Global Information Assurance Certification Paper

GIAC Certified Forensic Analyst

Practical Assignment (Version 1.5)

Richard S Grime

10th December 2004

Part 1 – Analyze an Unknown Image

David Keen, the security administrator for Ballard Industries, has asked for a forensic analysis of evidence recovered from a floppy disk belonging to Robert Leszczynski. We have been provided with a chain of custody form stating the following information:

- Tag# `fl-260404-RJL1`
- 3.5 inch TDK floppy disk
- MD5: `d7641eb4da871d980adbe4d371eda2ad fl-260404-RJL1.img`
- fl-260404-RJL1.img.gz

The evidence has been delivered to us by means of a web link to the GZip'd image noted at the bottom of the above form.

Verify the image

Our first task is to demonstrate that the evidence we are working on is in fact the same image collected by David Keen – i.e. demonstrate that the chain of custody is intact.

Before beginning the analysis, it is important to remember to keep a time-stamped chronological log of our activities. This will form the basis of this report, as well as providing us with a record of our activities should we be asked to recall them at a later date (e.g. in court).

The file as retrieved from the web server is named **v1_5.gz**. We need to demonstrate that this is the equivalent of the expected **fl-260404-RJL1.img.gz.** It is also important to show that the file has not been tampered with, and that the content is as expected.

After using wget to retrieve the file, we take some basic precautions to ascertain what the file is:

```
icsecl2 ictsec # ls -la v1_5.gz
-rw-r--r--  1 root root 502408 May 20 18:34 v1_5.gz

icsecl2 ictsec # md5sum v1_5.gz
f39239ed04e7c0c1b36bcd556d213623  v1_5.gz
```

Here we use the standard Unix "ls" command to list information held about the file in the associated inode, principally ownership and size. Using md5sum we initiate the chain of custody for the evidence we collect. By documenting both the steps taken during the analysis and recording the associated MD5 sums, we can demonstrate the integrity of the evidence and ensure our processes are repeatable.

We are assuming that the "ls" and "md5sum" binaries are trusted – that is, they are performing as expected. For this reason, we must also be able to demonstrate that the system we are performing the forensic analysis on is

trustworthy. Forensic analysis systems are often installed on air-gapped networks with no direct internet connection.

Under UK law, there is a rebuttable presumption[i] that such systems are operating as expected. However, should the defence produce evidence to the contrary, the prosecution must be able to demonstrate that the systems are operating correctly.

This MD5 sum is different from the one states on the chain of custody form, because this belongs to the gzip'd version of the file, rather than the image file itself.

```
icsecl2 ictsec # md5sum v1_5.gz > v1_5.gz.md5

icsecl2 ictsec # file v1_5.gz
v1_5.gz: gzip compressed data, was "fl-260404-RJL1.img", from Unix
```

We save the MD5 sum of the gzip archive, and then use the file command to determine whether the file has the headers we expect for a gzip archive. It seems it does.

```
icsecl2 ictsec # gunzip -l v1_5.gz
         compressed        uncompressed  ratio uncompressed_name
             502408             1474560  65.9% v1_5

icsecl2 ictsec # gunzip -c v1_5.gz | md5sum
d7641eb4da871d980adbe4d371eda2ad  -
```

Without writing anything potentially dangerous to our analysis system, we can check the MD5 sum of the contents. N.B. this will of course only work if the archive contains one file. The previous line (`gunzip -l`) listing the contents can be used to confirm this. By using `gunzip -c` to unzip to console, then piping to md5sum, we can retrieve the MD5 sum on the fly – as we have shown there is only one file in the archive, this second command has recovered a file with the same MD5 sum as given by David Keen on his original chain of custody form.

We have now demonstrated that the evidence has been passed on successfully to ourselves in an integral manner – precisely what is needed for "chain of custody".

Our written log should indicate the date and time at which this evidence was received for analysis as well as who / where we obtained it from, the names of the handlers performing the analysis and a description of the evidence in question.

The next step is to extract the image file from the archive, taking care to preserve any meta-data relating to the image file itself. Using gunzip's "-N" option, we preserve ownership and timestamp information when extracting.

```
icsecl2 ictsec # gunzip -Nd v1_5.gz
```

[i] http://www.hse.gov.uk/enforce/enforcementguide/investigation/physical/preparing.htm#P27_3980

```
icsecl2 ictsec # ls -l fl-260404-RJL1.img
-rw-r--r--  1 root root 1474560 Apr 26 01:45 fl-260404-RJL1.img

icsecl2 ictsec # md5sum fl-260404-RJL1.img
d7641eb4da871d980adbe4d371eda2ad  fl-260404-RJL1.img

icsecl2 ictsec # md5sum fl-260404-RJL1.img > fl-260404-RJL1.img.md5
```

We now have the original image file, `fl-260404-RJL1.img`. Using MD5sum, we can prove that this is the same image that David Keen originally added to the archive. Finally, we take a copy of the MD5 sum to a file.

```
icsecl2 ictsec # file fl-260404-RJL1.img
fl-260404-RJL1.img: x86 boot sector, code offset 0x3c, OEM-ID "
mkdosfs", root entries 224, sectors 2872 (volumes <=32 MB) ,
sectors/FAT 9, serial number 0x408bed14, label: "RJL        ", FAT
(12
bit)
```

Happily, the file command would suggest we have an image of a floppy disk, so we can begin analysis.

Note: we cannot prove that this is an image of the actual floppy disk removed from Robert Leszczynski – the burden of proof for this lies with David Keen, and any witnesses he has to the actions he took. In this sense, we are only forming a part of the "chain of custody". Without knowledge of how this image was obtained, any evidence we provide is hearsay. Our "expert" evidence is only admissible in a UK court of law if David Keen has first given evidence[ii] (either to the court directly, or via a statement).

Analysing the Image

Now that we have what we believe to be a floppy image, analysis can begin. However, it is important to remember that the evidence collected will only be as sound as the methodology and tools used to collect it.

As such, we must ensure that the system used for image analysis is properly configured. Installing the system "off-net" means we can demonstrate that the evidence collected has not been compromised remotely (and means we also need to ensure good physical security for the system).

Off-net has its own problems, such as not being able to synchronise with a network time server, and difficulty sharing evidence with other analysts. Ideally, we would like an analysis network with its own time servers, DNS servers, etc.

The key is repeatability – if we present a full history of how the evidence was gathered, rather than just the end product then in theory anyone can reproduce the same end result.

[ii] http://www.hse.gov.uk/enforce/enforcementguide/investigation/expert/court.htm#P7_1242

The analysis system used to produce this results was installed off-net, brought up with a default deny firewall in place then synchronised with an NTP server to ensure timestamps on evidence we generate are reliable.

We have already used `file` to ascertain some basic information about the image file. Using `fsstat [1]` we can get some more detail (we know the file system type to be FAT from the output of `file`):

```
icsecl2 ictsec # fsstat -f fat fl-260404-RJL1.img
FILE SYSTEM INFORMATION
--------------------------------------------
File System Type: FAT

OEM Name: mkdosfs
Volume ID: 0x408bed14
Volume Label (Boot Sector): RJL
Volume Label (Root Directory): RJL
File System Type Label: FAT12

Sectors before file system: 0

File System Layout (in sectors)
Total Range: 0 - 2871
* Reserved: 0 - 0
** Boot Sector: 0
* FAT 0: 1 - 9
* FAT 1: 10 - 18
* Data Area: 19 - 2871
** Root Directory: 19 - 32
** Cluster Area: 33 - 2871

METADATA INFORMATION
--------------------------------------------
Range: 2 - 45426
Root Directory: 2

CONTENT INFORMATION
--------------------------------------------
Sector Size: 512
Cluster Size: 512
Total Cluster Range: 2 - 2840

FAT CONTENTS (in sectors)
--------------------------------------------
105-187 (83) -> EOF
188-250 (63) -> EOF
251-316 (66) -> EOF
317-918 (602) -> EOF
919-1340 (422) -> EOF
1341-1384 (44) -> EOF
```

The OEM Name given by fsstat suggests that the disk was formatted under a Linux variant operating system. The FAT content areas are also continuous; meaning any attempts to hide raw data must be outside the content areas. More importantly, the fsstat information gives us enough information to load the image successfully into Autopsy [2].

After creating a new case in Autopsy, we need to add a new host. In this case, we only have one image – so the floppy disk is in effect our host.

We are told that the local time zone for the R&D labs is MST – which gives us an Autopsy time zone string of MST7MDT[iii]. There's no other information on

[iii] Lots more information on various time zones is available at http://wwp.greenwichmeantime.info/

time skew, because we don't (yet) know which systems Leszczynski may have used. As the media in question is portable, the time stamps may vary from system to system. This is worth bearing in mind when looking later at the MACTime information.

Once the new host is added, we need to add the image. We have gathered most of the information required by Autopsy. We can copy the image into the evidence locker, as we are only dealing with a floppy image. The file system type has been identified as FAT12 from the fsstat information gathered above.

We have guessed from the OEM Name property that the disk was formatted in a Linux machine, so we can guess a mount point of /mnt/floppy/. This is not of critical importance, and we can revise this later on.

Create a timeline

The next step in the analysis is to produce a timeline of file activity on the disk. This is a simple process in Autopsy – which will also automatically generate MD5 sums to ensure integrity.

An initial look at the time line shows us that the mount point is more likely to be **a:**, as the disk appears to contain files named in a Windows-style format.

File Name	Size (bytes)
_ndex.htm	727
CamShell.dll	36864
Acceptable_Encryption_Policy.doc	22528
Information_Sensitivity_Policy.doc	42496
Internal_Lab_Security_Policy.doc	33423
Internal_Lab_Security_Policy1.doc	32256
Password_Policy.doc	307935
Remote_Access_Policy.doc	215895

Table 1 - Listing of Files in Image

Appendix 1 shows the collected timeline. The first date in the listing is anomalous for our purposes – deleting a file (such as CamShell.dll) does not affect the modified flag. As such, this shows us the last time those files were modified before they were deleted. For a DLL type file, this would indicate the last time the library was compiled, and we would expect this date to match the timestamp on the distribution of whichever package includes CamShell.dll.

The separation of the files into distinct groups by flag, e.g. all the m(odified), then a(ccessed) then c(reated), is also distinctive.

On a disk that was being worked on, you would expect to see a more random order. This implies that this disk is merely a copy of files that were prepared on another system. It is therefore important that the system administrators at Ballard track systems that Leszczynski had access to as he may have been preparing further material not held on this floppy image.

The “c” flag shows us that the files were copied to this floppy on Monday 26th April, 2004. The copy was initiated at 09:46:00 and completed at 09:46:44. As the floppy disk was recovered by the staff security guard at 4:45 on the 26th of April, it would appear that this disk had not yet reached its recipient. The separation between the timestamps (of the order of seconds) indicates that the files were copied in bulk, rather than individually. Again, this supports the argument that the files were prepared in advance on the system.

Last access times (“a“ flag) shows Monday April 26th 2004 at 00:00:00. A timestamp like this is suspicious – it would appear that the last access time has been reset by some application which either did not support timestamps, or was deliberately told to use 00:00:00. It is possible that the utility used to copy the files modified the access timestamp on the file.

A FAT based file system does not have user based access control, so as such the concept of a file owner (user / group) does not apply in this case. As such, the timeline analysis shows a UID and GID of 0 (root). This, however, does not mean that Leszczynski was operating at root level.

There is of course a major caveat with timeline analysis. The time stamps on a file are trivially modifiable. Even without malicious intent, the timestamps are reliant on the local system clock. As such, timestamps are of little use when considered independently of other evidence – they can be used to demonstrate actions, but not necessarily prove them. For instance, if Leszczynski has an alibi for the times in question (Friday 23rd April or Monday 26th April) then these timestamps should not be considered proof that he could not have created the files.

Recovering Files

There are no immediately obvious executables on the disk, so to begin analysis we can start with what are assumed to be word documents. Using Autopsy, the six .doc files can be extracted. We can use file again to check what headers they have:

```
icsecl2 recovered # file fl-260404-RJL1.img-mnt* | cut -d"." -f5-
Acceptable_Encryption_Policy.doc..ACCEPT.1.DOC.:   Microsoft Office Document
CamShell.dll._AMSHELL.DLL.:                        HTML document text
Information_Sensitivity_Policy.doc..INFORM.1.DOC.: Microsoft Office Document
Internal_Lab_Security_Policy.doc..INTERN.2.DOC.:   Microsoft Office Document
Internal_Lab_Security_Policy1.doc..INTERN.1.DOC.:  Microsoft Office Document
Password_Policy.doc..PASSWO.1.DOC.:                Microsoft Office Document
Remote_Access_Policy.doc..REMOTE.1.DOC.:           Microsoft Office Document
_ndex.htm:                                         HTML document text
```

N.B. `cut` used to trim file names to fit page.

Viewing these documents using strings reveals them to be internal policy documents. None of the documents include a **distribution** section, or comments on the sensitivity of these documents. The Information_Sensitivity_Policy for Ballard Industries states that the sensitivity markings are optional. Hence, we cannot tell from this whether Leszczynski has in fact breached company policy.

There appears to be two versions of the same document – Internal_Lab_Security_Policy.doc and Internal_Lab_Security_Policy1.doc. The former is 1167 bytes larger than the latter. There is also the suggestion that the larger document was created after the smaller; the short file name compatibility (i.e. 8.3 format) used by Windows abbreviates the filename to 6 characters, and then appends a tilde and numeral. The numeral is used to ensure uniqueness, and is usually assigned in order of creation.

We can take a diff of the two files to discover where these extra bytes are being used.

```
icsecl2 recovered # diff -a \
fl-260404-RJL1.img-mnt.floppy..Internal_Lab_Security_Policy.doc..INTERN.2.DOC. \
fl-260404-RJL1.img-mnt.floppy..Internal_Lab_Security_Policy1.doc..INTERN.1.DOC. \
 | strings
26,29c26
Title
Ballard Industries, Inc.
G<\cB
>viV    /9
&[p
Q2fD
< M~

                          K

          8

                                  t

\ No newline at end of file
Title
Ballard Industries, Inc.
\ No newline at end of file
```

The diff shows that a chunk of data has been added at the end of the file. Microsoft Word documents have fairly indecipherable formats – so this could just be some document meta-data. We can have a look at the larger document in a Hex editor, which may reveal more. The extra chunk of data comes after a block of white space, and a normal Word document ends with the document title.

It is possible that Leszczynski has used steganographic techniques to hide commercially sensitive data. If this is the case, we need to discover the nature of the tool used to create the documents so that we can extract the hidden data. As the only program on the disk is the library CamShell.dll – this would seem a good place to start.

Before we can begin analysis of CamShell.dll, we first need to recover it. It appears that the file _ndex.htm has been used to overwrite the start of it. We can see this using `istat [1]`, which shows that the directory entry for _ndex.htm and _AMSHELL.DLL point to the same disk sector, and that the timestamps for _ndex.htm are later than those for _AMSHELL.DLL.

```
icsecl2 images # istat -f fat12 fl-260404-RJL1.img 28
```

```
Directory Entry: 28
Not Allocated
File Attributes: File, Archive
Size: 727
Num of links: 0
Name: _ndex.htm

Directory Entry Times:
Written:        Fri Apr 23 10:53:56 2004
Accessed:       Mon Apr 26 00:00:00 2004
Created:        Mon Apr 26 09:47:36 2004

Sectors:
33

Recovery:
33 34

icsecl2 images # istat -f fat12 fl-260404-RJL1.img 5
Directory Entry: 5
Not Allocated
File Attributes: File, Archive
Size: 36864
Num of links: 0
Name: _AMSHELL.DLL

Directory Entry Times:
Written:        Sat Feb  3 19:44:16 2001
Accessed:       Mon Apr 26 00:00:00 2004
Created:        Mon Apr 26 09:46:18 2004

Sectors:
33

Recovery:
33 34 35 36 37 38 39 40
41 42 43 44 45 46 47 48
49 50 51 52 53 54 55 56
57 58 59 60 61 62 63 64
65 66 67 68 69 70 71 72
73 74 75 76 77 78 79 80
81 82 83 84 85 86 87 88
89 90 91 92 93 94 95 96
97 98 99 100 101 102 103 104
```

The underscore at the start of this name arises from the way FAT marks files as deleted. Hence, the file has probably been overwritten by downloading the index.htm file from a local web server and saving it to the floppy. This could be used to demonstrate intent to cover up activity on the disk – but could equally be argued to be entirely accidental. However, under normal circumstances, files would only start being overwritten once the disk is full – which this floppy is not.

However, performing a strings analysis on the remainder of the file gives us some more clues (the full output of strings has been snipped for length, only interesting strings shown).

```
ll\SheCamouflageShell
ShellExt
VB5!
CamShell
BitmapShellMenu
CamouflageShell
CamouflageShell
Class
C:\WINDOWS\SYSTEM\MSVBVM60.DLL\3
VBRUN
FIShellExtInit
C:\My Documents\VB Programs\Camouflage\Shell\IctxMenu.tlb
```

The product appears to have been written in Visual Basic, and is called Camouflage. A Google search shows that the software can be downloaded from http://www.camouflage.freeserve.co.uk. Installing it gives us the CamShell.dll in its original form (we hope!).

The same web page describes the program as a steganographic tool, which matches our hypothesis.

```
C:\Program Files\Camouflage>dir
 Volume in drive C is RGPC
 Volume Serial Number is EC78-C225

 Directory of C:\Program Files\Camouflage

30/07/2004  16:08    <DIR>          .
30/07/2004  16:08    <DIR>          ..
29/03/2001  22:13           217,088 Camouflage.exe
03/02/2001  19:44            36,864 CamShell.dll
28/03/2001  19:50            11,649 Readme.txt
30/07/2004  16:08            19,758 Uninst.isu
               4 File(s)        285,359 bytes
               2 Dir(s)   1,009,537,024 bytes free

C:\Program Files\Camouflage>md5sum CamShell.dll
4e986ab0909d2946bed868b5f896906f *CamShell.dll
```

The problem is that the start of our CamShell.dll has been overwritten. We can use Autopsy to extract what's left of CamShell.dll. Knowing that the start has been overwritten with the 767 byte _ndex.htm file, we can use `tail` to cut the first 767 bytes from the file – doing the same with the version of CamShell.dll downloaded from the internet.

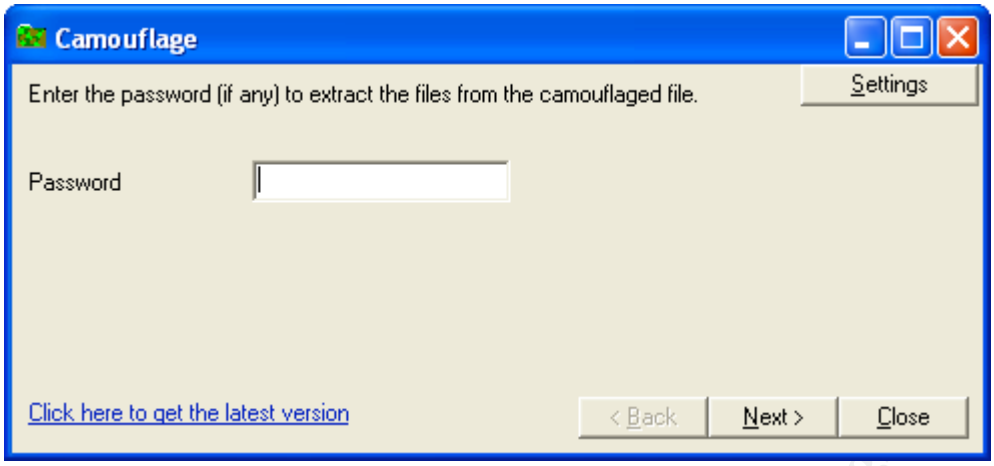
```
icsecl2 recovered # tail -c `expr 36864 - 767` \
v1_5-a...CamShell.dll._AMSHELL.DLL. > camshell-recovered.dll
icsecl2 recovered # tail -c `expr 36864 - 767` CamShell.dll > camshell-orig.dll

icsecl2 recovered # md5sum camshell-*
3fc80112adabfe4d3277938d4e679a7f  camshell-orig.dll
3fc80112adabfe4d3277938d4e679a7f  camshell-recovered.dll

icsecl2 recovered # ls -l camshell-*
-rw-r--r--  1 ictsec users 36097 Oct 25 09:11 camshell-orig.dll
-rw-r--r--  1 ictsec users 36097 Oct 25 09:11 camshell-recovered.dll
```

Because the start of the file has been lost, we cannot demonstrate that the recovered CamShell.dll has a matching MD5 sum with the version downloaded from the internet. However, we can demonstrate that the remaining 36097 bytes are identical to the version downloaded. That is, we can be 98% certain that the CamShell.dll on the disk had a matching MD5 sum with the downloaded version.

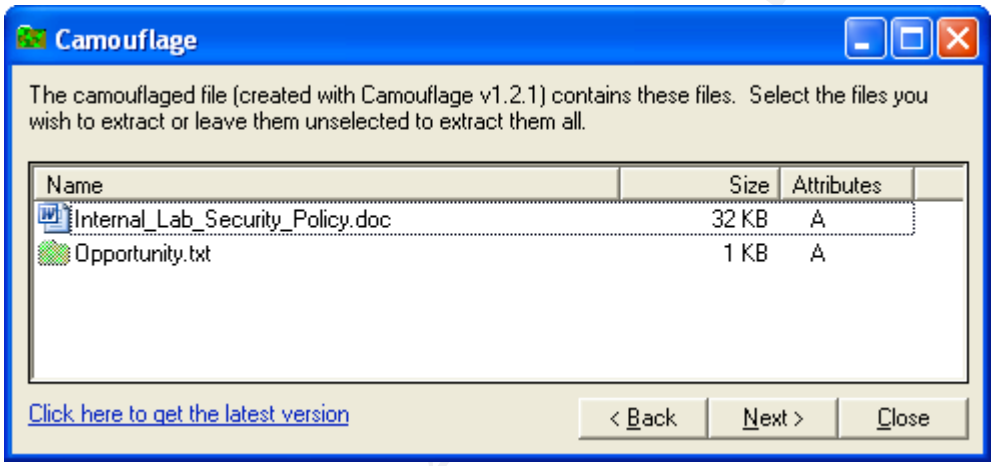
Of course, much of the functionality of the DLL is not in the first 767 bytes – so we can say that the DLL had steganographic capabilities, which were very similar, if not identical, to the downloaded version of Camouflage.

Now that we have identified a potential candidate for the tool – we can attempt to recover data from the first document we believe has hidden content, namely Internal_Lab_Security_Policy.doc

Running Camouflage on this file prompts:



This could be complicated – however the "(if any)" gives a glimmer of hope. Trying a blank password appears to work, and we get:



It appears the file **Opportunity.txt** was hidden in the document.

```
C:\forensics>md5sum Opportunity.txt
3ebd8382a19c88c1d276645035e97ce9 *Opportunity.txt

C:\forensics>dir Opportunity.txt
23/04/2004  20:03               312 Opportunity.txt

C:\forensics>type Opportunity.txt
I am willing to provide you with more information for a price.   I have included
 a sample of our Client Authorized Table database.  I have also provided you wit
h our latest schematics not yet available.  They are available as we discussed -
 "First Name".
My price is 5 million.

Robert J. Leszczynski
```

This is our first piece of evidence of actual intent. It also suggests that other data has either been already sent over, or is hidden in the other files on this disk image.

We know that Camouflage adds its information at the end of the document, and also that two of the documents appear very large for the amount of content that they actually contain. Trying camouflage against these other two documents (Remote_Access_Policy.doc and Password_Policy.doc) with a

blank password fails. However, the above recovered document suggests that extra information has been included – namely a client database, and some schematics.

Because Camouflage creates a "new" file containing the hidden data, we can infer the last time the program was run; namely Friday April 23rd 2004 at 11.54.32 (for Remote_Access_Policy.doc) and 11.55.23 on the same date for Password_Policy.doc , as this is the modified date on the camouflage files (see **Create a Timeline** and Appendix 1).

However, this only demonstrates when the camouflaged files were last modified – not necessarily the last time they were written to by *Camouflage*. Using Word to write to the files would remove the camouflaged section of the file, but it is trivial to modify the modification date on a file.

Understanding Camouflage Encryption

Camouflage uses a basic form of steganography to hide data inside one another. While more complex tools can use techniques such as placing data in the least significant bit of a pixel in an image, Camouflage is much less elegant.

Camouflage appends the "hidden" data in an encrypted form on the end of the existing document. As such, the presence of Camouflaged information is obvious – provided you are looking for it! Because Leszczynski left a copy of the original Internal_Lab_Security_Policy.doc on the disk, this is almost the steganographic equivalent of leaving your plain and cipher texts in the same place.

By running some basic checks, we can assess the level of encryption that Camouflage uses. Camouflage will not encrypt empty documents, so we create two documents, both text files with one character in them – the numeral one (1). We use Camouflage to take 1.txt, hide the data in 2.txt and create the output file 3.txt. The password for this is set to "a". To test password strength, we create another document, 4.txt, which is identical save for having a password of "aa".

Hex analysis of the files show that Camouflage creates a large series of hex value "20". A diff between the two files suggests that the encrypted password has a one to one mapping of plaintext value to cipher value. Repeated encryptions show that the cipher values are also always the same. This means that the encryption is weak – as this is a password with no seed based encryption values (e.g. using the content of the host document to seed the encryption key).

Further tests using different passwords show us that the encrypted password characters do not vary with position, nor are they dependant on other characters in the password. This means that the password can be cracked bitwise, and is likely to be encrypted using a simple transformation function, like XOR.

At offset 246H, 3.txt has the value 63; 4.txt has two values 63 F4. By creating files with longer passwords[iv], we can ascertain that these correspond to "a" and "aa" respectively.

Looking at the two recovered Word documents, we find the encrypted password strings of:

Password_Policy.doc	52 F4 09 51 7B C9 66 85
Remote_Access_Policy.doc	50 F0 17 4D 78 C3

So we know that the first document has an 8 character password, the second has 6. One option at this point would be to replace the password strings with a known value (e.g. create a Camouflaged document with the password "aaaaaaaa" and "aaaaaa", and then paste these in). If however Camouflage employs any form of checksum, this could break the document. Also, modifying the evidence is not a particularly forensically sound idea.

If however, we create a Camouflage document with the password "aaaaaaaa", we get the password string:

```
63 F4 1B 43 6D C7 75 80
```

This corresponds to the hex string:

```
61 61 61 61 61 61 61 61
```

The cryptographic techniques employed by Camouflage so far seem to be fairly basic. The random nature of the encrypted password string shows that a rotation cipher is not being used. We now need to discover the transformation function used by Camouflage, or in more mathematical notation:

$$y = f(x)$$

Where y is our cipher text, x is the plain text, and f(x) represents the function that transforms one into the other. Tests on the password have already shown that the password is encrypted bitwise, seemingly without a seed. As postulated above, the function could be XOR, which has been popular in a number of basic encryption routines. This is also supported by the fact Camouflage supports a maximum password length of 255 characters – presumably the maximum size of its internal private key.

$$y = k \oplus x$$

While XOR[v] is strong if we know neither *k* (some private key) nor *x*, the encryption is entirely reversible if (as in this case) we have access to the key or plain text. Because we are able to use the *Camouflage* application to

[iv] 255 character passwords appear to be the limit – after this Camouflage crashes!

[v] The symbol for the XOR operator does not have clear standard. Here we have used $\oplus$.

generate new documents, and then extract the password fields from that document, we can “reverse” the XOR operation to retrieve *k*.

$$k = y \oplus x$$

N.B. Due to the nature of the operation, x and y are commutable.

Using the information gathered above, we can perform the XOR operation:

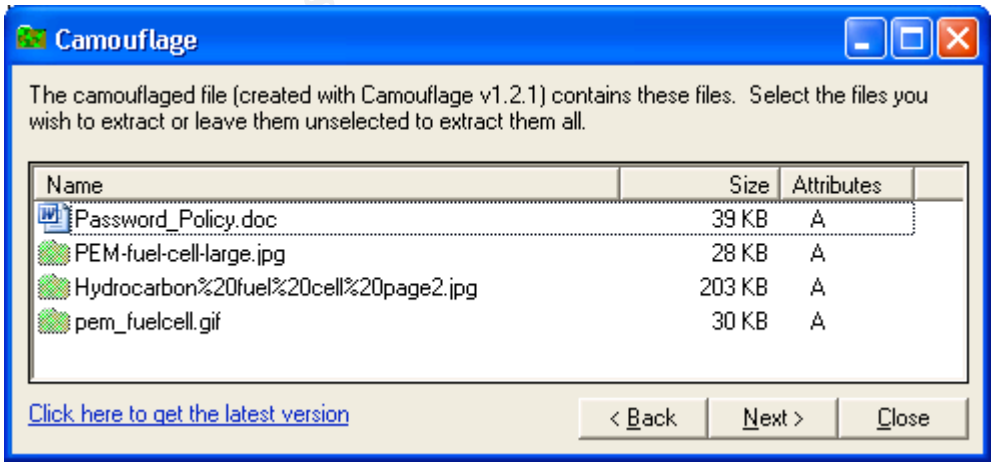
```
63 F4 1B 43 6D C7 75 80
61 61 61 61 61 61 61 61
=======================
02 95 7A 22 0C A6 14 E1
```

This represents the first 8 (of 255) bits of the *Camouflage* private key. We can now try an XOR operation with this key and the password strings recovered from the documents[vi]:

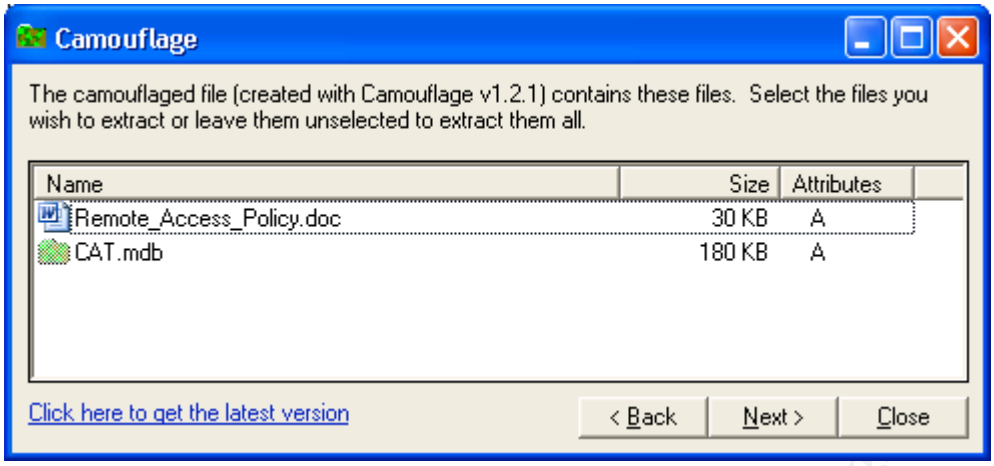
```
52 F4 09 51 7B C9 66 85
02 95 7A 22 0C A6 14 E1
=======================
50 61 73 73 77 6F 72 64
P  a  s  s  w  o  r  d

50 F0 17 4D 78 C3
02 95 7A 22 0C A6
================
52 65 6D 6F 74 65
R  e  m  o  t  e
```

Leszczynski’s clue of “first name” now becomes obvious. The documents are passworded with the first part of the document’s name.

We can now use Camouflage to extract the hidden data from these documents:



[vi] A useful reference at www.asciitable.com for conversion by hand



We will record the MD5 sums and files sizes / date stamps accordingly:

```
C:\forensics>md5sum *.jpg *.gif *.mdb
9da5d4c42fdf7a979ef5f09d33c0a444 *Hydrocarbon%20fuel%20cell%20page2.jpg
5e39dcc44acccdca7bba0c15c6901c43 *PEM-fuel-cell-large.jpg
864e397c2f38ccfb778f348817f98b91 *pem_fuelcell.gif
c3a869ff6b71c7be3eb06b6635c864b1 *CAT.mdb


C:\forensics>dir *.jpg *.gif *.mdb
23/04/2004  17:21           208,127 Hydrocarbon%20fuel%20cell%20page2.jpg
23/04/2004  17:23            28,167 PEM-fuel-cell-large.jpg
23/04/2004  17:15            30,264 pem_fuelcell.gif
23/04/2004  18:21           184,320 CAT.mdb
```

The Remote_Access_Policy.doc document contains an Access database CAT.mdb. The Password_Policy.doc file contains several fuel cell images, presumably the schematics to which Leszczynski refers in his Opportunity.txt.

This information is important to the systems administrators at Ballard. These documents are not marked according to the Ballard security policy, and in any case, we do not know what Leszczynski's access level was. If these documents were not knowingly made available to Leszczynski, then the access control methods used to restrict access to them has failed. Systems and procedure should be reviewed in light of this.

We can perform some time line analysis on the recovered files as well, as *Camouflage* happily preserves[vii] the original MAC times. Using the mac.pl script (Copyright 2000/2001 H. Carvey keydet89@yahoo.com.) we find that:

Filename	Modified	Accessed	Created
pem_fuelcell.gif	Fri Apr 23 16:15:16 2004	Fri Apr 23 20:59:36 2004	Fri Apr 23 16:19:47 2004
Hydrocarbon%20fuel%20cell%20page2.jpg	Fri Apr 23 16:21:02 2004	Fri Apr 23 20:59:36 2004	Fri Apr 23 16:21:26 2004
PEM-fuel-cell-large.jpg	Fri Apr 23 16:23:23 2004	Fri Apr 23 20:59:36 2004	Fri Apr 23 16:23:32 2004
CAT.mdb	Fri Apr 23	Fri Apr 23	Thu Apr 22

[vii] We have not performed extensive testing to determine the reliability of Camouflage when it comes to restoring MAC times.

17:21:06 2004	21:00:14 2004	21:57:35 2004

As Camouflage preserves the MAC times, we would expect these to reflect the last state of the files before they were encrypted with Camouflage. The access date suggests that the two host documents were created on Friday April 23rd at 20:59:00 and 21:00:14. This is at odds with the initial timeline analysis.

Without other supporting evidence, it is impossible to completely explain this discrepancy. One possibility is that the images files and database were retrieved from a server in a different time-zone, or that at least one of the systems involved had an incorrectly set clock.

The creation times are also later than the modification times on these files, suggesting they were copied from which ever system is used to edit these files to another system. This would be typical of a backup or central repository server, where files are copied back after the employee has finished editing them. Ballard system administrators should check systems that fit this role.

Since writing this section of the paper, I found much more information on Camouflage, including several tools to break the encryption.

http://www.guillermito2.net/stegano/camouflage/ - Gueillermito's explanation of the encryption algorithm as well as links to detection / data recovery tools.

String Search

Another important tool available via Autopsy is the string (keyword) search. Using this, we can search both allocated and unallocated space for information that may have been well hidden.

Even though we have uncovered important information via *Camouflage*, it is important to make the examination of the disk thorough. Also, based on the information above, we are better able to build our "dirty word" list. This list of relevant / suspicious keywords forms the basis of our search terms.

The "unallocated strings" are stored with a .dls extension – by way of example, we can see the original index.htm file.

```
icsecl2 output # head -n 17 fl-260404-RJL1.img.dls
<HTML>
<HEAD>
<meta http-equiv=Content-Type content="text/html;  charset=ISO-8859-1">
<TITLE>Ballard</TITLE>
</HEAD>
<BODY bgcolor="#EDEDED">

<center>
<OBJECT classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"

codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=
6,0,0,0"
 WIDTH="800" HEIGHT="600" id="ballard" ALIGN="">
 <PARAM NAME=movie VALUE="ballard.swf"> <PARAM NAME=quality VALUE=high> <PARAM
NAME=bgcolor VALUE=#CCCCCC> <EMBED src="ballard.swf" quality=high bgcolor=#CCCCCC
WIDTH="800" HEIGHT="600" NAME="ballard" ALIGN=""
```

```
 TYPE="application/x-shockwave-flash"
PLUGINSPAGE="http://www.macromedia.com/go/getflashplayer"></EMBED>
</OBJECT>
</center>
</BODY>
</HTML>
```

However, as the amount of data we are dealing with is small (a floppy disk image), we can run strings manually on the unallocated space file. This means that we won't miss anything through vagaries in spelling, or due to artefacts in the way Autopsy performs string searches (e.g. no searching across fragmented files[viii]).

For larger data sets, this manual searching becomes impractical – and we are forced to use the search options from within Autopsy.

Legal Implications

The above evidence demonstrates that Mr Leszczynski used *Camouflage* to hide materials inside public domain policy documents.

None of the documents (either the Word policy documents, or the recovered information) were marked according to the Ballard Information Sensitivity Policy, so we are unable to tell whether these fuel cell schematics and reports are sensitive information. The policy states that all markings are not mandatory.

If this information was not previously in the public domain (which we assume a client database would not be), then according to the policy it should be considered Ballard Industries Confidential. The enforcement clause for this policy states:

> **1.0 Enforcement**
> Any employee found to have violated this policy may be subject to disciplinary action, up to and including termination of employment.

The UK Theft Act (1968) does not regard electronic information as *property*[ix], so while Mr Leszczynski could be charged with theft of the physical media, but not the information stored within. They can be legally considered *trade secrets*[x]

Kaltons LLP publish an excellent overview on the Law of Confidentiality and Trade Secrets [3], which states;

> "Once a confidant has obtained *confidential information which was communicated to him in circumstances that import a duty of confidentiality*, he is under an obligation not to disclose it or use it without the consent of the person who provided that information."

[viii] The Autopsy help files explain this more completely – see http://localhost:9999/help/grep_lim.html on a system running Autopsy.
[ix] http://www.lawcom.gov.uk/348.htm
[x] http://www.intellectual-property.gov.uk/std/resources/other_ip_rights/trade_secrets.htm

Owing to the fiduciary relationship between employee and employer;

> "[The] employee's obligation in relation to information which he receives in the course of his employment arises from the indisputable proposition that an employee owes a duty of confidentiality to his employee"

Given the evidence above, and having established the duty of confidentiality, we can demonstrate there has been a breach of the Law of Confidentiality. (This of course assumes that Ballard Industries can demonstrate that the information is/was *confidential*).

Under this law, the principal remedies are;

> "…an injunction restraining the confidant from disclosing or using the confidential information. Damages, assessed on the basis of equal market value of the confidential information wrongly acquired or disclosed, are also available. In appropriate circumstances, orders to account of profits and delivery up may also be granted."

By disclosing the customer database, Mr Leszczynski also violates the Data Protection Act (1998)[xi], where §55 deals with *Unlawful obtaining etc. of personal data;*

> (1) A person must not knowingly or recklessly, without the consent of the data controller-
> (a) obtain or disclose personal data or the information contained in personal data, or
> (b) procure the disclosure to another person of the information contained in personal data.
>
> (2) Subsection (1) does not apply to a person who shows-
> (a) that the obtaining, disclosing or procuring-
> (i) was necessary for the purpose of preventing or detecting crime, or
> (ii) was required or authorised by or under any enactment, by any rule of law or by the order of a court,
> (b) that he acted in the reasonable belief that he had in law the right to obtain or disclose the data or information or, as the case may be, to procure the disclosure of the information to the other person,
> (c) that he acted in the reasonable belief that he would have had the consent of the data controller if the data controller had known of the obtaining, disclosing or procuring and the circumstances of it, or
> (d) that in the particular circumstances the obtaining, disclosing or procuring was justified as being in the public interest.
>
> (3) A person who contravenes subsection (1) is guilty of an offence.
>
> (4) A person who sells personal data is guilty of an offence if he has obtained the data in contravention of subsection (1).
>
> (5) A person who offers to sell personal data is guilty of an offence if-
> (a) he has obtained the data in contravention of subsection (1), or
> (b) he subsequently obtains the data in contravention of that subsection.
>
> (6) For the purposes of subsection (5), an advertisement indicating that personal data are or may be for sale is an offer to sell the data.
>
> (7) Section 1(2) does not apply for the purposes of this section; and for the purposes of subsections (4) to (6), "personal data" includes information extracted from personal data.
>
> (8) References in this section to personal data do not include references to personal data which by virtue of section 28 are exempt from this section.

Notably we have subsection 6, which refers to "an advertisement indicating that personal data are or may be for sale". We have this in the form of the recovered file `Opportunity.txt` which states;

[xi] http://www.hmso.gov.uk/acts/acts1998/19980029.htm

"I have included a sample of our Client Authorized Table database."

Under subsection 5, this constitutes an offence. As we have also recovered the database, we can demonstrate that the data is personal in nature. §60(2) states:

> (2) A person guilty of an offence under any provision of this Act other than paragraph 12 of Schedule 9 is liable-
> (a) on summary conviction, to a fine not exceeding the statutory maximum, or
> (b) on conviction on indictment, to a fine.

The Maximum fine the Magistrates can impose is £5000 [4] or the amount prescribed by the statute whichever is greater (PCC(S)A[xii] 2003, §78, and MCA[xiii], §32(2))

If Leszczynski was not authorised to access the materials in question, he may be guilty of further offences under the Computer Misuse Act (1990)[xiv]

Conclusion

The evidence collected from the recovered floppy disk proves intent by Leszczynski to commit offences under UK law.

Timeline analysis has demonstrated that over the course of 22nd to the 26th of April 2004, Leszczynski deliberately hid company sensitive information in otherwise innocuous documents with a view to illegally removing them from the company premises.

Additionally, information arising from this investigation indicates the possibility of an internal compromise. This should be investigated by the Ballard systems administrators.

[xii] Powers of the Criminal Courts Act 2003: http://www.hmso.gov.uk/acts/acts2000/20000006.htm
[xiii] Magistrates Court Act 1980
[xiv] Computer Misuse Act: http://www.hmso.gov.uk/acts/acts1990/Ukpga_19900018_en_1.htm

Useful Links

http://www.intellectual-property.gov.uk/ - All about IP, trade secrets, copyrights and trademarks under UK law.

http://www.informationcommissioner.gov.uk/ - Covers Data Protection Act, Freedom of Information, etc.

http://www.hmso.gov.uk/ - Her Majesty's Stationery Office. Full text of UK acts of Parliament, Statutory Instruments, etc.

http://www.lawcom.gov.uk/ - Law Commission for England and Wales

References

As indicated in the text by square parenthesis, e.g. [1].

1. TASK, Brian Carrier, www.sleuthkit.org
2. Autopsy Forensics Browser 2.01, Brian Carrier, http://www.sleuthkit.org/autopsy/
3. Karltons LLP Technology Lawyers, An overview of the Law of Confidentiality & Trade Secrets, http://www.kaltons.co.uk/articles/193.htm
4. Blackstone's Criminal Practice 2003, OUP, Murphy (Ed.) Stockdale (Ed.)

Appendix 1 – File Activity Timeline

```
Sat Feb 03 2001 19:44:16    36864 m.. -/-rwxrwxrwx 0        0        5        /mnt/floppy/CamShell.dll (_AMSHELL.DLL) (deleted)
                            36864 m.. -rwxrwxrwx 0        0        5        <fl-260404-RJL1.img-_AMSHELL.DLL-dead-5>
Thu Apr 22 2004 16:31:06    33423 m.. -/-rwxrwxrwx 0        0        17       /mnt/floppy/Internal_Lab_Security_Policy.doc (INTERN~2.DOC)
                            32256 m.. -/-rwxrwxrwx 0        0        13       /mnt/floppy/Internal_Lab_Security_Policy1.doc (INTERN~1.DOC)
Fri Apr 23 2004 10:53:56      727 m.. -/-rwxrwxrwx 0        0        28       /mnt/floppy/_ndex.htm (deleted)
                              727 m.. -rwxrwxrwx 0        0        28       <fl-260404-RJL1.img-_ndex.htm-dead-28>
Fri Apr 23 2004 11:54:32   215895 m.. -/-rwxrwxrwx 0        0        23       /mnt/floppy/Remote_Access_Policy.doc (REMOTE~1.DOC)
Fri Apr 23 2004 11:55:26   307935 m.. -/-rwxrwxrwx 0        0        20       /mnt/floppy/Password_Policy.doc (PASSWO~1.DOC)
Fri Apr 23 2004 14:10:50    22528 m.. -/-rwxrwxrwx 0        0        27       /mnt/floppy/Acceptable_Encryption_Policy.doc (ACCEPT~1.DOC)
Fri Apr 23 2004 14:11:10    42496 m.. -/-rwxrwxrwx 0        0        9        /mnt/floppy/Information_Sensitivity_Policy.doc (INFORM~1.DOC)
Mon Apr 26 2004 00:00:00      727 .a. -rwxrwxrwx 0        0        28       <fl-260404-RJL1.img-_ndex.htm-dead-28>
                            42496 .a. -/-rwxrwxrwx 0        0        9        /mnt/floppy/Information_Sensitivity_Policy.doc (INFORM~1.DOC)
                            36864 .a. -rwxrwxrwx 0        0        5        <fl-260404-RJL1.img-_AMSHELL.DLL-dead-5>
                            33423 .a. -/-rwxrwxrwx 0        0        17       /mnt/floppy/Internal_Lab_Security_Policy.doc (INTERN~2.DOC)
                              727 .a. -/-rwxrwxrwx 0        0        28       /mnt/floppy/_ndex.htm (deleted)
                           307935 .a. -/-rwxrwxrwx 0        0        20       /mnt/floppy/Password_Policy.doc (PASSWO~1.DOC)
                            22528 .a. -/-rwxrwxrwx 0        0        27       /mnt/floppy/Acceptable_Encryption_Policy.doc (ACCEPT~1.DOC)
                            36864 .a. -/-rwxrwxrwx 0        0        5        /mnt/floppy/CamShell.dll (_AMSHELL.DLL) (deleted)
                            32256 .a. -/-rwxrwxrwx 0        0        13       /mnt/floppy/Internal_Lab_Security_Policy1.doc (INTERN~1.DOC)
                           215895 .a. -/-rwxrwxrwx 0        0        23       /mnt/floppy/Remote_Access_Policy.doc (REMOTE~1.DOC)
Mon Apr 26 2004 09:46:18    36864 ..c -rwxrwxrwx 0        0        5        <fl-260404-RJL1.img-_AMSHELL.DLL-dead-5>
                            36864 ..c -/-rwxrwxrwx 0        0        5        /mnt/floppy/CamShell.dll (_AMSHELL.DLL) (deleted)
Mon Apr 26 2004 09:46:20    42496 ..c -/-rwxrwxrwx 0        0        9        /mnt/floppy/Information_Sensitivity_Policy.doc (INFORM~1.DOC)
Mon Apr 26 2004 09:46:22    32256 ..c -/-rwxrwxrwx 0        0        13       /mnt/floppy/Internal_Lab_Security_Policy1.doc (INTERN~1.DOC)
Mon Apr 26 2004 09:46:24    33423 ..c -/-rwxrwxrwx 0        0        17       /mnt/floppy/Internal_Lab_Security_Policy.doc (INTERN~2.DOC)
Mon Apr 26 2004 09:46:26   307935 ..c -/-rwxrwxrwx 0        0        20       /mnt/floppy/Password_Policy.doc (PASSWO~1.DOC)
Mon Apr 26 2004 09:46:36   215895 ..c -/-rwxrwxrwx 0        0        23       /mnt/floppy/Remote_Access_Policy.doc (REMOTE~1.DOC)
Mon Apr 26 2004 09:46:44    22528 ..c -/-rwxrwxrwx 0        0        27       /mnt/floppy/Acceptable_Encryption_Policy.doc (ACCEPT~1.DOC)
Mon Apr 26 2004 09:47:36      727 ..c -rwxrwxrwx 0        0        28       <fl-260404-RJL1.img-_ndex.htm-dead-28>
                              727 ..c -/-rwxrwxrwx 0        0        28       /mnt/floppy/_ndex.htm (deleted)
```

MD5: F79E26DDAD3C2953DF3FDD4D7AFCDCED output/body-timeline

Part 2 – Option 2: Forensic Tool Validation

NTLast by Foundstone Inc.

Scope

NTLast is used to query the Windows Security event log and return a list of logon events. The test will demonstrate that the tool can be used to list both successful and failed logins, either interactive or remote in nature.

Although this information is already present in the event logs of the systems being checked, this tool is used to prepare easily readable evidence of logon activity. In this sense, it is a log analysis tool. As such, we will take the assumption that the Windows accounting and security event logging facilities are considered "forensically sound", and that the check as to the validity of this tool will be related solely to its interpretation of the log files.

Windows does not employ digital signatures to sign the event logs – so we do not have a way of "proving" that the event logs in question came from the machine we are testing against. The test will assume that the logs returned via the SMB call to a remote system (or direct call to local ntsvcs process) are legitimate.

The tests will be limited to security log information from Windows XP Professional and Server 2003 systems. Although the tool can be used to check IIS logs, this will not be covered in the test.

Tool Description

This test is based on NTLast, v3.00, ©2000 Foundstone Inc. (www.foundstone.com)

http://www.foundstone.com/resources/proddesc/ntlast.htm

The MD5 sum for the NTLast.exe binary is calculated as:
`1128a558328023f6006327570c4d201f`

Foundstone provide NTLast free of charge, as part of their suite of utilities available from http://www.foundstone.com/resources/freetools.htm

NTLast is the Windows equivalent of the UNIX tool **last**, but has additional features. Last reads the `/var/log/wtmp` file to display a list of all users logged in and out of the local system since `wtmp` was created. NTLast performs similar actions, but checks against the Windows Security Event Log.

The Security Event Log is used by Windows to store audit information, as configured in the Local Security Policy. Before Windows Server 2003, the

default was to audit nothing. Audit policy can be configured via the Local Security Policy MMC[xv] snap in, or through the auditpol resource kit utility.

The tool provides a consistent and repeatable means of extracting login activity from the Security Event Log. This removes the onus from the investigator when demonstrating that their log analysis techniques are forensically sound (assuming these tests can demonstrate that the same is true of this tool!). It also presents the evidence is a clearly understandable format – which is important when presenting to non-technical audiences.

When running NTLast.exe, we need to be aware of any timestamps it could alter on the host filesystem. Using InspectEXE[xvi], we can retrieve a list of all the directly referenced DLLs. (This of course assumes InspectEXE to be accurate and complete).

DLL Name	Source	Description
ADVAPI32.dll	Windows	Advanced API services (used to read EventLog)
COMCTL32.dll	Windows	Common (GUI) Controls Library
Comdlg32.dll	Windows	Common Dialogs Library
GDI32.dll	Windows	Graphics Device Interface function library
KERNEL32.dll	Windows	Kernel process (processes, threads, environment)
SHELL32.dll	Windows	Shell API (file access)
USER32.dll	Windows	User interface API
WINSPOOL.DRV	Windows	Print spool handling

Table 2 - Libraries directly referenced by NTLast

Using FileMon[xvii], we can check this listing, and also find those library files which may be indirectly called when running NTLast (e.g. initialising COMCTL32.dll). FileMon displays file activity in real time on a system, which makes it ideal for assessing the impact running a tool would have on the MAC (modified, accessed, created) time stamps of any supporting libraries or other system files.

Forensically, it is crucial to minimise the impact of our actions on the target system. Equally, and changes we do make must be well understood and documented – hence it is vital to identify all the files that could be altered when running this tool.

As FileMon will generate massive amounts of data on a live Windows system, we can filter on “ntlast”, which shows us that the following files are accessed in addition to the above:

```
Ntdll.dll
Unicode.nls
```

[xv] Microsoft Management Console
[xvi] InspectEXE by Silurian Software; http://www.silurian.com/win32/inspect.htm
[xvii] FileMon by Sysinternals; http://www.sysinternals.com/ntw2k/source/filemon.shtml

```
Locale.nls
Sorttbls.nls
Shlwapi.dll
Msvcrt.dll
Rpcrt4.dll
Lpk.dll
Ctype.nls
Sortkey.nls
Uxtheme.dll
Usp10.dll
%systemroot%\WinSxS\*
```

We would expect running NTLast.exe to modify the "accessed" time on these system files if run on a live Windows system.

However, the real forensic power behind NTLast comes in its ability to analyse the event logs recovered from offline disk images. While analysis of live logs is useful (e.g. analysis of production servers than cannot be taken offline), it is hard to produce evidence to a high forensic standard, using any tool.

The Windows security model[xviii] prevents NTLast from using its offline[xix] mode on a live system, because the SecEvent.evt file is locked by the operating system and access is brokered by the system APIs. Accessing the event logs in this way on a live system is likely to impact a large number of files, as this requires access to the WMI interfaces and associated APIs.

After a Windows system disks have been imaged offline, the various event logs (App: Application, Sec: Security and Sys: System) can be recovered from:

```
C:\>dir %systemroot%\system32\config\*.Evt
 Volume in drive C is RGPC
 Volume Serial Number is EC78-C225

 Directory of C:\WINNT\system32\config

01/11/2004  09:59        16,384,000 AppEvent.Evt
01/11/2004  09:59        16,973,824 SecEvent.Evt
15/11/2004  02:05         3,866,624 SysEvent.Evt
               3 File(s)     37,224,448 bytes
               0 Dir(s)     369,506,304 bytes free
```

(Of course, offline, the %systemroot% environment variable will not exist, so the analyst will have to identify the Windows install directory).

Once these have been extracted from the disk image, we can use NTLast to perform file analysis:

```
C:\TOOLS>ntlast -m \\RGPC -file SecEvent.evt
```

[xviii] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/debug/base/event_logging_security.asp

[xix] NTLast can analyse live event logs (online), or using the –file parameter, you can specify a saved .Evt file to analyse (offline).

```
MyLogin           RGPC                IC              Wed Nov 17 09:21:55am 2004
MyLogin           RGPC                IC              Wed Nov 17 09:10:23am 2004
MyLogin           RGPC                IC              Wed Nov 17 08:20:59am 2004
MyLogin           RGPC                IC              Tue Nov 16 06:27:25pm 2004
NETWORK SERVICE                       NT AUTHORITY    Tue Nov 16 06:15:36pm 2004
MyLogin           RGPC                IC              Tue Nov 16 04:25:33pm 2004
MyLogin           RGPC                IC              Tue Nov 16 04:10:33pm 2004
MyLogin           RGPC                IC              Tue Nov 16 01:47:39pm 2004
MyLogin           RGPC                IC              Tue Nov 16 01:25:57pm 2004
MyLogin           RGPC                IC              Tue Nov 16 11:37:56am 2004
```

Under offline analysis, we do not affect system files, and so produce good forensic evidence. SecEvent.Evt is directly retrieved from the image, and so standard practice can be applied to ensure that this file is forensically sound (e.g. MD5 summing).

The tools can also be run from a CD based distribution, although as this is a Win32 binary, many of the standard forensic Linux distributions won't run it!

The tool is precompiled by Foundstone, so we do not have the option of changing the way it references system libraries. In common with a lot of Win32 applications, NTLast has a fairly high impact. The tool does not appear to have been compiled in a minimalist manner, as it references many seemingly irrelevant libraries (such as printing!). This is most likely to be an artefact of the Microsoft Visual Studio environment used to compile it.

Additional Options

NTLast also provides some additional features:

```
NTLast Copyright(c) 2000, Foundstone Inc. All Rights Reserved
v3.00 - http://www.foundstone.com
Command Line Switches
        -s      Last Successful Logons
        -f      Last Failed Logons
        -i      Last Interactive Logons
        -r      Last Remote Logons
        -u [u]  Logons by User - (u) = case sensitive username
        -m [m]  Name of machine to search - (m) = machine name
        -n [n]  Number of Last Logons - (n) = number of records records
        -from [\m]      Last logons from - (\m) = upper-case UNC server name
        -c      Condensed Output - Default
        -v      Verbose Output - shows logon/logoff/duration
        -not [u]        Filter out User - Case sensitive
        -null   Include null sessions - Ignored by default
        -mil    Military Time Output - Default matches event log time
        -file   File name - Saved .evt sec log to open
        -csv    Print output as CSV
        -rt     Raw date/times in CSV output
        -l      Last Successful Logon
        -l:i    Last Interactive Logon
        -l:r    Last Remote Logon
        -iis    IIS 4.0 logons only
        -ad     Include entries after this date/time - Specify military time
        -bd     Include entries before this date/time - Specify military time
                Switches -ad and -bd can be combined to get between date
/time(s)
        - or /  Either switch statement can be used
        -?      Help
*Note - Switch arguments are case sensitive - 'UserX' and 'userx' are different
                                              'HOSTX' and '\HOSTX' are different

*Note - To view IIS 4.0 logons, use IIS switch - off by default
*Note - Blank spaces for user name map to NT Anonymous logons
*Note - File switch usage #1-> ntlast -m \\HOSTX -file c:\log\sec.evt
*Note - File switch usage #2-> ntlast -file \\HOSTX\log\sec.evt
*Note - Time searches use 24hr time - Sept, 20th 7am is 20/9/1999-7:0:0
                                      Sept, 20th 7pm is 20/9/1999-19:0:0
```

```
*Note - Using the /u switch last with no username generates list of NULL logons
*Note - For best usage - Set you console buffer to 1,000 lines or more
*Note - Redirect your output to a file - 'ntlast -v > report.txt'
*Note - Append your output like this  - 'ntlast -v >> report.txt'


See http://www.foundstone.com for updates/fixes
```

Many of these are filtering and output options, such as selecting the log types, timestamps to use, filtering users and outputting to CSV. All of these are useful in producing a concise output, while demonstrating that that the evidential output is easily repeatable.

Without a tool like NTLast, offline analysis of event log files would be (at best) difficult. The files are not in a particularly human friendly format!

```
C:\TOOLS>strings SecEvent.evt

Strings v2.1
Copyright (C) 1999-2003 Mark Russinovich
Systems Internals - www.sysinternals.com

Security
RGPC
Security
MyLogin
(0x0,0x559009)
MyLogin
(0x0,0x559009)
SeIncreaseBasePriorityPrivilege
Security
RGPC
Security
MyLogin
(0x0,0x559009)
MyLogin
(0x0,0x559009)
SeIncreaseBasePriorityPrivilege
Security
RGPC
Security
MyLogin
(0x0,0x559009)
MyLogin
(0x0,0x559009)
SeIncreaseBasePriorityPrivilege
Security
RGPC
NT Local Security Authority / Authentication Service
LsaRegisterLogonProcess()
RGPC$
(0x0,0x3E7)
RGPC$
(0x0,0x3E7)
SeTcbPrivilege
Security
RGPC
```

Comparing the output from `strings` to the output NTLast shown previously gives us a clear indication of the usefulness of this tool.

Test Apparatus

We will install Windows on an HP d530C desktop. For two tests on different versions of Windows, we will use:

- Windows Server 2003 Standard Edition. Original release media, no subsequent patches. (5.2.3790 Build 3790)
- Windows XP Professional, Service Pack 2 (5.1.2600 SP2 Build 2600). No post-SP2 patches.

Systems will be installed with the defaults as chosen by the Windows install process. The Windows system will at no point be connected to a network.

To take an offline image of the drive, we will use a Gentoo Linux based system (running latest ebuilds at time of writing). Once the event log has been extracted from the image, we will transfer to a Windows XP SP2 machine (patched up to date of this report) and run NTLast on the file from here.

The Gentoo system used to perform disk imaging is not connected to a network. Once extracted, the .Evt file is transferred to the Windows XP analysis system via floppy disk. As we have an MD5 sum generated on the off-net linux system of the evidence we will be transferring, we can demonstrate the integrity of this data further on in the analysis process.

Environmental Conditions

The systems used in the test are physically located in an office which remains locked when unattended. As mentioned above, the systems used to collect the data and generate associated MD5 sums are not connected to a network.

The analysis system is connected to a network, and is access restricted so that only the tester can gain access to the system (via known methods). This network is a public network, protected by border firewall from the internet as a whole. Internal attacks are possible, but they would have to bypass the local BlackICE firewall as well.

As tools are retrieved from the internet, we are also reliant on the web site of the tool supplier. We assume that the tool obtained is as the author intended it to be, and has not been replaced with a modified version by a third party.

Description of the Procedures

Configuration of the Windows system

Install Windows via bootable CD, onto 3 Gb NTFS partition. The size of the partition is deliberately limited to allow us to image the disk easily. Windows is installed using the default options presented in the setup wizard.

Additional for Windows XP – setup will automatically log the initial user in, so once this is done, set a password on that account. Pressing Ctrl+Alt+Delete at the graphical login screen will produce the traditional username and password prompt.

Initial login as administrator to:

- Check clock is set accurately
- Configure local Audit Policy to audit success and failure on all events. We will use the auditpol utility to perform this. As auditpol is only present in the Windows resource kit, we will need to install the resource kit on another Windows system (to avoid unnecessarily contaminating the test system) and copy over the auditpol executable via floppy disk.
- Set event logs to maximum size of 64Mb and to overwrite events as needed. This is done via the Event Viewer application (eventvwr). Right clicking the event log and choosing properties allows us to set these logs up as required.

We can use `auditpol` from the Windows Resource Kit to show that auditing has been enabled, and that we are recording success and failure events for everything the Security Event log can record.

```
C:\>auditpol
Running ...

(X) Audit Enabled

System                    = Success and Failure
Logon                     = Success and Failure
Object Access             = Success and Failure
Privilege Use             = Success and Failure
Process Tracking          = Success and Failure
Policy Change             = Success and Failure
Account Management        = Success and Failure
Directory Service Access  = Success and Failure
Account Logon             = Success and Failure
```

Once the system is configured as above, we will perform the following tests to place events in the Security log:

- Two login failures by misspelling Administrator – to make this obvious in the results, we will use “admin”
- Two login failures for Administrator by giving the wrong password
- Successful login for Administrator
- Non-interactive login test using command line tool “runas” to launch a notepad process: `runas /user:sans\administrator "notepad.exe"`

At the start of the tests, we will record the time. Using a stopwatch, each the time offset of each subsequent test will be recorded. This will allow us to build a time line of events to which we can compare the output from NTLast.

Once these tests have been completed, the system will be shut down. Simply powering off a Windows system can result in a corrupt event log. Due to protection built into the event log service, only the original system can open and repair them. This is an added complication in performing offline analysis of a Windows system in an unknown state. However, as the scope of the test is NTLast’s ability to analyse event log files, rather than the Windows event logging service, it will simply be noted that NTLast cannot analyse corrupt event log files. Once the system has been shut down, the hard disk will be removed and connected to the Linux system.

Using dd, we will image the entire disk, as per standard forensic methodology. Using mmls and dd again, the logical image can be extracted from this and mounted as a read only loop device.

From here, we can extract the SecEvent.Evt file and transfer via SCP to the Windows analysis system. Using MD5 sums, we can demonstrate that the file recovered from inside the image is not altered by NTLast and show a chain of custody for this particular piece of evidence.

We can then use NTLast to compare the event log files to our known timeline, to ensure that events are accurately interpreted by NTLast.

To ensure that the results are stored clearly, we will use the following notation to construct the file names for recovered event logs:

```
SecEvent_[operating system].Evt
```

Where *operating system* will be either *winxp (Windows XP Professional)* or *win2k3 (Windows Server 2003)*. To record the MD5 sums, ".MD5" will be appended to the filename, and the MD5 sum stored here, using md5sum as per:

```
md5sum SecEvent_[operating system].Evt > SecEvent_[operating system].Evt.MD5
```

As per best practice, we will take MD5 sums of each file collected.

To test each of the above events that we recorded in the event log, we will use NTLast's various options to test for different logon types. We shall use the –v flag to set verbose outputs for each of the test types, producing results thus:

Flags	Output Filename Suffix
-s	Success
-s –v	Success.verbose
-f	Fail
-f –v	Fail.verbose
-i	Interact
-i –v	Interact.verbose
-r	Remote
-r -v	Remote.verbose

Table 3 - Tests to be run

Because we attempted no remote logins, we expect to find no results for the last two tests. Not all the logins will be considered "Interactive" by the Window event logging service – namely those events generated using the runas command. NTLast should be able to distinguish these events.

As we are testing NTLast, rather than the event log service itself (see **Scope**) these tests are repeatable, assuming we can show a chain of custody for the event log retrieved from the target system.

The tests will be run using the following batch scripts, to ensure they are easily repeatable:

```
@echo off
echo Testing Event File: %EVTFILE%

.\md5sum %EVTFILE% > %EVTFILE%.Evt.MD5

.\NTLast -s -file %EVTFILE% > %EVTFILE%.Success
.\md5sum %EVTFILE%.Success > %EVTFILE%.Success.md5

.\NTLast -s -v -file %EVTFILE% > %EVTFILE%.Success.Verbose
.\md5sum %EVTFILE%.Success.Verbose > %EVTFILE%.Success.Verbose.md5

.\NTLast -f -file %EVTFILE% > %EVTFILE%.Fail
.\md5sum %EVTFILE%.Fail > %EVTFILE%.Fail.md5

.\NTLast -f -v -file %EVTFILE% > %EVTFILE%.Fail.Verbose
.\md5sum %EVTFILE%.Fail.Verbose > %EVTFILE%.Fail.Verbose.md5

.\NTLast -i -file %EVTFILE% > %EVTFILE%.Interact
.\md5sum %EVTFILE%.Interact > %EVTFILE%.Interact.md5

.\NTLast -i -v -file %EVTFILE% > %EVTFILE%.Interact.Verbose
.\md5sum %EVTFILE%.Interact.Verbose > %EVTFILE%.Interact.Verbose.md5

.\NTLast -r -file %EVTFILE% > %EVTFILE%.Remote
.\md5sum %EVTFILE%.Remote > %EVTFILE%.Remote.md5

.\NTLast -r -v -file %EVTFILE% > %EVTFILE%.Remote.Verbose
.\md5sum %EVTFILE%.Remote.Verbose > %EVTFILE%.Remote.Verbose.md5
```

Note that the batch file requires that NTLast and md5sum be in the current path. This ensures we are running with a “known” copy, rather than one retrieved at random from the PATH environment variable. To perform the tests we need, we will use the following batch file to call our test script:

```
@echo off
echo GCFA Part 2 - Richard Grime
echo Tool Analysis :: NTLast (C) Foundstone
echo.

set EVTFILE=SecEvent_win2k3.Evt
.\md5sum %EVTFILE% > %EVTFILE%.md5.before
call .\TestEvtFile.bat
.\md5sum %EVTFILE% > %EVTFILE%.md5.after

set EVTFILE=SecEvent_winxp.Evt
.\md5sum %EVTFILE% > %EVTFILE%.md5.before
call .\TestEvtFile.bat
.\md5sum %EVTFILE% > %EVTFILE%.md5.after
```

This batch file takes MD5 sums of the evidence files both before and after NTLast has been run against it – demonstrating that NTLast has not modified the evidence.

Criteria for Approval

The log information presented by NTLast should be a match for the events that we simulated on the system. Under these direct criteria, we are assuming the event log service to be accurate**.** Strictly speaking, we should compare NTLast’s results with a manual interpretation of the binary .evt file.

However, if NTLast's results are accurate (i.e. it correctly reports the simulated events), then this will both uphold the assumption and demonstrate that NTLast displays the "correct" results.

We will seek to prove that NTLast can correctly retrieve and display, by login type, events from Windows XP and Windows Server 2003 event log files.

NTLast will be executed with the flags given in table 2, using the additional flag "-file" to specify the Windows XP and Windows 2003 event logs in turn. The resultant output should match the events as simulated on the system. The tool does not manipulate the evidence files as it is being run – we can use MD5 sums to demonstrate this.

While NTLast does call system files, we are running this test offline on an analysis system, not on the original system. However, for completeness, these files are referenced in table 1. As mentioned above, to ensure repeatability, we will use batch scripts to execute the tool.

The output files should contain an accurate time-stamped list of login events, according to the type specified by the flags passed at execution time. The verbose output should give us full information about these events (as you would expect to see in the Windows event viewer application). As such, we can produce evidence that is easily human readable (non-verbose), and detailed (verbose) for closer examination.

Results

For the full result set, please see Appendices 1a and 1b below.

	Operating System		Notes
Test	**Windows XP Pro**	**Windows Server 2003**	
Two failed logins under "admin"	Pass	Pass	
Two failed logins under "Administrator"	Pass	Pass	
Successful Administrator login "runas" Administrator	Pass	Pass	1
Pass/Fail	Pass	Pass	

Notes:

[1] We can tell this was successful via timestamps. Runas is still considered "interactive" by the Windows event logging service.

The results demonstrate that the tool is successful in interpreting events recorded on a system that has no network access.

Collected MD5 sums also show that the evidence has not been modified by these operations:

```
C:\part2>type SecEvent_win2k3.evt.md5.before
\635735a1da86971e12adf8d317465b0c *C:\\part2\\SecEvent_win2k3.Evt
C:\part2>type SecEvent_win2k3.evt.md5.after
\635735a1da86971e12adf8d317465b0c *C:\\part2\\SecEvent_win2k3.Evt

C:\part2>type SecEvent_winxp.evt.md5.before
\0efb287645d6ae215b83d4bf873465dd *C:\\part2\\SecEvent_winxp.Evt
C:\part2>type SecEvent_winxp.evt.md5.after
\0efb287645d6ae215b83d4bf873465dd *C:\\part2\\SecEvent_winxp.Evt
```

Analysis

NTLast can be used by a forensic investigator to easily discover logged activity on the system, and even infer whether log files have been tampered with. In short, it provides a method of making otherwise indecipherable binary logs "human readable". In this way, the standard log processing techniques used by investigators (looking for log on events, chunks of time missing from the logs – indicating possible erased events, etc.) can be utilised.

Use of the tool still requires an understanding of what log events will be recorded by Windows – the output is only as good as the input. For the investigator, this means knowing the audit policy on the system. Many compromised systems we have looked at have had the auditing disabled by the attacker. To determine what *should* be being logged, the investigator will need access to the HKEY_Local_Machine registry hive from the target system[xx].

The investigator should bear in mind that the format of the event logs prevents a user legitimately clearing selective parts of the event log. The same checksums that prevent us from analysing a log that has been recovered from a system that was not shut down cleanly protect the log file. As such – a user with sufficient privilege can clear the entire event log. This activity is recorded in the log file[xxi] but is not interpreted by NTLast (in its capacity as a "last logon" tool).

The other option open to attackers who cannot leverage this privilege is to fill the event log – pushing their activity off the end. This is a particular problem for systems which use default log sizes of 512kb and are set to "overwrite as necessary".

Finally, sophisticated attackers may be able to modify the event log, and get the local system to repair it. In theory (at least) this should not be possible. It is however feasible to deliberately corrupt the log file so that the Event service cannot open it – forcing it to create a new one.

This tool does however provide login data which can be very valuable for the investigator, especially where attacks have been unsuccessful (hence the

[xx] How to Determine Audit Policies from the Registry:
http://support.microsoft.com/default.aspx?scid=kb;en-us;246120&sd=tech
[xxi] Appendix B: Audit Categories and Events:
http://www.microsoft.com/technet/Security/topics/issues/w2kccscg/w2kscgcb.mspx

ability to manipulate log information was not available). It also provides the ability to analyse live systems in environments where systems cannot be taken off line.

Presentation

The tool delivers results that are readily understandable by people with a basic understanding of system administration – however, this may not be entirely appropriate for a lay audience (such as court).

Although not shown in these results, the tool can output in a CSV (comma separated values) format. This means that the data can be easily imported into others tool which can produce a more visual output.

The investigator may also consider products such as Microsoft Visio, which can be used to draw a timeline of events[xxii].

Any "representations" of the data produced should be backed up by both the original log file and the verbose output from the NTLast tool, so that other parties can examine the data in more detail should they desire.

Conclusion

This test attempted to demonstrate that NTLast could be used to successfully interpret log events from a Windows security event log, do so in a forensically sound manner, and display the results in a usable format.

To that end, the test has been successful. All the features of the tool are not covered by this test, and some of its most useful features are not as "forensically sound". This caveat does however apply to all tools analysing data on live systems.

Given that the security event logs are effectively un-presentable as evidence in their binary form, NTLast is a valuable tool for demonstrating log activity to a lay person. As an analysis tool (and as the tests have shown) it does not modify the evidence gathered from the target system.

It would be useful if NTLast were capable of reading event logs from systems that had not been shutdown cleanly. Forensically, it makes more sense to power systems off in this manner – so if NTLast could warn that the log was corrupt, but still read out the remaining events, this would be useful. Without enforcing the checks on the event log, an attacker could leverage this feature to potentially replace information in the event log and bypass Windows Event Logging security. However, for the majority of less sophisticated attacks, it would be useful method of retrieving information from an otherwise unreadable event log. Provided adequate warning was given as to the state of the event log, this would be a valuable addition.

Removing references to un-needed libraries would also reduce the forensic footprint of NTLast. Some of the libraries referenced are heavy duty (and

[xxii] http://office.microsoft.com/en-gb/assistance/CH010423821033.aspx

seemingly unnecessary for a console application) and calling them in turn invokes a whole spate of file access.

Appendix 1a: Result Files – Windows XP Pro

SecEvent_winxp.Evt.Fail

```
Administrator   SANS                SANS            Mon Nov 22 11:15:36pm 2004
Administrator   SANS                SANS            Mon Nov 22 11:15:35pm 2004
admin           SANS                SANS            Mon Nov 22 11:15:32pm 2004
admin           SANS                SANS            Mon Nov 22 11:15:30pm 2004
```

SecEvent_winxp.Evt.Fail.Verbose

```
Record Number: 65
ComputerName: SANS
EventID: 529     - Failed Logon Attempt
Time Attempted:  - Mon Nov 22 11:15:36pm 2004
Details -
       ClientName:    Administrator
       ClientMachine: SANS
       ClientDomain:  SANS
       LogonType:     Interactive

Record Number: 63
ComputerName: SANS
EventID: 529     - Failed Logon Attempt
Time Attempted:  - Mon Nov 22 11:15:35pm 2004
Details -
       ClientName:    Administrator
       ClientMachine: SANS
       ClientDomain:  SANS
       LogonType:     Interactive

Record Number: 61
ComputerName: SANS
EventID: 529     - Failed Logon Attempt
Time Attempted:  - Mon Nov 22 11:15:32pm 2004
Details -
       ClientName:    admin
       ClientMachine: SANS
       ClientDomain:  SANS
       LogonType:     Interactive

Record Number: 59
ComputerName: SANS
EventID: 529     - Failed Logon Attempt
Time Attempted:  - Mon Nov 22 11:15:30pm 2004
Details -
       ClientName:    admin
       ClientMachine: SANS
       ClientDomain:  SANS
       LogonType:     Interactive

Record Number: 56
```

SecEvent_winxp.Evt.Interact

```
Administrator   SANS                SANS            Mon Nov 22 11:16:18pm 2004
Administrator   SANS                SANS            Mon Nov 22 11:15:39pm 2004
```

SecEvent_winxp.Evt.Interact.Verbose

```
Record Number: 122
ComputerName: SANS
EventID: 528   -   Successful Logon
Logon:  Mon Nov 22 11:16:18pm 2004
Logoff: Not Recorded
Details -
       ClientName:    Administrator
       ClientID:      (0x0,0x74CE8)
       ClientMachine: SANS
       ClientDomain:  SANS
       LogonType:     Interactive

Record Number: 67
ComputerName: SANS
EventID: 528   -   Successful Logon
Logon:  Mon Nov 22 11:15:39pm 2004
Logoff: Not Recorded
Details -
       ClientName:    Administrator
       ClientID:      (0x0,0x59741)
       ClientMachine: SANS
```

```
        ClientDomain:  SANS
        LogonType:     Interactive

SecEvent_winxp.Evt.Remote
 - No Records - Check to see if auditing is on

SecEvent_winxp.Evt.Remote.Verbose
 - No Records - Check to see if auditing is on

SecEvent_winxp.Evt.Success
Administrator   SANS                SANS            Mon Nov 22 11:16:18pm 2004
Administrator   SANS                SANS            Mon Nov 22 11:15:39pm 2004
NETWORK SERVICE                     NT AUTHORITY    Mon Nov 22 11:14:09pm 2004
NETWORK SERVICE                     NT AUTHORITY    Mon Nov 22 11:14:09pm 2004
NETWORK SERVICE                     NT AUTHORITY    Mon Nov 22 11:14:09pm 2004

SecEvent_winxp.Evt.Success.Verbose
Record Number: 122
ComputerName: SANS
EventID: 528   -   Successful Logon
Logon:  Mon Nov 22 11:16:18pm 2004
Logoff: Not Recorded
Details -
        ClientName:    Administrator
        ClientID:      (0x0,0x74CE8)
        ClientMachine: SANS
        ClientDomain:  SANS
        LogonType:     Interactive

Record Number: 67
ComputerName: SANS
EventID: 528   -   Successful Logon
Logon:  Mon Nov 22 11:15:39pm 2004
Logoff: Not Recorded
Details -
        ClientName:    Administrator
        ClientID:      (0x0,0x59741)
        ClientMachine: SANS
        ClientDomain:  SANS
        LogonType:     Interactive

Record Number: 43
ComputerName: SANS
EventID: 528   -   Successful Logon
Logon:  Mon Nov 22 11:14:09pm 2004
Logoff: Not Recorded
Details -
        ClientName:    NETWORK SERVICE
        ClientID:      (0x0,0x3E4)
        ClientMachine:
        ClientDomain:  NT AUTHORITY
        LogonType:

Record Number: 41
ComputerName: SANS
EventID: 528   -   Successful Logon
Logon:  Mon Nov 22 11:14:09pm 2004
Logoff: Not Recorded
Details -
        ClientName:    NETWORK SERVICE
        ClientID:      (0x0,0x3E4)
        ClientMachine:
        ClientDomain:  NT AUTHORITY
        LogonType:

Record Number: 37
ComputerName: SANS
EventID: 528   -   Successful Logon
Logon:  Mon Nov 22 11:14:09pm 2004
Logoff: Not Recorded
Details -
        ClientName:    NETWORK SERVICE
        ClientID:      (0x0,0x3E4)
        ClientMachine:
        ClientDomain:  NT AUTHORITY
        LogonType:
```

Appendix 1b: Result Files – Windows Server 2003

SecEvent_win2k3.Evt.Fail

```
Administrator   SANS-8JNEZNPDK2      SANS-8JNEZNPDK2 Thu Nov 18 06:02:00pm 2004
Administrator   SANS-8JNEZNPDK2      SANS-8JNEZNPDK2 Thu Nov 18 06:01:58pm 2004
admin           SANS-8JNEZNPDK2      SANS-8JNEZNPDK2 Thu Nov 18 06:01:50pm 2004
admin           SANS-8JNEZNPDK2      SANS-8JNEZNPDK2 Thu Nov 18 06:01:47pm 2004
```

SecEvent_win2k3.Evt.Fail.Verbose

```
Record Number: 121
ComputerName: SANS-8JNEZNPDK2
EventID: 529     - Failed Logon Attempt
Time Attempted:  - Thu Nov 18 06:02:00pm 2004
Details -
       ClientName:    Administrator
       ClientMachine: SANS-8JNEZNPDK2
       ClientDomain:  SANS-8JNEZNPDK2
       LogonType:     Interactive

Record Number: 119
ComputerName: SANS-8JNEZNPDK2
EventID: 529     - Failed Logon Attempt
Time Attempted:  - Thu Nov 18 06:01:58pm 2004
Details -
       ClientName:    Administrator
       ClientMachine: SANS-8JNEZNPDK2
       ClientDomain:  SANS-8JNEZNPDK2
       LogonType:     Interactive

Record Number: 117
ComputerName: SANS-8JNEZNPDK2
EventID: 529     - Failed Logon Attempt
Time Attempted:  - Thu Nov 18 06:01:50pm 2004
Details -
       ClientName:    admin
       ClientMachine: SANS-8JNEZNPDK2
       ClientDomain:  SANS-8JNEZNPDK2
       LogonType:     Interactive

Record Number: 115
ComputerName: SANS-8JNEZNPDK2
EventID: 529     - Failed Logon Attempt
Time Attempted:  - Thu Nov 18 06:01:47pm 2004
Details -
       ClientName:    admin
       ClientMachine: SANS-8JNEZNPDK2
       ClientDomain:  SANS-8JNEZNPDK2
       LogonType:     Interactive
```

SecEvent_win2k3.Evt.Interact

```
Administrator   SANS-8JNEZNPDK2      SANS-8JNEZNPDK2 Thu Nov 18 06:03:24pm 2004
Administrator   SANS-8JNEZNPDK2      SANS-8JNEZNPDK2 Thu Nov 18 06:02:03pm 2004
```

SecEvent_win2k3.Evt.Interact.Verbose

```
Record Number: 143
ComputerName: SANS-8JNEZNPDK2
EventID: 528   -   Successful Logon
Logon:  Thu Nov 18 06:03:24pm 2004
Logoff: Not Recorded
Details -
       ClientName:    Administrator
       ClientID:      (0x0,0x43A90)
       ClientMachine: SANS-8JNEZNPDK2
       ClientDomain:  SANS-8JNEZNPDK2
       LogonType:     Interactive

Record Number: 124
ComputerName: SANS-8JNEZNPDK2
EventID: 528   -   Successful Logon
Logon:  Thu Nov 18 06:02:03pm 2004
Logoff: Not Recorded
Details -
       ClientName:    Administrator
       ClientID:      (0x0,0x3D707)
```

```
        ClientMachine: SANS-8JNEZNPDK2
        ClientDomain:  SANS-8JNEZNPDK2
        LogonType:     Interactive


SecEvent_win2k3.Evt.Remote
 - No Records - Check to see if auditing is on

SecEvent_win2k3.Evt.Remote.Verbose
 - No Records - Check to see if auditing is on

SecEvent_win2k3.Evt.Success
Administrator   SANS-8JNEZNPDK2      SANS-8JNEZNPDK2 Thu Nov 18 06:03:24pm 2004
Administrator   SANS-8JNEZNPDK2      SANS-8JNEZNPDK2 Thu Nov 18 06:02:03pm 2004
NETWORK SERVICE                      NT AUTHORITY    Thu Nov 18 04:42:00pm 2004

SecEvent_win2k3.Evt.Success.Verbose
Record Number: 143
ComputerName: SANS-8JNEZNPDK2
EventID: 528   -   Successful Logon
Logon:  Thu Nov 18 06:03:24pm 2004
Logoff: Not Recorded
Details -
        ClientName:    Administrator
        ClientID:      (0x0,0x43A90)
        ClientMachine: SANS-8JNEZNPDK2
        ClientDomain:  SANS-8JNEZNPDK2
        LogonType:     Interactive

Record Number: 124
ComputerName: SANS-8JNEZNPDK2
EventID: 528   -   Successful Logon
Logon:  Thu Nov 18 06:02:03pm 2004
Logoff: Not Recorded
Details -
        ClientName:    Administrator
        ClientID:      (0x0,0x3D707)
        ClientMachine: SANS-8JNEZNPDK2
        ClientDomain:  SANS-8JNEZNPDK2
        LogonType:     Interactive

Record Number: 105
ComputerName: SANS-8JNEZNPDK2
EventID: 528   -   Successful Logon
Logon:  Thu Nov 18 06:01:15pm 2004
Logoff: Not Recorded
Details -
        ClientName:    Administrator
        ClientID:      (0x0,0x3CA72)
        ClientMachine: SANS-8JNEZNPDK2
        ClientDomain:  SANS-8JNEZNPDK2
        LogonType:

Record Number: 96
ComputerName: SANS-8JNEZNPDK2
EventID: 528   -   Successful Logon
Logon:  Thu Nov 18 04:42:00pm 2004
Logoff: Not Recorded
Details -
        ClientName:    NETWORK SERVICE
        ClientID:      (0x0,0x3E4)
        ClientMachine:
        ClientDomain:  NT AUTHORITY
        LogonType:
```

Appendix 2: Resultant MD5 sums

```
\635735a1da86971e12adf8d317465b0c *C:\\part2\\SecEvent_win2k3.Evt
\d0e56da6b5f2cce5f3d594d13b851abf *C:\\part2\\SecEvent_win2k3.Evt.Fail
\c987c423c752ad9410b1a201244744cb *C:\\part2\\SecEvent_win2k3.Evt.Fail.Verbose
\34212b955437d79c45d802645ffdd29e *C:\\part2\\SecEvent_win2k3.Evt.Interact
\8d350c175d34562bbf8cc39ee8642729 *C:\\part2\\SecEvent_win2k3.Evt.Interact.Verbose
\2428169b31c96fe4a767215000aa0008 *C:\\part2\\SecEvent_win2k3.Evt.Remote
\2428169b31c96fe4a767215000aa0008 *C:\\part2\\SecEvent_win2k3.Evt.Remote.Verbose
\d6bd182f298d66cf79f0fd0c92b48ad4 *C:\\part2\\SecEvent_win2k3.Evt.Success
\75debd7c275530ce63fb447ccf62f24d *C:\\part2\\SecEvent_win2k3.Evt.Success.Verbose
\0efb287645d6ae215b83d4bf873465dd *C:\\part2\\SecEvent_winxp.Evt
\8577ac047ffb0196c9a131ec2154205b *C:\\part2\\SecEvent_winxp.Evt.Fail
\115b5c6ed1ee55c9b389942acf3533ca *C:\\part2\\SecEvent_winxp.Evt.Fail.Verbose
\57ba36038b89aba4a0f257973b0054fa *C:\\part2\\SecEvent_winxp.Evt.Interact
\d6f0cce7c86d0a694c1733af9512f576 *C:\\part2\\SecEvent_winxp.Evt.Interact.Verbose
\2428169b31c96fe4a767215000aa0008 *C:\\part2\\SecEvent_winxp.Evt.Remote
\2428169b31c96fe4a767215000aa0008 *C:\\part2\\SecEvent_winxp.Evt.Remote.Verbose
\64c4ca69a5a1517bd86afd7f031daaf8 *C:\\part2\\SecEvent_winxp.Evt.Success
\708b197f3795bbf3f5ff9ed4e701e5ef *C:\\part2\\SecEvent_winxp.Evt.Success.Verbose
```

Miscellaneous Files

```
\5f169128f69b0104834cddf8a82bc4c1 *C:\\part2\\runtest.bat
\95916ae5a3937716657444e5116bd02c *C:\\part2\\testevtfile.bat
\1128a558328023f6006327570c4d201f *C:\\part2\\ntlast.exe
\16680bef0c81ef117a638154d82fed02 *C:\\part2\\getopt.dll
\4e846747a60ae629aa7f211322361e12 *C:\\part2\\md5lib.dll
\01ccdb12282dd542182fef660cc2a574 *C:\\part2\\md5sum.exe
```

Last Updated: May 28th, 2017

Upcoming Training

SANS Madrid 2017	Madrid, Spain	May 29, 2017 - Jun 03, 2017	Live Event
SANS Rocky Mountain 2017	Denver, CO	Jun 12, 2017 - Jun 17, 2017	Live Event
SANS Secure Europe 2017	Amsterdam, Netherlands	Jun 12, 2017 - Jun 20, 2017	Live Event
DFIR Summit & Training 2017	Austin, TX	Jun 22, 2017 - Jun 29, 2017	Live Event
SANS Paris 2017	Paris, France	Jun 26, 2017 - Jul 01, 2017	Live Event
SANS London July 2017	London, United Kingdom	Jul 03, 2017 - Jul 08, 2017	Live Event
SANS Cyber Defence Singapore 2017	Singapore, Singapore	Jul 10, 2017 - Jul 15, 2017	Live Event
SANS Los Angeles - Long Beach 2017	Long Beach, CA	Jul 10, 2017 - Jul 15, 2017	Live Event
SANSFIRE 2017	Washington, DC	Jul 22, 2017 - Jul 29, 2017	Live Event
Community SANS Columbia FOR508	Columbia, MD	Aug 21, 2017 - Aug 26, 2017	Community SANS
SANS Virginia Beach 2017	Virginia Beach, VA	Aug 21, 2017 - Sep 01, 2017	Live Event
Network Security 2017 - FOR508: Advanced Digital Forensics, Incident Response, and Threat Hunting	Las Vegas, NV	Sep 10, 2017 - Sep 15, 2017	vLive
SANS Network Security 2017	Las Vegas, NV	Sep 10, 2017 - Sep 17, 2017	Live Event
SANS Dublin 2017	Dublin, Ireland	Sep 11, 2017 - Sep 16, 2017	Live Event
Data Breach Summit & Training	Chicago, IL	Sep 25, 2017 - Oct 02, 2017	Live Event
SANS DFIR Prague 2017	Prague, Czech Republic	Oct 02, 2017 - Oct 08, 2017	Live Event
SANS October Singapore 2017	Singapore, Singapore	Oct 09, 2017 - Oct 28, 2017	Live Event
SANS Tysons Corner Fall 2017	McLean, VA	Oct 14, 2017 - Oct 21, 2017	Live Event
SANS Tokyo Autumn 2017	Tokyo, Japan	Oct 16, 2017 - Oct 28, 2017	Live Event
SANS vLive - FOR508: Advanced Digital Forensics, Incident Response, and Threat Hunting	FOR508 - 201710,	Oct 16, 2017 - Nov 22, 2017	vLive
SANS Seattle 2017	Seattle, WA	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS Gulf Region 2017	Dubai, United Arab Emirates	Nov 04, 2017 - Nov 16, 2017	Live Event
SANS Sydney 2017	Sydney, Australia	Nov 13, 2017 - Nov 25, 2017	Live Event
SANS Austin Winter 2017	Austin, TX	Dec 04, 2017 - Dec 09, 2017	Live Event
SANS Cyber Defense Initiative 2017	Washington, DC	Dec 12, 2017 - Dec 19, 2017	Live Event
SANS OnDemand	Online	Anytime	Self Paced
SANS SelfStudy	Books & MP3s Only	Anytime	Self Paced