modified from the patch created for the DFRWS 2008 forensic challenge to work with Crash 4.1.1 (A. Walters, 2008). In order to patch Crash, copy the *volcrash* patch into the crash-4.1.1 directory and type the following:

```
$ cd crash-4.1.1  
$ patch -p1 < volcrash-4.1.1.patch
```

After Crash is patched you can compile it by typing:

```
$ make
```

Also needed is the *pykdump* extension which can be downloaded from Sourceforge. Make sure you download the source code and not the compiled library file. You can obtain the latest source code from the SVN:

```
$ svn co \ https://pykdump.svn.sourceforge.net/svnroot/pykdump/testing \ pykdump
```

In order to compile the library file, you will also need to download the source code for your version of Python. You can find the version by issuing the following command:

```
$ python -V
```

In my case I have version Python 2.5.2 and have downloaded the source code for that version. First you will need to compile Python by typing the following (Sidorenko, 2009):

```
$ tar xvfj /ar/new/Python-2.5.2.tar.bz2  
$ cd Python-2.5.2  
$ ./configure --enable-shared --prefix=$PWD/local
```

You will have to copy the Setup.local file to your Python/Modules directory:

```
$ cp pykdump/Extensions/Setup.local Python-2.5.2/Modules/  
$ touch Modules/Setup.local
```

Compile Python and its libraries:

```
$ make  
$ make libinstall
```

Once that is complete, change into the pykdump directory and type the following:

```
$ ./configure -p <replace with path>/Python-2.5.2 -c \  
/ <replace with path>/crash-4.1.1  
  
$ cd build_standalone
```

```
$ make
```

Copy the resulting library `mpykdump.so` file into your `crash-4.1.1\extensions` directory and you are set. You can refer to the installation manual online if you encounter any problems (Sidorenko, 2009).

Compiling the Linux Kernel

In order to use the Redhat Crash Utility you need a *System.map* mapfile and a *vmlinux* file (uncompressed Linux kernel object file with debugging symbols). The *System.map* file is often easy to find under the `/boot` directory. Most systems do not have a *vmlinux* file readily available, so you will have to make your own.

If the target machine (from which the memory dump was obtained) is running a recent kernel/distribution then obtaining a *vmlinux* file will be somewhat easier (Levy, 2009). If not you can find older kernels at www.linux.org or www.linuxhq.com. Once you have obtained the code, you can begin the compiling process. Extract the code on your Linux system (does not necessarily have to be the same distribution as the target machine) and change into the resulting directory.

Since the kernel must be compiled with the `-g` flag to enable debugging symbols, some modification to the *Makefile* is needed. For most modern kernels this is as simple as changing the following line from:

```
CFLAGS_KERNEL =
```

to

```
CFLAGS_KERNEL = -g
```

Older kernels may require some other options added to the *Makefile*. With some older kernels you may also have to change other options for SMP or other changes that may have been made to the target system. If you have the *System.map* file for the target you can see if there are any other options that you may need to accommodate.

Once you have finished making all of your changes to the *Makefile* you should be able to type `make` at the command line to obtain your new *System.map* and *vmlinux* files. Now you should be ready to use the Redhat Crash Utility with your obtained memory dump.

Using the Redhat Crash Utility

In order to start the Redhat Crash Utility, change into the `crash-4.1.1` directory and invoke `crash` like this:

```
$ ./crash <memory dump> -f <System.map file> <vmlinux file> \  
--volatile
```

Make sure the paths are correct. An example run might be:

```
$ ./crash /storage/mem.dd -f /storage/System.map \ /storage/vmlinux --volatile
```

When the Crash Utility starts up you should see the system information. For example on a BT 4 target:

```
SYSTEM MAP: /usr/src/linux-source-2.6.28.1/debian/linux-image-2.6.28.1/boot/System.map-  
2.6.28.1  
DEBUG KERNEL: /usr/src/linux-source-2.6.28.1/vmlinux (2.6.28.1)  
DUMPFILE: /storage/mem-bt2.dd  
CPUS: 1  
DATE: Wed Dec 2 14:45:23 2009  
UPTIME: 04:11:59  
LOAD AVERAGE: 2.04, 2.00, 1.55  
TASKS: 162  
NODENAME: bt  
RELEASE: 2.6.28.1  
VERSION: #2 SMP Wed Feb 4 21:50:02 EST 2009  
MACHINE: i686 (1995 Mhz)  
MEMORY: 511.5 MB  
PID: 0
```



```
COMMAND: "swapper"
TASK: c084d340 [THREAD_INFO: c0908000]
CPU: 0
STATE: TASK_RUNNING
```

There is a lot of information obtainable from the Redhat Crash Utility such as:

- Currently running processes
- Open files
- Kernel stack backtrace
- Kernel log
- Kernel modules
- Network data
- Mounted file systems

The base set of commands is listed below in Table 1. There are extensions available that add to the command set. A list is available online from the RedHat Crash Utility site (Anderson 2003).

*	files	mod	runq	union
alias	foreach	mount	search	vm
ascii	fuser	net	set	vtop
bt	gdb	p	sig	waitq
btop	help	ps	struct	whatis
dev	irq	pte	swap	wr
dis	kmem	ptob	sym	q
eval	list	ptov	sys	exit
log	rd	task	extend	mach
repeat	timer			

Table 1: Basic RedHat Crash Utility Commands

More information is available about any of the above commands by issuing a help command at the "crash>" prompt, for example:

```
crash> help foreach
```

The obvious commands that should be used are:

- ps - process listing
- log - system log (dmesg)
- dev - device data
- mount - mounted file systems
- swap - swap data
- net - network data

We will cover some of these commands as well as some others not yet mentioned. Commands can be typed in at the "crash>" prompt or given via a text file to the Redhat Crash Utility when it starts up with the -i flag as well as placed in a .crashrc file in either the \$HOME directory or directory that contains the Crash program. Creating commands ahead of time is very useful for getting output from a lot of basic commands at once. You can also use redirection [<, >, <<, >>] in Crash commands as well as basic bash commands and pipes [|].

The *foreach* command is also very useful for getting information from more than one object. For example, in order to get all of the open files for each process that is running type:

```
crash> foreach files
```

To extend the command set issue the "extend" command with the path of the extension, in the case of our pykdump extension:

```
crash> extend ./extensions/mpykdump.so
```

If all goes well you will see a note that the object has been loaded. If you type help at the "crash>" prompt you will see some new commands, one of which is "xportshow" which shows network data in more detail than the "net" command. It also loads an embedded Python interpreter that allows one to run Python scripts within Crash with the *epython* command (Sidorenko, 2009). To get all network information type one of the following depending on how detailed you would like your output:

```
crash> xportshow --everything
crash> xportshow -a
crash> xportshow -iv
```

Linux Process Memory Space

There are three structures of interest that Linux uses to keep track of a process' virtual memory space: **task_struct**, **mm_struct** and **vm_area_struct**. The **task_struct** is a linked list that allows the kernel to keep track of what task is currently running (Bovet 2000). The **mm_struct** contains the memory management information for the task (Bovet 2000). Finally the **vm_area_struct** points to a set of virtual memory handling routines (Bovet 2000). We will discuss each of these in this section.

We can see the **task_struct** for any process by typing the following command at the "crash>" prompt:

```
crash> task <pid>
```

where <pid> is replaced by the process ID of the target process from the *ps* process listing. Below is a shortened sample output:

```
PID: 25351 TASK: df096380 CPU: 0  COMMAND: "firefox"
struct task_struct {
  state = 1,
  stack = 0xc5d7c000,
  usage = {
    counter = 2
  },
  flags = 4202752,
  ptrace = 0,
  lock_depth = -1,
  ...
```

This is a nice feature since we can see the values of the items in the **task_struct** structure. To get an idea about the variable types used and defined system structure you can use the *whatis* command for example on **task_struct**:

```
crash> whatis task_struct
```

```
struct task_struct {
  volatile long int state;
  void *stack;
  atomic_t usage;
  unsigned int flags;
  unsigned int ptrace;
  int lock_depth;
  int prio;
  ...
```

So if we continue our walk through these structures we will look for the **mm_struct**'s associated with this process we can read their contents:

..continued from above

```

mm = 0xd4ac8380,
active_mm = 0xd4ac8380,
...

crash> struct mm_struct 0xd4ac8380

struct mm_struct {
    mmap = 0xd4b88b58,
    mm_rb = {
        rb_node = 0xcc85e960
    },
    mmap_cache = 0xc639a840,
    ...
    ...
    exe_file = 0xdf9d3180,
    ...
}

```

Now we can find the **vm_area_struct** pointed to by the **mm_struct**:

```

crash> struct vm_area_struct 0xd4b88b58

struct vm_area_struct {
    vm_mm = 0xd4ac8380,
    vm_start = 134512640,
    vm_end = 134541312,
    vm_next = 0xd4b88688,
    ...
}

```

And we can examine each memory section of the process if we continue.

Notice in the code of **mm_struct** that there is a **exe_file** variable with a memory address. This is a **struct file** structure and if we want to see the values for this, we can issue the following command:

```

crash> struct file 0xdf9d3180

struct file {
    ...
    f_path = {
        mnt = 0xdf1f3380,
        dentry = 0xdbed4680
    },
    ...
}

```

If we examine the code for **struct file** we will see that **f_path** is defined as:

```
struct path f_path;
```

which enumerates to:

```

struct path {
    struct vfsmount *mnt;
    struct dentry *dentry;
}

```

Therefore we can examine these two items by typing "struct vfsmount 0xdf1f3380" and "struct dentry 0xdbed4680". We can examine these items further to extract and verify mount information and **dentry** and **inode** information.

Files

The **dentry** and **inode** are useful for extracting file information such as subdirectories, ownership and MAC times. We can extract this information without having to traverse each of memory mapping structs above by using a few of the basic commands from Crash. First we will issue a process listing and pick a process to examine, in this case firefox:

```
crash> ps |grep firefox
```

```
25351 4369 0 df096380 IN 11.4 154764 59976 firefox
```

```
crash> files 25351
```

```
PID: 25351 TASK: df096380 CPU: 0 COMMAND: "firefox"
```

```
ROOT: / CWD: /root
```

FD	FILE	DENTRY	INODE	TYPE	PATH
0	df0ffe40	dbcfd80	dbc77d20	FIFO	
1	df330240	dbc85180	dbc838d0	REG	/root/.xsession-errors
2	df330240	dbc85180	dbc838d0	REG	/root/.xsession-errors
...					

Notice that the **file**, **dentry** and **inode** entries are already populated for each file. Now we can examine each of these by issuing the struct commands like before. Looking at the **inode** information we see the following (Bovet 2000):

```
crash> struct inode dbc838d0
```

```
...
i_nlink = 1,      //number of hard links
i_uid = 0,        //user id (root)
i_gid = 0,        //group id (root)
...
i_size = 14397,   //file length in bytes
...
i_atime = {       //time of last file access
  tv_sec = 1259781534,
  tv_nsec = 0
},
i_mtime = {       //time of last modification
  tv_sec = 1259781548,
  tv_nsec = 0
},
i_ctime = {       //time of last inode change
  tv_sec = 1259781548,
  tv_nsec = 0
},
...
```

We can get the time stamps back in human readable format with a quick Perl command from another terminal:

```
$ perl -e 'print localtime(1259781548) . "\n";'
Wed Dec 2 14:19:08 2009
```

```
$ perl -e 'print localtime(1259781534) . "\n";'
Wed Dec 2 14:18:54 2009
```

The *rd* (read) command allows one to read memory contents at a given memory address. The memory address should be: a physical address, user virtual address, dumpfile offset or xen host address (Anderson 2003). Suppose there is a program called "test" that is running:

```
crash> ps|grep test
25620 24493 0 dde44700 RU 0.1 1636 368 test
```

To get the memory map of this process type:

```
crash> vm 25620
```

```
PID: 25620 TASK: dde44700 CPU: 0 COMMAND: "test"
```

MM	PGD	RSS	TOTAL_VM
c61b68c0	dda9c000	368k	1636k


```

VMA      START    END    FLAGS  FILE
c6237420 8048000 8049000 8001875 /root/test
c62373c8 8049000 804a000 8101871 /root/test
c6237528 804a000 804b000 8101873 /root/test
...
c6237268 b808b000 b808c000 8100075 /root/strcmp.so
c6237210 b808c000 b808d000 8100071 /root/strcmp.so
c62374d0 b808d000 b808e000 8100073 /root/strcmp.so

```

There is a non-standard library file loaded by this program. Suppose we want to see the contents of it. First we need a physical address:

```

crash> vm -p 25620
...
c6237268 b808b000 b808c000 8100075 /root/strcmp.so
VIRTUAL  PHYSICAL
b808b000 8a20000
VMA      START    END    FLAGS  FILE
c6237210 b808c000 b808d000 8100071 /root/strcmp.so
VIRTUAL  PHYSICAL
b808c000 5e50000
VMA      START    END    FLAGS  FILE
c62374d0 b808d000 b808e000 8100073 /root/strcmp.so
VIRTUAL  PHYSICAL
b808d000 1b464000

```

The physical addresses are in bold. In order to read the contents type:

```

crash> rd -p 8a20000 100

8a20000: 464c457f 00010101 00000000 00000000 .ELF.....
8a20010: 00030003 00000001 00000370 00000034 .....p...4...
8a20020: 000014a8 00000000 00200034 00280005 .....4. ...(.
8a20030: 001d0020 00000001 00000000 00000000 .....
8a20040: 00000000 000004d0 000004d0 00000005 .....
...

```

We can see this is has an ELF header, however the HEX view is not in normal byte order. Suppose we want to extract this library file to get a better look at it. We can capture it by redirecting our output:

```

crash> rd -p 8a20000 100 > strcmp_so

```

To reconstruct it, one can use a Perl script to adjust the line numbers (to start at 00000000:) and the correct byte order format for the HEX view. Then to recover using xxd:

```

$ cat strcmp_so_fixed | xxd -r > strcmp_so_final

```

One thing to note about ELF executables is that the ELF format differs depending on whether or not it is loaded into memory (Santosa, 2006). For further information on ELF format one may consult the ELF manual (<http://www.x86.org/ftp/manuals/tools/elf.pdf>).

Conclusion

The Redhat Crash Utility is a useful tool for Linux memory forensics, which enables one to extract information about running processes, files, network information and other items from memory dumps. Crash can also be extended using the *extend* command (Anderson, 2003). With the *mpykdump* extension, one can run Python scripts, which creates the possibility for extracting more information in a more efficient manner.

References

Anderson, D. (2003) White Paper: Red Hat Crash Utility, Retrieved from http://people.redhat.com/anderson/crash_whitepaper/

Bovet, D., M. Cesati (2000). Understanding the linux kernel. Cambridge: O'Reilly Media.

Levy, J. (2009, August 24). /dev/crash driver. Retrieved from <http://gleeda.blogspot.com/2009/08/devcrash-driver.html>

Santosa, M. (2006, June 16). Understanding ELF using readelf and objdump. Retrieved from http://www.linuxforums.org/articles/understanding-elf-using-readelf-and-objdump_125.html

Sidorenko, A. (2009, July 31). Building from svn. Retrieved from <http://sourceforge.net/apps/mediawiki/pykdump/index.php?title=Building>

Walters, A., M. Cohen, D. Collett (2008, July 15). Linux Memory Forensics. Retrieved from <http://volatilesystems.blogspot.com/2008/07/linux-memory-analysis-one-of-major.html>

Hardware Quick Tip

Written by Don C. Weber



Quick Tip

Before destroying an external hard drive for data protection purposes, don't forget to check for the hidden prize. Depending on the external USB hard drive it may contain a ~\$30 hard drive adapter that would make a valuable addition to any tool kit. The following images come from a Western Digital WD5000ME 500 GB external USB hard drive. The black casing contains a Western Digital WD5000BEVT hard drive with a mini-USB-to-serial ATA (SATA) adapter. Although internal hard drive adapters will probably vary depending on the vendor and hard drive model, Figure 1

and 2 show that this external USB hard drive contains a mini-USB port to SATA data and power ports.

The advantage of an having this adapter is that it is a very easy method for accessing SATA drives. This adapter supplies power and access to the data via a short mini-USB cable (avoid long cables as they may not supply sufficient power to the drive). Data can be accessed by mounting the SATA drive read-only (e.g. "sudo mount -o ro /dev/sourcedrive /media/client-mnt"). The drive can also be imaged using bit-stream commands (e.g. "dcflddd if=/dev/sourcedrive hash=md5,sha256 hashwindow=10G md5log=md5.txt sha256log=sha256.txt \ hashconv=after bs=512 conv=noerror,sync split=10G splitformat=aa of=driveimage.dd" [sic] (Forensics Wiki, 2009)) or by

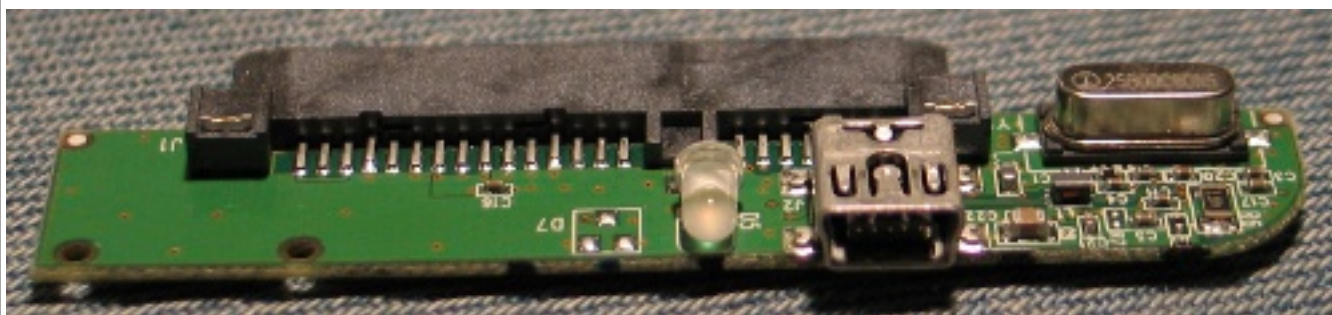


Figure 1: Mini-USB Port

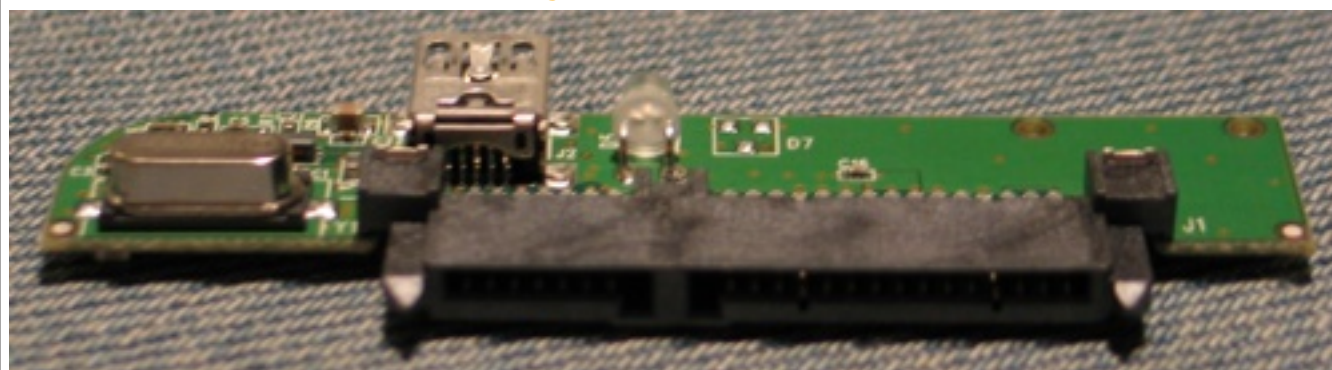


Figure 2: SATA Data and Power Ports

using a USB write blocker such as the Forensic USB Bridge Model T8 (Tableau, 2009) or the Image MASster™ Solo-4 Forensic Hand-Held Duplicator (Intelligent Computer Solutions, Inc., 2009).

References

Tableau, . (2009, November 1). Tableau t8 forensic usb bridge . Retrieved from <http://www.tableau.com/index.php?pageid=products&model=T8>

Intelligent Computer Solutions, Inc. (2009, November). Imagemasster solo-4 forensic. Retrieved from http://www.icsforensic.com/index.cfm/action/product.show/id_product/20058a69-21b5-443b-96b9-f658987f0855/id_category/d89a0493-1e36-4538-b0e5-6c6fbd2c704b?CFID=28584658&CFTOKEN=68233947

Forensics Wiki, . (2009, August 1). Dcfldd. Retrieved from <http://www.forensicswiki.org/wiki/Dcfldd>

Beware The Preview Pane

Written by Don C. Weber



Introduction

FTK Imager (AccessData, 2009) is an acquisition tool employed by many digital forensic and incident response (DF/IR) analysts. Along with the regular version, AccessData also provides a Lite version, which is often used to conduct acquisition of live systems via some type of removable media such as CD-ROM or external USB drive. FTK Imager Lite provides the same functionality as the FTK Imager with the same version number; however, the Lite version has been compiled so

that its functionality is portable. With the introduction of FTK Imager version 2.6, both of these tools have the ability to capture the memory of a live Windows system in addition to providing read-only access to physical drives and hard drive images. As with all tools, the DF/IR analyst must understand how this tool functions to employ it properly and to ensure that using the tool does not compromise the system or the data being analyzed.

The Preview Pane

One feature of FTK Imager that must be closely monitored is how it is configured to display file data. AccessData refers to the file display feature as the "Preview Modes." Depending on which "Preview Mode" is selected (see figure 1) will determine how the file data is displayed in the Preview Pane (lower right pane in figure 1) of the tool's primary window. These modes include (AccessData - 2, 2009);

Automatic mode automatically chooses the best method for previewing a file's contents.

Text mode allows you to preview a file's contents as ASCII or Unicode characters, even if the file is not a text file.

Hex mode allows you to view every byte of data in a file as hexadecimal code.

The default configuration for FTK Imager is set to use "Automatic mode." Per the user guide, this means that FTK Imager will determine the best method to display the information contained within the selected file. For text-based files, executables, or dynamically-linked library files, this mode works perfectly well. These files are displayed in text or hex mode. However, when web pages, graphics, and other files "for which Internet Explorer plugins have been installed" are opened in the Preview Pane, these files "are displayed by an embedded version of Internet Explorer in the Viewer." (AccessData - 2, 2009)

The ability of FTK Imager to display web-based files and graphics might not seem like a problem. In fact, this feature can be very helpful during analysis. But weighed with facts such as the Internet Explorer vulnerability detailed by Microsoft Security Bulletin MS-09-054

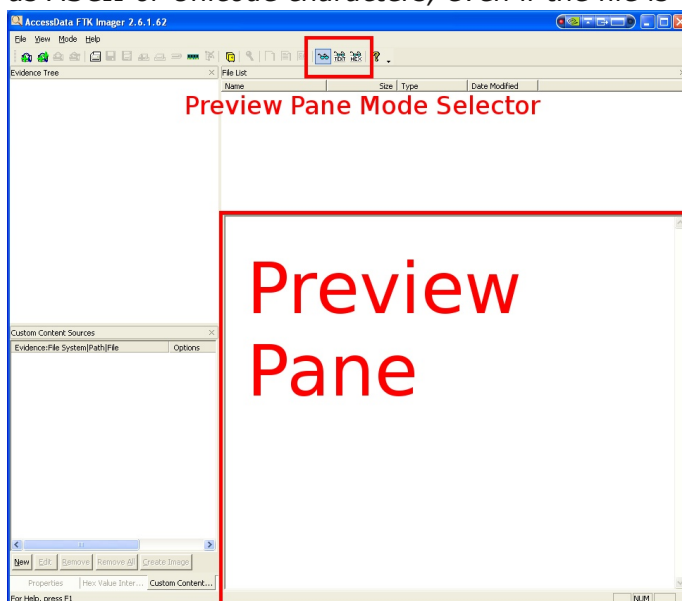


Figure 1: FTK Imager user interface

(Microsoft, 2009) the rendering of web pages from compromised systems should be a concern. Other files and embedded scripting languages, such as Javascript, Adobe Flash, Adobe Portable Document Format (PDF), should also be a concern because of the fact that Internet Explorer uses plugin capabilities to attempt to display these files and therefore the system is subjected to any vulnerabilities or functionality associated with these applications. But perhaps the most concerning aspect of Automatic mode is the fact that it will attempt to connect to remote systems without notifying the user. When rendering web pages FTK Imager will attempt to acquire images, Javascript, and any other embedded functionality contained within the file from the remote servers.

The following images demonstrate how FTK Imager displays an HTML file in Text and then Automatic mode. A simple HTML file that calls an image from a web server located on the Internet is accessed using FTK Imager. When accessed in Text mode the HTML is displayed and can be analyzed for specific functionality, as illustrated in figure 2. Figure 3 demonstrates what occurs when the same file is accessed in Automatic mode. Notice that the image is displayed for analysis.

Although unlikely, it is possible that FTK Imager is displaying this image from the analysis system's Internet cache. To validate that requests for the image were being generated while FTK Imager was displaying the HTML file, Wireshark was used to capture network traffic from the analysis system's network interface and detect any network request and responses. Figure 4 shows the HTML requests and replies associated with the image displayed in FTK Imager's Preview Pane when the HTML file is viewed in Automatic mode.

To help provide a better example of why analysts should be concerned about this functionality a web page was generated demonstrating several capabilities. Figure 5 demonstrates the following web browser capabilities.

Javascript to grab the UserAgent of the browser integrated into FTK Imager. This code resulted in the value "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; Trident/4.0; .NET CLR 2.0.50727; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)".

Javascript code from the MaxMind GeoIP Javascript Web Service (MaxMind, 2009) to determine the physical location of the analyst system. Although not particularly precise, the general location of the system was correctly determined to be located in Corpus Christi, Texas.

Iframe functionality to display an image from an Internet-based webserver.

Although leaking this type of information is a concern, there is a much bigger concern with FTK Imager providing the Automatic mode as the default mode for the Preview Pane. Specifically, Javascript, Iframe, and other functionality provided by the embedded Internet Explorer can be leveraged to compromise the analysis system. Security Focus provides several javascript-related exploit proof-of-concept examples (SecurityFocus, 2009) to exploit the vulnerability outlined in Microsoft Security Bulletin MS-09-054. Javascript and Iframe security concerns are outlined by

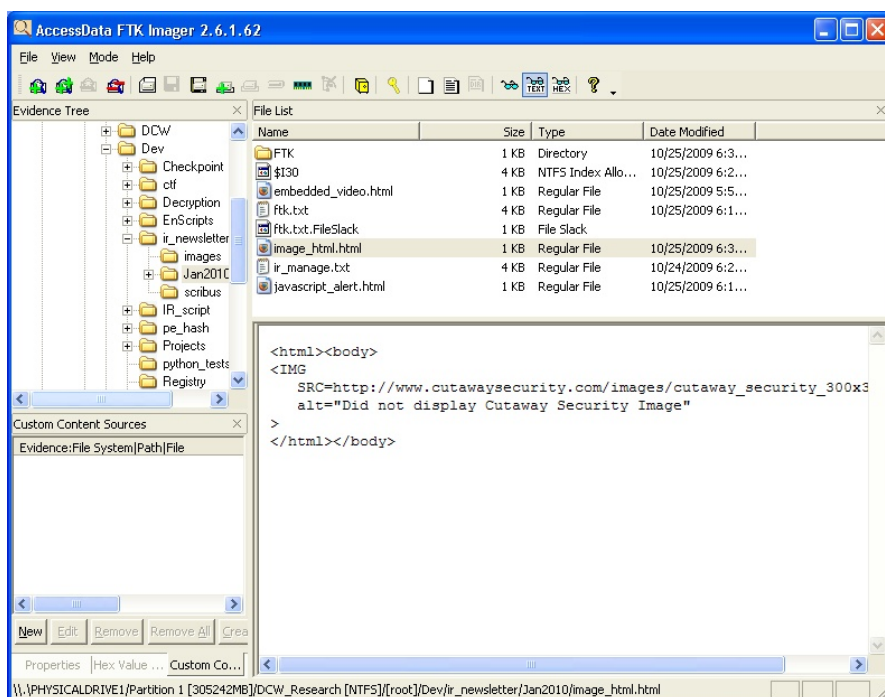


Figure 2: HTML file accessed in Text mode



Figure 3: HTML file accessed in Automatic mode

Gareth Heyes in his blog The Spanner (<http://www.thespanner.co.uk/>). His post "Iframes Security Summary" (Heyes, 2007) details and provides proof-of-concept code where Javascript and Iframes are leveraged to exploit several vulnerabilities including scanning internal networks.

Conclusion

All operating systems and applications have their own idiosyncrasies and quirks. Tools designed for DF/IR are no different, and its critical that analysts understand the strengths and weaknesses of their tools. In order to use these tools safely and to not threaten the integrity of the analysis system or the data being analyzed, the DF/IR analyst must understand the overall capabilities of the tool. There are several things that can be done to protect against these threats while using FTK Imager. The first is to ensure that the analysis system does not have an active Internet connection while reviewing data using FTK Imager. The second protection measure is to immediately change the configuration of the Preview Pane to utilize the Text or Hex mode instead of the default Automatic mode before accessing any data. Implementing these protections automatically every time FTK Imager is implemented takes awareness and discipline on the part of the DF/IR analyst. Another protection mechanism would be to require, as a matter of policy, that the analysis system has a firewall product installed that is configured to explicitly block all outbound connections from "FTK Imager.exe". Finally, FTK Imager might be a better tool if the developers at AccessData had decided to modify the default capability so that Automatic mode is not the default mode for the Preview Pane.

References

AccessData. (2009, October). Product downloads. Retrieved from <http://accessdata.com/downloads.html>

AccessData.- 2 (2009, October). FTK Imager User Guide. Retrieved from http://accessdata.com/downloads/media/en_us/print/manuals/ImagerUsersGuide.pdf

Microsoft. (2009, October 13). Microsoft security bulletin MS09-054 - critical. Retrieved from <http://www.microsoft.com/technet/security/bulletin/ms09-054.msp>

MaxMind, . (2009, October). Geoip javascript web service. Retrieved from http://www.maxmind.com/app/javascript_city

SecurityFocus. (2009). Microsoft Internet Explorer Uninitialized Memory Remote Code Execution Vulnerability. SecurityFocus. Retrieved (2009, October 26) from <http://www.securityfocus.com/bid/33627/exploit>

Heyes, G. (2007). IFrames security summary. The Spanner. Retrieved (2009, October 26) from <http://www.thespanner.co.uk/2007/10/24/iframes-security-summary/>

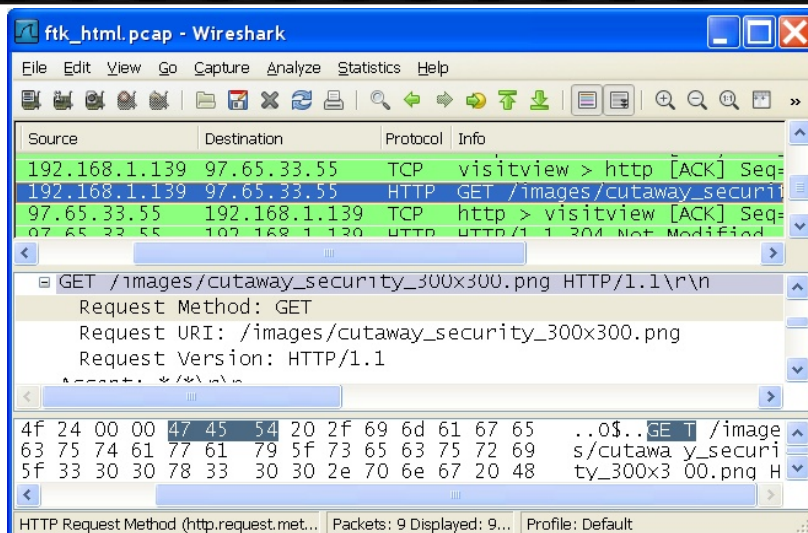


Figure 4: Network traffic capture displayed in Wireshark

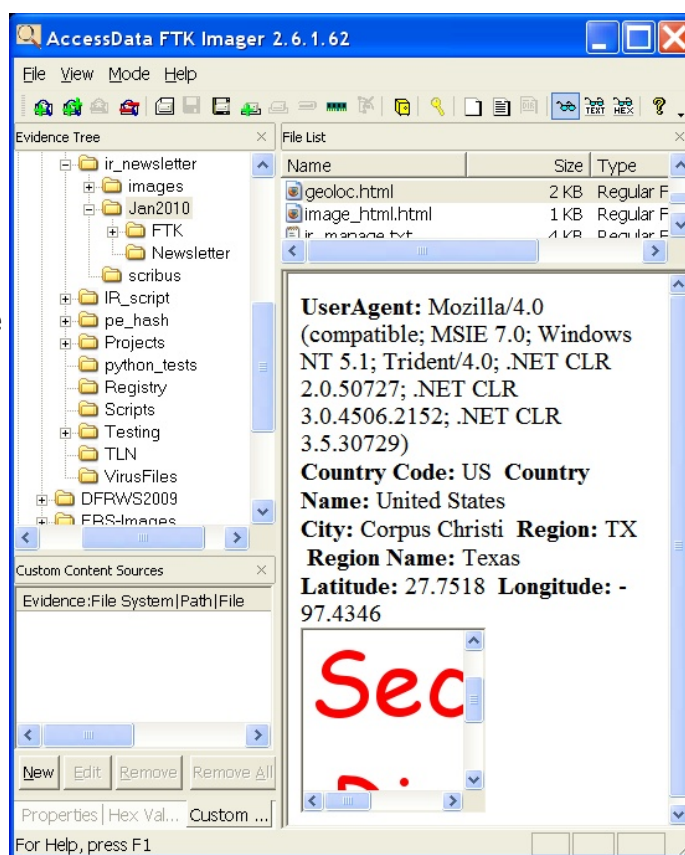


Figure 5: Multiple remote capabilities illustrated in Automatic mode

PCI Interview with Harlan Carvey

Interview Questions by Don C. Weber



How long have you been conducting digital forensics investigations and how long have you been conducting PCI-specific engagements?

I've been conducting forensics investigations for over 9 years, and I had conducted PCI-specific engagements for a bit more than 2 years while I was with IBM ISS.



Harlan Carvey

What type of experiences do security professionals need to work for a company that conducts PCI-related digital forensic investigations?

Aside from the QSA certification, there are no other requirements mandated by Visa or the PCI Council.

Do you receive more calls to conduct digital forensic investigations from companies who have completed their PCI-audits or non-audited companies?

My experience with the IBM ISS team, while they remained on the QIRA list, was that most of the organizations to which I responded were not PCI compliant.

How well are PCI-compliant companies implementing and integrating incident response teams, policies, and procedures into their day-to-day business activities?

I have a hard time answering this question, as most of the organizations I responded to weren't PCI compliant. In fact, some of them stored and/or processed card holder data, and hadn't heard of "PCI"; mentioning it brought rounds of blank stares.

What are the most important recommendations you can make to an business to help them reduce the scope and impact of a PCI-related investigation before it occurs?

Understand the nature of your business, and understand...to an extremely fine, extremely granular degree...where PCI data lives and how it is used within your infrastructure. The PCI Data Security Standard (DSS) is not so much a set of compliance requirements that must be met, as they are a roadmap to minimum, common sense security measures. The vast majority of the PCI engagements I participated in or conducted started with the "deer in the headlights" look from the "victim" staff when I asked, "Where is your PCI data?"

In the few instances where proactive measures had been taken, the organization was able to clearly illustrate to Visa and the PCI Council what had occurred and to what degree data had been compromised, if at all. In those very few cases, the "victim" organization most often received a small fine for possessing card holder data, rather than suffering larger losses due to that data being exposed.

Which recommendations do you make to businesses most often at the conclusion of your PCI-related investigations?

Most recommendations amounted to repeating the PCI DSS. The PCI DSS isn't about security...it's about compliance. Often, the step up from where an organization is to some modicum of security is too big, and what's needed is that intermediate step up to compliance.

Do PCI investigations usually involve single servers/systems or do they usually end up involving multiple business resources? Please elaborate.

Initially, multiple systems are involved simply because the "victim" organization has no idea where PCI data rests or is in use within their infrastructure. Unencrypted card holder data often flows across an enterprise, and once you start digging into where it flows and where it "rests", the possibilities of which systems are involved in or associated with the incident grow.

In instances where only a small number of systems were involved, the attack was very focused, the intruder was targeting very specific information, and they knew exactly where to find that

data.

How long after an incident is identified do organizations involve digital forensic and incident response experts?

Very often, too long. PCI requirements aren't really very well understood, so there needs to be consultations with attorneys, shopping around for a QIRA firm, etc. Personally, I've responded to a number of instances where the intrusion took place weeks or months before we received our first call, and the "victim" only knew because they received a call from an outside third party.

When it comes to PCI policies, what recommendations would you make to the PCI Council to help companies prepare for PCI-related incident response digital investigations?

The PCI Council isn't really as draconian as most people might think. What the council is trying to do is take an industry that has a deeply embedded culture of not securing card holder data, and moving toward security. This can't be done in one big leap, moving everyone from 0 to 60 at once...that simply won't work.

The PCI DSS is really nothing more than what most of us who've been in the industry for a while would consider "common sense"... "well, sure...yeah...we NEED to protect that data."

Where this approach seems to be unfair is that large organizations that get hit may or may not "suffer" from a fine imposed as a result of a data breach, whereas smaller organizations...mom-and-pop restaurants...will simply disappear...these small organizations don't have \$10K or \$15K sitting around, just in case the PCI Council imposes a fine.

Analysts who work on PCI-related digital forensic investigations have to be QSA certified. Does the current QSA training and certification ensure that analysts conducting these investigations have the proper experience in computer forensics and incident response?

The current, required QSA certification and retraining is geared toward PCI assessments, attesting to the compliance level, and NOT at all directed toward the forensic response engagements. From a reporting perspective, experience has shown that much of what Visa wants to see can (and should) be encapsulated in a template, but the scoping and analysis of the forensic response are the keys. Familiarization or certification in a single product isn't enough...for these types of investigations, some experience is required. Most responder organizations will have a methodology they use for examining systems or acquired images for card holder data (i.e., PANs, track data, etc.), but the toughest questions include, how did the intruder get in and compromise the system, did they expose card holder data, and if so, how much? That's where experience in performing examinations is essential.

Is there a need for a PCI-related training and certification or is this need already fulfilled by certifications such as the EnCase Certified Examiner and the forensic and incident response training and certifications offered by SANS/GIAC?

Most vendor-certified examiners don't bother to question the product's results...which is the really scary part of all this. My experience with some vendor products has shown that there are built-in functions that aren't exposed for review, and yet through testing have been shown to not report data correctly. Vendor-provided training and certification, as well as other sources, such as SANS, are only the beginning, and it's really up to the individual examiner to decide to rest on their laurels and do only what they were shown in training, or to use that training as a basis and foundation and expand from there.

Strictly speaking, there are only a few businesses that are listed on the QIRA list and are authorized to conduct PCI-related digital investigations. How does the PCI Council distribute information from investigations to the different businesses and analysts in these organization to facilitate the distribution of current threat vectors and attack methodologies? Are these methods effective and how can they be improved?

The PCI Council sits at the top of the pyramid, so to speak, with a number of QIRA-certified firms reporting into them. The Council sits as a central collection point for information...information that they could easily turn into very valuable, actionable intelligence. However, prior to my employer dropping from the QIRA list (IBM ISS voluntarily dropped from that list in June, 2009) the only information that we received was a list of file names, with brief

descriptions, sizes and MD5 hashes, and some IP addresses and domain names, and we were told to search every acquired image for these values.

In my opinion, this approach misses a very significant opportunity to really have an impact on those involved in perpetrating data breaches. I've seen files of the same name be uploaded to systems within minutes of each other; the first iteration was picked up by the installed AV product, that the second wasn't. The bad guys change domain names and IP addresses of the systems that they send their collected data to all the time. Sending out static information weeks or months after the fact only makes it look like something is actually being done...the reality is that this is an ineffective approach that only ends up bearing fruit (in the form of fines) against the victim organization. In my opinion, there is an extraordinary opportunity here for a flow and exchange of intelligence information to have a significant impact on these types of crimes, but that opportunity is being missed.

Biographies

Jamie Levy

Jamie Levy is currently an EnScript Programmer and Consultant at Guidance Software. She has taught classes in Computer Forensics and Computer Science at Queens College (CUNY) and John Jay College (CUNY). She has an MS in Forensic Computing from John Jay College and is supporter of the open source Computer Forensics community. Additional technical articles and blog posts by Jamie can be found at <http://gleeda.blogspot.com>.

Didier Stevens

Didier Stevens (CISSP, GSSP-C, MCSD .NET, MCSE/Security, RHCT, OSWP) is an IT Security Consultant currently working at a large Belgian financial corporation. He is employed by Contraste Europe NV, an IT Consulting Services company (<http://www.contraste.com>). You can find open source security tools on his IT security related blog at <http://blog.DidierStevens.com>.

Don C. Weber

Don C. Weber (CISSP, GSNA-G, GCIH, GCUX, GSEC-G, GAWN) has been blogging about technical and managerial security issues at the Security Ripcord (<http://securityripcord.com>) since March 21st, 2006. Don is currently employed in the Digital Forensics and Incident Response industry where he utilizes his background in system architecture, network security, security management, technical research, and security tool development to provide leadership and capable technical insight during stressful and challenging information security situations.

Harlan Carvey

Harlan Carvey (CISSP), author of the acclaimed Windows Forensics and Incident Recovery, is a computer forensics and incident response consultant based out of the Northern VA/Metro DC area. He currently provides emergency incident response and computer forensic analysis services to clients throughout the United States. His specialties include focusing specifically on Windows 2000 and later platforms with regard to incident response. Registry and memory analysis, and post mortem computer forensic analysis. Harlan's background includes positions as a consultant performing vulnerability assessments and penetration tests and as a full-time security-engineer. He also has supported federal government agencies with incident response and computer forensic services.

Editors: Harlan Carvey and Don C. Weber

Into The Boxes website: <http://intotheboxes.wordpress.com>