GIAC
CERTIFICATIONS

Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permited without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Advanced Incident Response, Threat Hunting, and Digital Forensics (Forensics
at http://www.giac.org/registration/gcfa

Analysis of WinHex

GIAC Certified Forensic Analyst (GCFA)

Practical Assignment

Version 1.5

Option Two

Jessica Dillinger
SansFire '04 Monterey, CA

© SANS Institute 2005 Author retains full rights.

Table of Contents

© SANS Institute 2005, Author retains full rights.

© SANS Institute 2005 Author retains full rights.

List of Figures and Tables

© SANS Institute 2005, Author retains full rights.

© SANS Institute 2005 Author retains full rights.

Abstract

This practical assignment was created to meet the criteria for the GIAC Certified Forensic Analyst certification, which consists of two parts. The first part of the certification requirement is to analyze a floppy disk that contains several files. This will be done using several forensic utilities including Autopsy, shred and WinHex. This portion of the paper will include how the computer that was used for the investigation was prepared. It will also explain the steps taken to examine the media and what was found during each step of the investigation. The tool that was used in this case will be studied in order to give a accurate description of how it works and what it was used for. A brief conclusion will give an overview of the evidence found and conclude what the suspect was trying to accomplish. Finally, the legal implications will be discussed along with the punishments the suspect could face based first on business policies and then on state and federal laws.

For the second portion of the paper the second option was chosen. Analyzing a tool that could be used for forensics will be investigated to determine how well it works and what typed of forensic capabilities it has. The tool that will be reviewed is WinHex. WinHex is best known for its disk editing capability. However, it has many other capabilities, including RAM editor, erasing entire hard drives, drive cloning and random number generator.

© SANS Institute 2005, Author retains full rights.

© SANS Institute 2005 Author retains full rights.

Introduction

A brief analysis of the floppy disk will be discussed in this section of the paper. Robert Leszczynski has taken the floppy disk out of Ballard Industry against corporate policy, on April 26th 2004 at approximately 4:45pm MST. A security guard had seized the floppy disk from Mr. Leszczynski and given it to the security administrator. A chain of custody form as been provided by David Keen, the security administrator, the information included on the form consists of:

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

David Keen has asked for a report including what is on the floppy disk and how Robert Leszczynski may have used it.

Preparation Steps for Analysis

The analysis was performed on one machine containing two operating systems. The computer is an AMD Athlon XP1900 with 768 megabytes of RAM. The first step in preparing the hard drive, which the evidence would be tested on, was to wipe the drive. This insures that the drive will not contain any data from previous operating systems. Knoppix, which is a Free and Open source bootable CD used for forensics because it allows the investigator to view information on the drive without touching or altering the data. Knoppix has a distribution of Linux installed. Knoppix is used for data recover and includes several open source programs included in other Linux distributions. One of which is the program shred. Shred is a wiping utility that was used to cleanse the hard drive of any data that may have interfered with the examination process. This allows the investigator to be certain that the data viewed is not from a previous investigation. It is important to understand how shred works to ensure that the shred utility does what it claims. The process that shred uses to wipe a file will be investigated and determined. The default number of passes that shred performs is 25. The command used to begin the process was:

```
#shred –fvxz hda1
```

A description of each argument is as follows:

- `-f` change permissions to allow writing if necessary
- `-v` show progress
- `-x` do not round file sizes up to the next full block; this is the default for non-regular files
- `-z` add a final overwrite with zeros to hide shredding

© SANS Institute 2005 Author retains full rights.

`hda1` The specified drive/file/partition to be shredded

© SANS Institute 2005, Author retains full rights.

© SANS Institute 2005 Author retains full rights.

The shredding process included a default 25 passes. The first, thirteenth and twenty-fifth pass included wiping the drive with random characters. The rest of the process included one pass with each of the following eight bit Hex representation of a series of ones and zeros. Based on the experiments to establish how shred works the pass is randomly selected from the table below. However, it was determined that each of the passes shown in table 1 was used to wipe the hard drive.

Hexadecimal	Binary
aaa...	101010101010...
bbb...	101110111011...
ccc...	110011001100...
ddd...	110111011101...
eee...	111011101110...
fff...	111111111111...
111...	000100010001...
222...	001000100010...
333...	001100110011...
444...	010001000100...
555...	010101010101...
666...	011001100110...
777...	011101110111...
888...	100010001000...
999...	100110011001...
000...	000000000000...
924...	100100100100...
492...	010010010010...
249...	001001001001...
B69...	101101101001...
6db...	011011011011...
db6...	110110110110...

Table 1 - Conversion table from hexadecimal to binary

Once the drive was verified as clean an operating system was installed on the drive. The host operating system install was Fedora Core 1, which is a Linux distribution. Since the integrity of the CDs that contain the operating system had been verified in the past, an MD5 checksum was not performed on the fresh install of Fedora Core 1. After the operating system was installed correctly, Autopsy version 2.03[1] a forensic browser was installed. Autopsy was installed along with the backbone of the forensic investigation, Sleuthkit version 1.72[2]. Sleuthkit is a compilation of several different command line tools usually used in the UNIX environment. As stated in the Sans Institute handout titled "Forensic

[1] http://www.sleuthkit.org/autopsy/index.php

[2] http://www.sleuthkit.org/sleuthkit/index.php

© SANS Institute 2005 Author retains full rights.

Methodology Illustrated using Linux, Parts 1 and 2 states “autopsy is just an interface to forensic tools”.

The next step to prepare the machine for forensic investigation was to install VMware version 4.5.2[3]. Inside of the VMware workstation Microsoft 2000 was installed. This operating system will be set up to maintain documentation and will be used to host any Microsoft based tools that may be used for the investigation. VMWare was used because it allows the user to set a snapshot of the state of the machine. A snapshot is where the entire state of the computer is frozen and saved. This includes all files, the state of the network and the state of the operating system. This was it allows the user to revert to this state. VMWare is essential to the forensic process. VMWare could be reverted to a saved, clean snapshot if any malicious code is accidentally downloaded along with an intended program. This is important because it insures if anything happens time will not be wasted reinstalling a fresh operating system ensuring there is not any unwanted data on the system. The investigator can be sure that the data processed in the investigation is the intended data.

Examination Details

A security guard at Ballard industries confiscated a floppy disk on April 26, 2004 at approximately 4:45pm MST. The floppy disk was then transferred from the security administrator, David Keen to me for investigation. In order to maintain a chain of custody, the proper paper work was created and is included in figure 1. A chain of custody form also is needed to maintain who has had custody of the evidence in question, in this case the floppy. The evidence should be stored in a secure area where anyone who wants to check out the evidence must first fill out the proper information in the chain of custody form. The chain of custody form also allows an investigator to determine who checked out the evidence, when and why and when it was returned.

Evidence Custody Form

Tag #: fl-26040-RJL1
Description: 3.5 inch TDK floppy disk
MD5: d7641eb4da871d980adbe4d371eda2ad
Image Name: fl-260404-RJL1.img.gz

Chain of Custody Form

Date/Time	Released By	Received By	Reason
Date 4/26/04	Name Ballard Industries	Name David Keen	Against co. policy to remove floppy from property

[3] http://www.vmware.com

© SANS Institute 2005 Author retains full rights.

Time 4:45 pm MST	Signature	Signature	
Date 4/26/04	Name David Keen	Name Jessica Dillinger	To perform investigation on floppy and generate report of findings.
Time 6:00 pm MST	Signature	Signature	

Figure 1 - Evidence Custody Form and Chain of Custody Form

The first step taken to insure integrity of the data on the floppy was to verify the MD5sum hash that was taken by Ballard Industries when the media was confiscated. The MD5 sum of the floppy taken upon receiving the floppy is shown in figure 2.

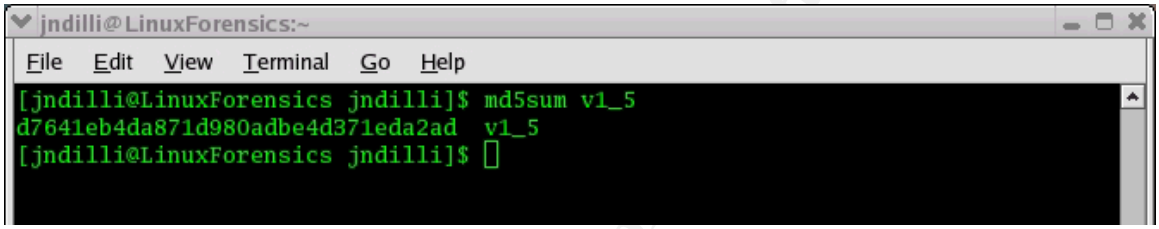


Figure 2 - MD5sum verification of floppy

This ensures that the data wasn't altered and the chain of custody was unbroken. As seen in the figure the MD5 matches the MD5 that was recorded in the evidence custody form shown in figure 1. This verifies the integrity of the date. Meaning the media that was received has not been altered since the first MD5 hash was taken. The original media should never be used for forensic investigation. Therefore, a complete backup of the information was created to maintain the integrity of the data. In the unlikely case that time stamps are accidentally touched or data lost in the investigation process, new backup can always be created and the investigation continued with the unaltered backup.

© SANS Institute 2005 Author retains full rights.

Autopsy Setup Process

The first step was to create a new case using the Autopsy program. Basic information was inserted into Autopsy pertaining to the case. This included the name of the case, a description and the name of the investigator. Next a host must be added, during this portion of the setup it was important to insert the correct time zone, in this case it was recorded as Mountain Standard Time (MST) when the security guard at Ballard Industries recorded the information about the confiscated floppy. The final step in the setup process was to add the image.

During this portion of the Autopsy setup the location of the image that needs to be investigated will be specified. The file system type must also be specified, since floppies use FAT12, it was selected from the options available in the list. It is important to complete the data integrity portion. Checking the box that calculates an MD5um before the file is imported and also the last box that verifies the MD5 has remained the same after the file is imported. This will ensure that the image has not been modified in the process of importing the image. After this step has been completed the investigation process begins.

Image Details

The investigation process began by determining what files remain on the floppy and their status. The following figure shows the MAC (modified, accessed and created) times for each file along with the size of the file. These times can be

© SANS Institute 2005, Author retains full rights.

© SANS Institute 2005 Author retains full rights.

used to show the evidence is in original state and has not been modified since receiving the evidence.

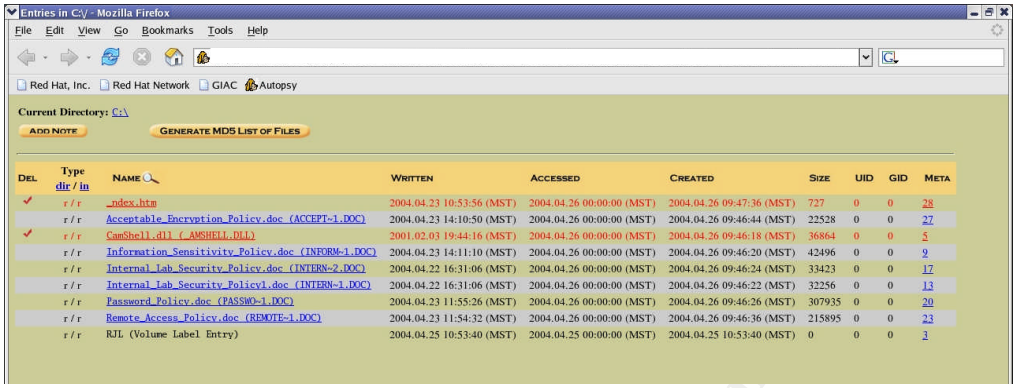


Figure 4 - File details viewed in Autopsy

There are a several things that should be noted when looking at this portion of the process. First, there are two files that have been deleted. One of them is strangely a .dll file, which is usually specific to a system and cannot be easily modified by a user. Each of these files will need further investigation to determine what they once contained. Second, there are a few files that are unusually large for Word documents, particularly the files Password_Policy.doc and Remote_Access_Policy.doc. Further investigation of these files, later in the investigation will determine why these files are so large. The next step was to view the image details.

The image detail section of Autopsy allows the investigator to see what file system is being used, in this case FAT12. The most important information shown in figure 5 is the section titled FAT CONTENTS (in sectors). This revels to the investigator which sectors contain files on them and how many sectors belong to each file.

© SANS Institute 2005 Author retains full rights.

Figure 5 - Image details of confiscated floppy disk viewed in Autopsy

Next, the permission will need to be determined this was done using a terminal in the Fedora operating system. This will show who had permissions for each of the files. Explaining who had permission to read, write and/or execute each of the files. In Figure 6 the far left column shows the permissions for each group. The first option is whether or not the item listed is a directory. This is only shown twice and refers to information that was on computer when the files were saved on the floppy. The next three characters are the permission of the owner of the file. As shown in the figure the owner of the file has permission to read, write and execute each of the files. The next three characters represent the group's permissions. The group only has permission to read and execute. The last three characters represent the world's permissions, which only have permission to execute the files.

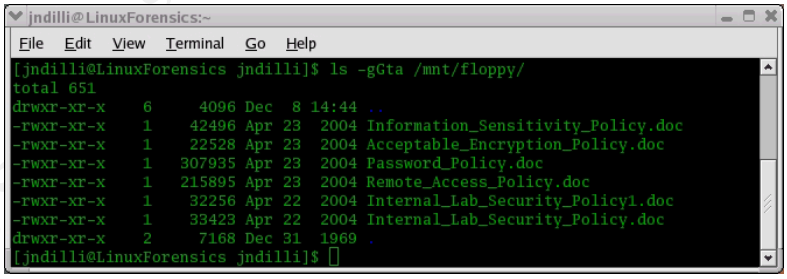


Figure 6 - file owners

The dates shown in figure 6 are dates signifying when the files were last modified.

Analysis of Files

© SANS Institute 2005 Author retains full rights.

Each of the files on the floppy was examined for more clues on what Mr. Leszczynski was doing with the floppy he attempted to remove from Ballard Industries. All of the Word documents were copied, creating a backup. This is done so that the floppy used for the rest of the investigation is not touched or altered. If the files on the floppy were opened the timestamps would be changed to indicate that the files had been accessed. The timestamps need to be in their original state for the rest of the investigation. By creating backups of the files an additional copy of the floppy will not have to made using the original floppy taken from Mr. Leszczynski. This also insures that the timestamps of the files that will be used for further analysis will not be altered. The backup of the files could be opened and viewed. The files were all Word documents so they were opened in the Windows 2000 operating system that was installed in VMWare. The information found in the documents did not look suspicious. The information contained in the Word documents described Ballard Industries security policies. The only data that looked peculiar was the size of the two files, Password_Policy.doc and Remote_Access_Policy.doc. Since nothing can be determined based on the current search, all of the Word documents will be viewed in Autopsy's hex viewer to determine if there is anything suspicious that was not found when viewing the files in Word. There were only two Word documents that contained suspicious data within the file. They were Password_Policy.doc and Remote_Access_Policy.doc.

Since the initial investigation did not reveal anything that would explain the size of the two Word documents question earlier, a more in depth investigation was necessary. The following figures show the Word documents in Autopsy's hex viewer.

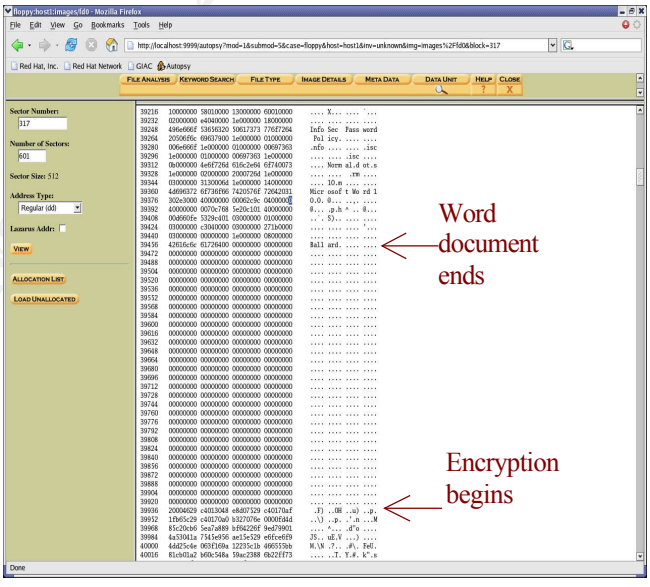


Figure 7 - Password_Policy.doc viewed in hex editor

© SANS Institute 2005 Author retains full rights.

Figure 7 shows the file Password_Policy.doc in Autopsy's hex editor. Every type of file has a signature at the beginning of the document called a header and many also have a similar signature at the end known as a footer. This makes it easy to identify in hex where that file begins and ends. Unfortunately, Microsoft Word only has a header it does not have a footer making it more difficult to determine where the file ends. Within every document there is metadata that gives information about the document. You can figure out where the document ends by reading the metadata in the hex viewer located in Autopsy. You will see in Figure 7 where the document ends there is metadata describing the name of the file, the type of file Microsoft Word and the companies name Ballard. Then you see an area where there is no data and towards the bottom of the figure more data begins. The data, also shown in figure 7, seems to be encrypted because it is not readable. Also, because the data is encrypted the file signature at the beginning of this data can not be determined which would describe what type of file this encrypted data is. When investigating the Remote_Access_Policy.doc file the same sequence of data was found within the given sectors for that file. At this point it can speculated that a tool was used to encrypt a file and inject it into Word documents. This explains why the documents sizes are so large. The next step is to determine what tool was used to encrypt and inject the file into the Word documents. Before this is done however, the deleted files should be viewed to see if they contain any information that would help the investigation.

The sectors where the two deleted files existed were inspected. The sectors that correspond to the files in questions are as follows:

File	Sectors
_ndex.htm	33 – 34
Camshell.dll (_amshell.dll)	33 – 105

Table 2 - Files and their corresponding sectors

It is interesting that both the files correspond to sectors 33 and 34. The next step was to view the data unit section in autopsy. The sectors that were viewed first were 33 and 34, which hold the first file _ndex.htm. It appeared that these sectors contained an image. Next sectors 33 through 105 were viewed. In order to view only the Camshell.dll file sectors will be eliminated at the beginning of the file to insure the Camshell.dll is the only file viewed. Autopsy allows the sectors viewed to be changed along with the amount of sectors viewed. As mentioned above it was interesting that both files point to sectors 33 and 34, this was because _index.htm was included in Camshell.dll. After sector 34 there was a large portion of empty sectors between the end of the _index.htm file and the beginning of the Camshell.dll file. This empty data was also taken out of the total sectors viewed in Autopsy. The sectors that ended up corresponding with Camshell.dll were sectors 42 through 105. The sectors that correspond to Camshell.dll were then investigated using Autopsy's hex viewer.

© SANS Institute 2005 Author retains full rights.

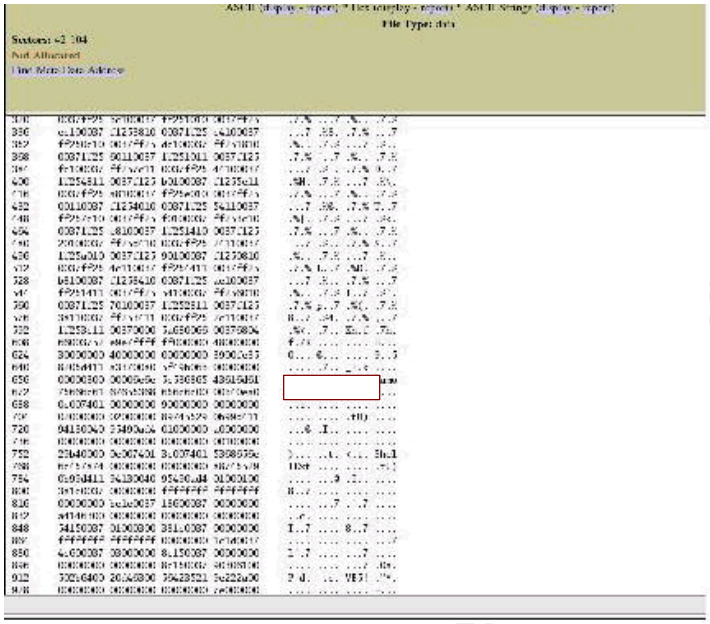


Figure 9 – Camshell.dll file viewed in hex

Within the Camshell.dll file, a reference to “camouflage shell” was made. Since all of the files have been investigated, the camshell.dll reference should be researched, to determine what it is referring to

© SANS Institute 2005 Author retains full rights.

Examining Camouflage

The next step was to determine what exactly the camouflage shell was. A search was performed using the Google search engine on “camouflage files”. After reviewing the sites that were found by the Google search a definition of camouflage was found. “Camouflage allows you to hide files by scrambling them and then attaching them to the file of your choice. This camouflage file then looks and behaves like a normal file, and can be stored, used or emailed without attracting attention.”[4] A few more facts about camouflage to note, camouflage only runs on the Windows platform and allows the user to add extra security by password protecting the camouflaged file. Lastly a camouflaged file can be camouflaged again. In order to determine that camouflage is the same as the tool that Leszczynski used. A hash was taken of the camouflage 1.2.1 camshell.dll file. This file is saved into C:\Program Files\Camouflage when camouflage was downloaded onto VMWare running Windows 2000. The hash of that file was compared to the deleted Camshell.dll file that was found on Mr. Leszczynksi’s floppy. This comparison was done in WinHex so that only the blocks that correspond to the camouflaged files could be evaluated. This step is important because without it the data would be different causing each file to have different MD5sums. The MD5sum for each of the files are as follows:

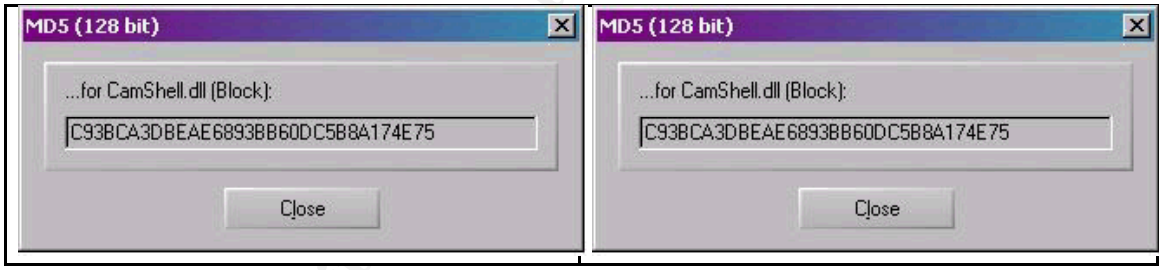


Figure 10 - Comparison of Camshell.dll MD5 hashes

The image on the left labeled CamShell.dll is the file saved on the operating system, in this case the VMWare machine when camouflage was downloaded for testing. The image on the right, labeled Sector33_105.dll is the deleted CamShell.dll file on the confiscated floppy. The name is based on the sectors that were taken from the floppy in Autopsy, 33 through 105. It was found and explained during the image details of this investigation, that these sectors correspond to the entire CamShell.dll file that was deleted. Based on the hashes found it can be concluded that the tool Mr. Leszczynski was using to attempt to hide information was camouflage 1.2.1 and that the information is hidden in the document.

[4] camouflage.unification.com/Overview.html

© SANS Institute 2005 Author retains full rights.

Further searches on the Internet lead to a very useful website that explained how camouflage works, was titles (easily) Breaking a (very weak) steganography software: Camouflage. The author of the site who went by Guillermito, explained that the algorithm that camouflage uses is very weak and very easy to detect and break. Following the same steps shown on Guillermito's website a few tests were conducted to determine what a camouflaged file looks like. Certain precautions were taken because the tool could contain malicious code. First the camouflage software was downloaded from packetstormsecurity.nl/crypt/stego/camouflage and installed in the VMware image running Windows 2000 as the operating system. This is a safe environment to test unknown tools and code. A snapshot was taken prior to downloading the tool. This is done to ensure that if malicious code was found along with the tool camouflage that, VMware could be reverted to the prior state of the machine. This would remove camouflage and any damage to the machine would be reverted to a clean state. A Word document was created containing the text "This is a test!" to find the camouflage option the file was clicked on with the right mouse button. Camouflage was opened and a dialog box appears allowing the user to choose the file to camouflage. Next, a file needs to be chosen for the camouflaged file to look and act like. For the test a picture of a reflection pond located at the University of New Mexico was chosen, the name of the file is art05.jpg. Camouflage created a new file the default file name is the same as the original art05.jpg for the test this file name was maintained. This file appeared to contain the same contents as the original however, it had the test file encrypted and injected into the picture. The last step is the option to add a password to protect the camouflaged file. The password "test" was used to protect the camouflaged test file. The images were opened and compared.

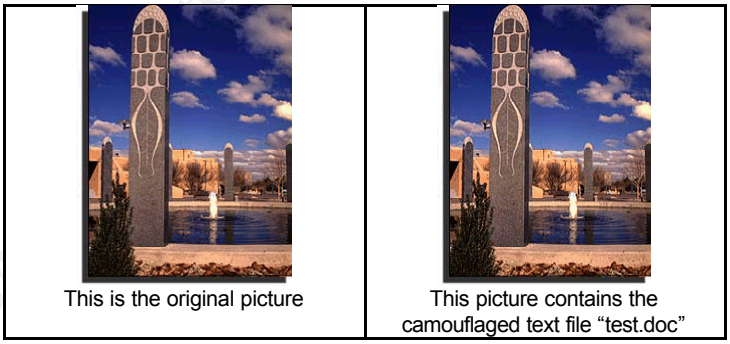


Figure 11 – Comparison of images

To the naked eye the images look identical. However, there are two attributes that can be looked at to prove that there is a difference between the original file and the one with the test.doc file was injected into it. First, the size of the original picture is 18KB the size of the image containing the camouflaged text file is 38KB. This is a very obvious difference between the two. The second test to determine if the documents are different is by viewing them in a hex editor.

© SANS Institute 2005 Author retains full rights.

When the document that contains the camouflaged file was looked at in the hex editor it was a little more difficult to determine where the picture ended and the camouflaged file began. This is because of two reasons, first there was not a distinction between the files or a space where there was no data present. Second, the data in a picture looks similar to encrypted data because it contains random characters. The end of the file could be found by comparing the original file to the file containing the camouflaged text file in a hex editor. For this experiment WinHex was used to view the files. WinHex is a hex editor that runs in the Microsoft Windows platform, it makes it easy to view both files in the same window. This helps to determine where the camouflaged file began. The last line of the original picture was copied and pasted into the search box and the search was conducted in the image that contained the camouflaged file. Once found it can be assumed that the following line is the first line of the camouflaged file. An image was used in the original experiment because it allows a person to see that injecting data into another document does seem to change the outer characteristics of the file. This is very easy to see with an image, as shown in figure 11. The second experiment will allow the Word documents, created to test the effects of camouflage, to be compared with the documents Mr. Leszczynski attempted to remove from Ballard Industries property.

To duplicate the experiment with a Word document each of the steps listed above were repeated using a Word document instead of a picture. The same test document was camouflaged and injected into the Word document using the same password, as it was with the picture. When viewed in WinHex the results were similar to the Word documents found on the floppy confiscated from Mr. Leszczynski. There was a portion of the file that did not contain any data. Further down began the portion where data that looked encrypted resides.

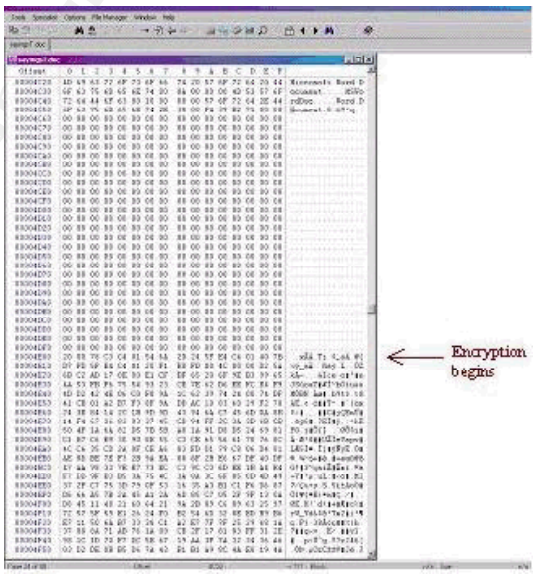


Figure 12 - Beginning of encrypted data in the test Word document

© SANS Institute 2005 Author retains full rights.

The previous image shows the file that was used to test camouflage. In this figure the similarity between the test document and the Word documents found on the floppy can be seen. The next step was to determine whether or not the files that may have been camouflaged are password protected and what the passwords are. This would lead the investigation a step further by uncovering what if anything was hidden in the Word documents found on the floppy. The first step was to see if there was a password, each of the files were right clicked on and the uncamouflage option was chosen. When asked what to enter the password nothing was entered. An error box appeared stating that either the correct password was not entered or the file is not camouflage. It is assumed that a password is protecting each of the files. The next step of the investigation is to determine how camouflage encrypts the password. This will allow passwords protecting the files to be cracked.

Deciphering Camouflage

Based on the information found, camouflage disguises the password by XORing it. XOR is an encryption algorithm that for example is used for example by taking one string, for example a password and XORing it with a key to created an encrypted password. Based on this description if two parts of the equation are known or found than the third can easily be determined. During the search on XOR it was found that it is a very common encryption algorithm and that it is not a very secure means of encryption. A presentation done at black hat titled "Steganography, Steganalysis, & Cryptanalysis" was found while doing a Google search on XOR. Michael T. Raggo, CISSP who is the principle security consultant for VeriSign, created the presentation for a Black Hat conference. Within the document Raggo describes how to find the string used by camouflage. He explains that the first step is to locate the password within the camouflaged document. This will allow the key to be deciphered using a known password and the string it corresponds to. The hex string associated with the password is located in a fixed position. The encrypted password found within the document was "76 F0 09 56". The password "test" was translated into hex, which is "74 65 73 74". The next step was to reverse the XOR to find the key. The string and the password was XORed against one other to find the key. There is an option located in the scientific calculator found in Microsoft Windows that computes the XOR equation. This is the weakness of XOR, only two parts of the equation were known, the password and the string. From these the key could be determined. Finding the key is an important step because it can now be used to decipher the passwords in the files camouflaged by Leszczynski.

© SANS Institute 2005 Author retains full rights.

Encrypted Password		Password		Key
76	*XOR*	74	=	02
F0	*XOR*	65	=	95
09	*XOR*	73	=	7A
56	*XOR*	74	=	22

Table 3 - Finding the key to camouflage

The key is “02 95 7A 22”. This same technique was described in the "Cracking Camouflage” paper written by Guillermito. In his paper, which was the basis for the Black Hat presentation written by Raggo, Guillermito extended the key a bit further, “02 95 7A 22 0C A6 14 E1 E1”. He also created a utility based on the information just found on the weaknesses in the XOR encryption that camouflage uses. The utility, Camouflage_Password_Finder, can supposedly crack the password that protects the camouflaged files contained within other files. Both using the key and the utility created by Guillermito will be used to decipher the passwords. This will verify that the passwords are correct, by using both techniques to determine the passwords that were set for the two Word documents in question, Password_Policy.doc and the Remote_Access_Policy.doc. This allowed for more certainty when concluding the findings on the passwords.

The first technique used to determine the password was to use the key and XOR it against the data found in each of the Word documents in question. The first step is to find the position where the encrypted password should reside in each of the documents. In the “Cracking Camouflage” paper Guillermito states that the password is stored in a fixed location, which is at –257 relative to the end of the file according to Guillermito. This is fairly easy to find, it is obviously very close to the end of the document. The string containing the password in hex, is located after all of the encrypted data, which is the camouflaged file. It is one of the last blocks containing data in the files. Each of the files viewed, so far, end with a lower case t followed by an upper case T. Depending on the hex editor a dot (.) or a character lies between the two. This is not the encrypted password but is a terminating symbol that can be used to determine the location of the password. The actual password string is located just above the symbol (t T). The following image shows Password_Policy.doc in Autopsy’s hex viewer. An arrow indicates the area where the encrypted password string is located. Also shown in the image is the terminating symbol “t .T” that was discussed earlier.

© SANS Institute 2005 Author retains full rights.

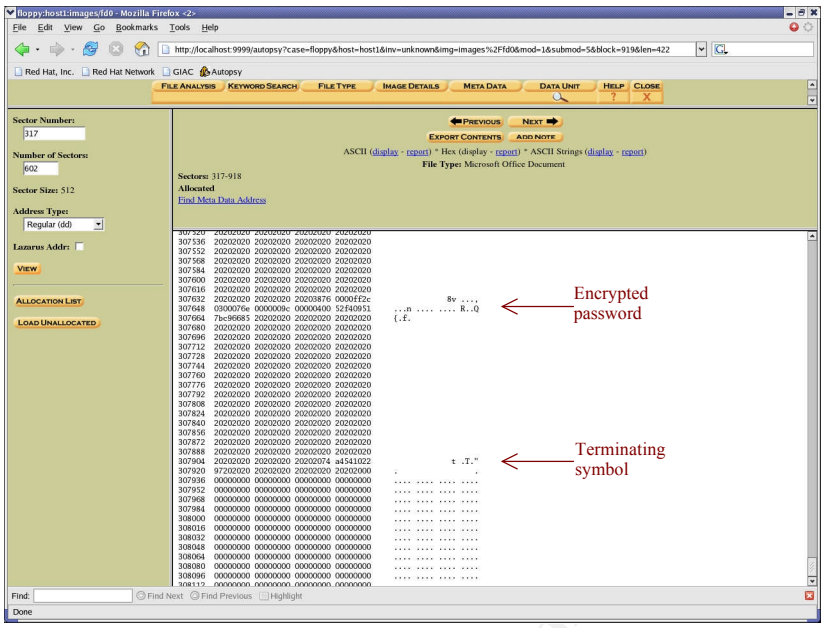


Figure 13 - Location of encrypted password in Password_Policy.doc

This is very similar to what was found in Remote_Access_Policy.doc, which is shown in Figure 14.

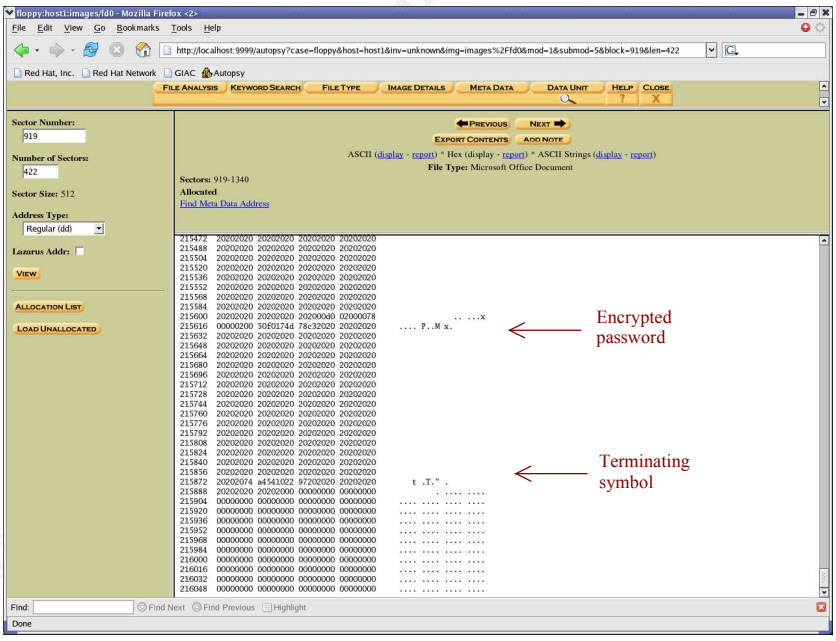


Figure 14 - Location of encrypted password in Remote_Access_Policy.doc

The next step was to reveal what the actual passwords were. To determine the passwords the encrypted data found in each file will be XORed with the key. The first block from the area where the password was located in the Password_Policy.doc file will be XORed. The scientific calculator in Windows

© SANS Institute 2005 Author retains full rights.

was used to XOR the equation.

52 F4 09 51
XOR
02 95 7A 22
=
50 61 73 73

The first four bytes of the string found within the Word document, which is XORed with the first four bytes of the key. The last line indicates in hex the solution. The last step in deciphering the password is to convert the solution from hex to ASCII. A simple ASCII chart can be used to make this conversion. The conversion concluded that the password for the Password_Policy.doc file is “Pass”. This does not seem to be the complete password. This observation comes from an important fact about how camouflage stores the password with the file. This is discussed in Guillermito paper. He determined that the length of the password corresponds to the length of the string. For example, a four byte password corresponds to a four byte string located in the document. When reviewing the document it is obvious that the password is more than four bytes long. Based on this information the last four bytes of the password or taken and XORed with the second four bytes of the key. This generates the solution, which is the last line of the equation listed below.

7b c9 66 85
XOR
0C A6 14 E1
=
77 6F 72 64

The same method of conversion was followed to convert the hex string into ASCII. After the conversion it was found that the second portion of the string converts to “word” the password for this file is “Password”. Before the password was tested on the document and any files uncamouflaged, the same procedure was performed on the Remote_Access_Policy.doc file. Based on what was found in the first Word document, that the password string is surrounded by spaces, the entire string of data was used. This consisted of one and a half blocks or six bytes of data.

50 F0 17 4D 78 C3
XOR
02 95 7A 22 0C A6
=
52 65 6D 6F 74 65

After converting the password in hex, found after XORing the key with the encrypted password found in the file, to ASCII it is determined that the password

© SANS Institute 2005 Author retains full rights.

to the Remote_Access_Policy.doc file was “Remote”. After the passwords were found for both of the files it can be seen that Mr. Leszczynski used the first word of the file names as the password for the corresponding file. In order to determine if the utility provided by Guillermito works, it will also be used to decipher the passwords.

In order to ensure the safety of the operating system the Word document that was used to test camouflage in the original experiment was used to test Guillermito’s utility. The utility was downloaded into the VMWare environment running Windows 2000. As always this environment was set up so if any unwanted or malicious code was downloaded along with the intended program, it could be removed from the operating system or VMWare could be reverted to a safe state. After concluding that the utility was safe it was tested on the Word document. There were two files that were needed to run this utility. The first is the executable and the second has an .asm extension, which stands for assembly code. This portion of the utility can be opened in notepad and includes a readme and the code for the program which it written in assembly language. This was most likely done because the size of the program would be smaller and more efficient than if the program was written in a high-level program language. The utility, Camouflage_Password_Finder, works by finding the end of the password, which as stated early has a fixed location. Next, it XORs the password string with the key, finally the solution is translated from hex to ASCII. This is the same procedure used above to determine the password.

As expected the results of the utility matched the results that were found earlier in the process. To use the utility, the executable portion of the file is selected. The utility brings up a dialog box where the file, that is suspected to contain a camouflaged file with a password, can be chosen. While using the program a few interesting responses were found. First, if the file that is suspected of having a camouflage password has a file attribute of read only the program does nothing. A dialog box should appear, however this does not happen in this case. Second, if the file is not camouflaged or does not have a password protecting the camouflaged data, the dialog box will appear after the file is selected but it contains only random data. Finally, if the file does contain a password, the password will appear in the dialog box and will also automatically be copied to the clipboard. This is done so that the password can be pasted into camouflage without having to retype the password. This was what occurred when the Camouflage_Password_Finder program was used to uncover the passwords of the two documents in question Password_Policy.doc and Remote_Access_Policy.doc.

© SANS Institute 2005 Author retains full rights.

	
Password for Password_Policy.doc	Password for Remote_Access_Policy.doc

Table 4 - Recovered camouflage passwords

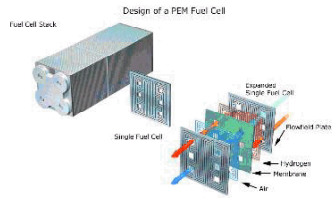
Table four shows the Camouflage_Password_Finder program revealing the passwords to the Word documents. They match the passwords that were found in the previous example. It was a good practice to uncover the passwords by hand and also by using the tool. There were a few bugs that were found in the tool that had to be discovered with trial and error. By using the two methods it can be guaranteed with a high level of confidence that the two passwords were uncovered correctly.

Recovering Camouflaged Files

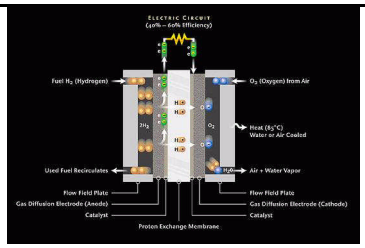
The final step is to test the passwords on the corresponding Word documents. This will reveal the camouflaged data. After the password “Password” was inserted into the Password_Policy.doc camouflage prompts the user to save the camouflaged file. A folder was created in the C: directory named “camouflagedFiles” the file that was recovered was saved to that folder. The recovered files where PEM-fuel-cell-large.jpg, Hydrocarbon fuel Cell page2.jpg, and pem_fuelcell.gif. Each of these files were images and will be viewed after all of the camouflaged files are recovered. The password “Remote” was then inserted into the Remote_Access_Policy.doc file and the camouflage file was saved to the same location as the other recovered files. There was only one file recovered from this Word document: Cat.mdb. The extension .mdb refers to a Microsoft Access database. Finally, each of the recovered files can be viewed in a safe environment to determine what exactly Mr. Leszczynski was trying to acquire from Ballard Industries. This will be done in VMWare running Windows 2000 so that the database can be viewed in access. As will all files of unknown state a snapshot will be taken in VMWare, prior to saving the files in case these files contain malicious code it can be reverted to a clean and safe state. The first file to be reviewed was PEM-fuel-cell-large.jpg. This image describes the design of a PEM fuel cell. It shows a single fuel cell, a fuel cell stack and an image of the single fuel cell expanded. This expansion explains the design of a single fuel cell and each part in sequence. The second image viewed was “Hydrocarbon fuel Cell page2.jpg.” Based on the name of the file it is most likely the second page of a document. The image contains an image that is probably out of a scientific document explaining how improvements of the fuel cells performance can be found. It also discusses different experiments that were conducted and the

© SANS Institute 2005 Author retains full rights.

outcomes. The next image, pem_fuelcell.gif contained an image of a Hydrogen fuel cell battery. This image describes how the battery is designed, including each element and how it works. These figures are almost certainly proprietary information that belongs to Ballard Industries.



Hydrocarbon fuel Cell page2.jpg



pem_fuelcell.gif

The final image viewed came from the Remote_Access_Policy.doc. The file Cat.mdb is an access database containing customer information. This database stored the customers first and last name, phone number, company name, and address including city, state and zip code. The most significant information found is the database was the account name and password for each of the clients. This supports Ballard Industries suspicions that someone, perhaps an insider, was stealing customer information and proprietary information on their new specialized fuel cell battery. The information found in this database is listed below.

First	Last	Phone	Company	Address	Address1	City	State	Zipcode	Account	Password
Bob	Esposito	703-233-2048	Cook Labs	245 Main St		Alexandria	VA	20231	espomain	y4NSHMNf
Jerry	Jackson	410-677-7223	Double J's	11561 W. 27 St.		Baltimore	MD	20278	jack27st	JLbW3Pq5
David	Lee	866-554-0922	Tech Vision	300 Lone Grove Lane		Wichita	KS	30189	leetechv	O1A26a3k
Marie	Horton	800-234-king	King Labs, Inc.	700 King Labs Ave	Suite 900	Biloxi	MS	39533	hortking	Yk7Sr4pA
Lenny	Jones	877-Get-done	Quick Printing	99 E. Grand View Dr		Omaha	NE	56098	joneeast	868y48RH
Jeff	Hayes	404-893-5521	Big Sky First	90 Old Saw Mill Rd		Billings	MT	59332	hayeolds	3R30bb7i
Roger	Forrester	210-586-2312	TCFL	188 Greenville Rd		Austin	TX	77239	forrgree	si4OW8UV
Edward	Cash	212-562-0997	E & C Inc.	76 S. King St	Suite 300	Santa Barbara	CA	80124	cashking	Of8uQ1fC
Steve	Bei	616-833-0129	Island Labs	65 Kiwi Way		Honolulu	HA	93991	beikiwiw	JDH20u26
Jodie	Kelly		Data Movers	7256 Beerwah Ave.	Suite 110	Wetherby	U.K.	LS22 6RG	kellbeer	tmu0ENOk
Patrick	Roy		The Magic Lamp	4150 Regents Park	Row #170	Calgary	CAN	R4316DF	roythema	rJag6Q0O

Table 5 - Cat.mdb, Access database storing customer information

© SANS Institute 2005 Author retains full rights.

Conclusion

The evidence shows that the information contained on the floppy was more than just Word documents about Ballard's company policies. The Word documents had been used to cover files that had been intentionally encrypted using the tool camouflage. These files that had been hidden included proprietary information that belonged to Ballard Industries. Within the floppy camouflaged pictures were found that explained the function of the new fuel cell battery designed by Ballard Industries. The floppy also contained a camouflaged database that contained client's personal information including usernames and passwords. Mr. Leszczynski had access to the information and did have possession of the floppy when it was confiscated. It can be concluded, by reviewing the information found within the camouflaged files that Mr. Leszczynski was trying to remove sensitive information from his place of business, Ballard Industries. Based on this information it cannot be determined what Mr. Leszyczynski's intensions were.

However, information was provided at the beginning of the investigation that explained that Ballard Industries had recently began an investigation on the leakage of information on their fuel cell battery. This was based on the knowledge that Rift Inc. Ballard's largest competitor had recently began manufacturing the same fuel cell battery. Ballard Industries believed Rift Inc. had received proprietary information somehow. It was also suspected that someone had leaked information about Ballard's clientele along with the proprietary information on the fuel cell battery. This information and the fact that Mr. Leszcznski was caught with a floppy containing intentionally concealed information implicates Mr. Leszczynski as the source of the leak. This cannot be concluded based on the information. A new investigation will need to begin in order to determine whether or not Mr. Leszczynski was responsible for the breach of information.

Legal Implications

The first set of consequences that Leszczynski faces deals with the policies of Ballard Industries that he broke. It is against Ballard's company policy to take media, in this case a floppy, out of the research and development laboratories. This was the first and most obvious rule that he broke. This will be dealt with according to the consequences that have been set by Ballard's company policies. It will be more severe because the information taken from the property was not policy documents, as it seemed upon first glance. The data that he was taking from Ballard was proprietary information on the design and development of their new fuel cell battery. The floppy also contained client information stored in an Access database. Mr. Leszczynski had access to this information because

© SANS Institute 2005 Author retains full rights.

of his job position at Ballard Industries. If Ballard Industries had a privacy policy in place Mr. Leszczynski violated it when he copied client's personal information.
This may lead to legal actions being taken against Mr. Leszczynski. Not only could the clients whose information was taken from Ballard Industries file suit against him, there may be federal or state privacy laws broken by his actions.

The next set of laws that would be applicable would be at the state level. Based on New Mexico state laws Mr. Leszczynski violated Article 45 "computer crimes" section 30-45-5 unauthorized computer use. This law states that a person who uses authorization to "access, use, take, transfer, conceal, obtain, copy, or retain" computer property, which includes databases, or any part thereof. Leszczynski used his job position and authority to access, conceal and copy information from Ballard's client database. He also accessed, concealed and copied data pertaining to a new fuel cell battery that was created by Ballard. Ballard Industries alone had the instructions on how to create this new fuel cell battery. This fact is pertinent to the investigation and determination of laws that were broken because it clarifies that the information on the fuel cell battery and on Ballard's clientele was not known by other business in the industry. The level of punishment is dependant on the amount of monetary damage done to the business or individual, in this case Ballard Industries. There are five levels of punishment from a petty misdemeanor to second-degree felony, many of these may also include imprisonment and monetary restitution for financial loss sustained by the individuals actions.

Another state law broken by Mr. Leszczynski was the N.M. Stat. Ann.§§ 57-3A-1 et seq. This is a version of the uniform trade secrets act. First a trade secret should be defined to determine if the fuel cell battery fell into this category. "A trade secret may consist of any formula, pattern, physical device, idea, process, and compilation of information."[5] This information must both provide the owner with a competitive advantage in the marketplace and must be handled so that competitors or the public cannot easily learn about it[5]. Based on this definition the fuel cell battery created by Ballard Industries could be considered a trade secret.

Because Ballard was the only company in its industry that had the design and was producing the fuel cell battery, this gives Ballard a competitive advantage in their industry. However, it is not know how well Ballard protected the design of the fuel cell battery. Ballard Industries, the owner of the trade secret has the right to restrict it employees who have contact with the information because of there job title from disclosing the information. Mr. Leszczynski did have access to the information because he was the lead process control engineer on the fuel cell battery project. He does not however, have the right to disclose the

[5] http://cobrands.business.findlaw.com/intellectual_property/nolo/faq/90781CA8-0ECE-4E38-BF9E29F7A6DA5830.html

© SANS Institute 2005 Author retains full rights.

information and procedures on how to create the fuel cell to anybody. If it is found that he disclosed any information pertaining to the fuel cell battery project to Rift Inc. he could be found guilty of intentional theft of a trade secret. This is also dependent on the how well Ballard Industries tried to protect the information of the fuel cell and if they tried to establish a nondisclosure agreement with its employees.

If each of these requirements for the information of the fuel cell battery were considered a trade secret they would be protected under the laws. If Mr. Leszczynski is found guilty of stealing trade secrets, he could be punished according to state and/or federal laws. The federal law that is applicable to this situation is the Economic Espionage Act of 1996 section 1831. Under this act, if found guilty of economic espionage Leszcznski could serve up to 15 years in prison and/or could be fined up to $500,000. If Rift Inc. is found guilty of conspiring with Leszcznski in order to obtain trade secrets the company could be fined or if an individual at Rift Inc. was found guilty they may also be subjected to the same punishment as Leszcznski.

© SANS Institute 2005, Author retains full rights.

© SANS Institute 2005 Author retains full rights.

Additional Information

For more information on camouflage:

http://guillermito2.net/stegano/camouflage/index.html

Information on cryptography and steganography:

http://www.blackhat.com/presentations/bh-usa-04/bh-us-04-raggo/bh-us-04-raggo-up.pdf

For more information on every aspect of digital forensics, a reference page:

http://staff.washington.edu/dittrich/forensics.html

For more information on file signatures:

http://www.garykessler.net/library/file_sigs.html

© SANS Institute 2005, Author retains full rights.

© SANS Institute 2005 Author retains full rights.

Part Two: Forensic Tool Validation: WinHex

Scope

For the second portion of the certification a tool will be analyzed to determine whether or not it would be useful to a forensic investigator. The tool that will be analyzed is WinHex version 11.9. WinHex is a hex editor that runs in the Windows operating system. This tool allows the user to view a file, several different types of media or RAM. X-Ways Software Technology AG offers this tool. X-Ways headquarters are located in Germany. WinHex is their main product and has over 12,000 registered users throughout the world. This tool is very useful in allowing the user a long list of capabilities. Including being a disk editor, file editor, RAM editor and is capable of cloning an entire disk.

The major goals of the experiment are to determine if WinHex does all of the things it claims it is capable of doing and to conclude if WinHex is a valuable forensic tool. Answering the following questions will help to complete the conclusion.

1. Does WinHex accurately display code without dropping important information such as headers or footers? How can this be used?
2. How well does WinHex wipe a hard drive? What are the wiping options?
3. Can WinHex open files that are as large as the website claims, several terabytes?
4. What effect does WinHex have on the evidences time stamps, in hex editor mode?

Tool Description

WinHex version 11.9 is a powerful file editor, disk editor, and RAM editor. It is a commercial tool that costs from approximately $60 to $350. The price depends on what license typed is bought. A personal license type includes the basic functions, which include cloning, wiping, viewing and manipulating files including files that cannot normally be changed, Data recovery and many other tasks. The next license type is the professional this includes everything a personal license allows and the scripting and API features. Next the specialist also includes the specialist menu with allows the user to have the ability to gather free, slack and inter-partition space. It also has a new search feature.

© SANS Institute 2005 Author retains full rights.

Lastly, the forensic license type includes X-Ways Forensics feature, which contains a plethora of new functions. X-Ways Forensics is an environment created specifically for computer
forensic examiners. X-Ways also has an evaluation version for users to test before investing in the full version. For this experiment the evaluation version of WinHex was used to determine the most basic functions of WinHex and how well they perform.

The basic hex editor will be the focus of the investigation. However, there are some other functions of this tool that are very interesting and should be looked at The start center is the window that by default is shown when the program is started. This is where a file can be opened and viewed in hex. This is important to forensic analysts because this is where they can view every part of a file, drive or the RAM. Based on WinHex's manual the speed of the tool is supposed to be very fast. This will be tested later in an experiment. The size of the media that WinHex can view at once is dependant on the type of file system. With FAT16 only 2GB is supported. FAT32 can support up to 4GB and NTFS supports an almost unlimited file size. X-Ways claims that the NTFS file system can handle several terabyte files. However, if the user looks at a very large file an equally large temporary file will also need to be created. The temporary file will only be created if bytes are being removed, or inserted into the file.

The tool can be found at found at X-Ways website which is www.x-ways.net/winhex/index-m.html. WinHex can also be found at many other free download websites. This tool is very easy to find on the Internet, however there is another tool that goes by the name WinHex this tool is used for number conversions.

This tool has several capabilities the following is a list of the most basic functions:

- Disk editor, file editor, RAM editor
  WinHex is a binary editor that allows the user to access every part of a piece of media.
- Directory Browser for FAT and NTFS
  Lists all files including deleted files and directories along with all their details.
- Hard Drive cleaning and wiping
  Wipes the drive, gives the user the chose of what to overwrite the disk with, ones zeros random or any sequence. The user can also choose how many wipes the tool uses.
- Data Interpreter and Data analysis
  Can interpret any type of data and can find out what kind of binary data the user has.
- Binary or text search
  The tool can search for any type of data.

© SANS Institute 2005 Author retains full rights.

- Position manager
  Automatically saves the logged occurrences of strings searched for.
- Checksum
  WinHex can calculate many different types of hashes.
- Data Recovery
  Files can be automatically recovered. This can be done by name or by type.
- Partition recovery/Boot record recovery
  WinHex allows the user to edit several different file types boot records, including FAT12, FAT16, FAT32 and NTFS.

Most of these capabilities will be, at the least reviewed to ensure that the company X-ways, is accurately describing the functionality of their tool.

Based on the capabilities described above WinHex would be used for many different forensic circumstance. A forensic investigator would be able to use the editor portion of WinHex to view a file and increase their understanding of what is contained within that file. This is very useful when comparing files to determine what the difference between the two. In order to determine if the two files are identical WinHex could be used to compute a number of different hash values. The types of hash values that WinHex is capable of calculating, CRC16, CRC32, MD5, SHA-1, SHAW-256, PSCHF. As stated in the "Computer Forensics Incident Response Essentials" book by Warrren Kruse and Jay Heiser "a cryptographic hash algorithm is a one-way form of encryption, taking a variable-length input and providing a fixed-length output." At this point in time it is considered to be computationally impossible to forge a hash algorithm such as MD5 or SHA1. Both of theses hash algorithms can be computed using WinHex.

A forensic investigator can use WinHex to view a directory in the FAT or NTFS file system. This includes files that have been deleted. This is beneficial to the investigator because it allows them to see if any files have been deleted. Then they could use the data recovery capability to retrieve the information that has been deleted and the sectors associated with that file are marked as unallocated. WinHex can then be used to view the file and its contents to determine why the file was deleted.

WinHex can also be used to prepare a machine for forensic analysis. The wiping option in WinHex allows the user to wipe a file or drive with as many wipes as the user desires. The wipe option in WinHex can be set to any sequence of ones and zeros, including a random wipe. This is important because the amount of times the file or drive is wiped is dependent on the significance of the data in the file or located on a piece of media.

Another useful function in WinHex is its ability to search for any type of data.

© SANS Institute 2005 Author retains full rights.

The forensic investigator can search for a known string in ASCII for example a name or phrase. This is most useful in text documents or RAM where the forensic investigator is searching for remnants of a file that holds information that is pertinent to an investigation. This function is also helpful when comparing two documents and the investigator is trying to determine what the difference. This was used in the part one of this certification. A file was tested to see where the injected information was stored. This was done by taking the file without the injected information coping the last line of data and searching for that line in the file that contained the injected data. This made it easier to determine where the end of the first file was located and which in turn pointed to where the beginning of the injected data was located.

WinHex is a single tool that incorporates many different functions that help the forensic investigator examine a file or any type of media. This is an important aspect of WinHex because it is one tool that has many functions. There are most likely numerous forensic investigators that have countless tools that they use for investigative purposes. WinHex allows the user to have one tool with many different functions. Another important characteristic of WinHex is that the functions that come with a basic license of WinHex are enough to get the job done. This is important to businesses that are financially strapped and cannot afford a $350 tool for the system administrator to use when their systems become compromised. However, WinHex also plays to the corporations that can afford such a tool and gives them extra functions. They also give discounts to many different government agencies, police agencies and government funded laboratories.

There was no evidence to support that WinHex is a statically compiled, so it is assumed that it was not. With some reverse engineering of WinHex it could most likely be determined whether or not it was statically compiled. However, this is out of the scope of this investigation. Since, WinHex is a commercial tool and the source code is not included it is difficult to determine what it system libraries are accesses.

Test Apparatus

The computer that the tool was test on had an Intel Pentium four 3.2GHz processor with 512 megabytes of RAM containing a Gigabyte motherboard. The computer was running Microsoft Windows 2000 operating system. The operating system was updated with the most recent patches, at the time this paper was written the computer was updated with service pack four. The hard drive had a capacity of 60 gigabytes. The computer also had a floppy drive and a CD-ROM drive.

© SANS Institute 2005 Author retains full rights.

WinHex was installed into the Windows 2000 operating system. This process is simple the zipped file is downloaded from the website. A tool will be needed to unzip the files. Winzip was installed and used to unzip the files that are needed to run WinHex. There were not any other requirements for WinHex.

© SANS Institute 2005, Author retains full rights.

© SANS Institute 2005 Author retains full rights.

Environmental Conditions

All of the tests that will be preformed on WinHex will be executed on the computer described in the section titled Test Apparatus. The computer used to complete the test was not connected to the Internet or a network. All of the tools were needed to conduct the experiments, were downloaded on a separate machine that had Internet connection.

There should not be any outside forces that would interrupt the experiments. This is due to the computer being a standalone machine. The only opportunity something or someone could interfere with the experiment is while the tools are being downloaded. The integrity of the tools is based off the authenticity of the website they are downloaded from. This is because a MD5 or similar hash is not offered to ensure the program that is downloaded is identical to the program that the site offers. The files were scanned for viruses before they were saved to the drive ensuring that the files do not contain any malicious code.

Description of the Procedures

The preparation of the machine and the installation of the files will be explained in step-by-step instructions.

1. The test machine was prepared with a fresh install of Windows 2000. The patches were updated to insure that the service pack and any other security holes were patched with the most current updates from Microsoft's webpage.
2. WinHex and WinZip were downloaded from a trusted machine, that we will refer to as the research machine, also running 2000. This machine was also up to date on patches. The tools were downloaded onto a new CD.
3. The CD was inserted into the machine that the tests will be performed on and WinZip was installed. This is a simple procedure. The executable winzip.exe is run and the instructions are followed.
4. WinHex is installed second, this is done by first opening the zipped file in WinZip. The files that are unzipped must be extracted to a folder on the machine. The default is C:\winhex. Next, winhex.exe was ran, a wizard was invoked to walk the user, installing the software, through the installation steps. There will be an option to put a shortcut in programs. This is encouraged because it is useful to the investigator.

© SANS Institute 2005 Author retains full rights.

The documentation will be stored in a folder created on the test machine. The files that will be saved in this folder will be named based on what they contain. To protect the test results the test machine remains a standalone machine throughout the procedure. The files will be saved as read only so that they cannot be modified to protect the integrity of the test results.

Criteria for Approval

Each of the results should coincide with what the company that created WinHex claims it is capable of doing. WinHex is a commercial tool and the source code is not available. This makes it difficult to determine if the tool obtains any information from the system. It does not however, receive any information from the network. The tool should not manipulate any evidence, as long as the file is opened as read only.

Data and Results

The steps that were taken to determine if WinHex accurately displays the information in a file are as follows:

1. Several files were zipped using WinZip. The files that were used for this experiment were pictures that were taken at the San's conference in Monterey, California.
2. The next step was to determine what the header and footer were for zipped files. A Google search was performed on the research machine to determine what the signature was. It was found that the file should begin and end with PK.
3. The zipped file was opened in WinHex and the beginning and end of the file were examined to determine if the information could be located.

The results of this experiment were as expected the headers and footers of the zipped file were kept in the file. This is important to a forensic investigator because with this information viewable in WinHex they can determine what type of file is being viewed. The following figures show the header and footer of a zipped file, in WinHex. The header and footer which are indicated by the characters PK is outlined in red.

© SANS Institute 2005 Author retains full rights.

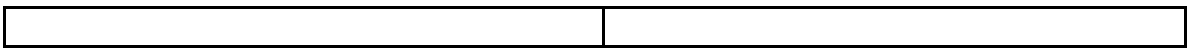


Figure 15 - Zipped file viewed in WinHex

The next test was done to determine how well WinHex wipes a drive. To save time a floppy was used in the place of a hard drive.

1. A floppy was taken with a file that had been saved on it and opened in WinHex.
2. The File Manager option was chosen from the menu. From that option wipe securely is selected. This opens a dialog box where the file or drive that the user wants to delete can be chosen. The floppy drive was selected. However, a problem occurred WinHex would not allow the drive to be selected only the file on the floppy. The file was chosen to see the next step.
3. Another dialog box appeared allowing the user to choose what the passes consisted of. There was a help option on this dialog box, it was selected to get a better understanding of how WinHex wipes the drive. It also explains how the data should be inserted into the dialog box.

4. When the help option was selected some important criteria was gathered. WinHex's wiping function can be used to wipe/sanitize slack space, free space, unused NTFS records, or an entire piece of media. This could be useful for a forensic analyst to ensure that their machine is clean and can be certain that information from a previous investigation will not interfere with the current investigation. The help file also explained the options for wiping. A DoD option consists of three wipes the first is a full overwrite of 0101 the second pass is 1010 and the last wipe is a pass with all random byte values. The wipe function can also be configured for up to three passes and include any characters the user desires.
5. The DoD option was selected and the file was wiped.
6. Then the floppy was then opened in WinHex to see what the contents included.

The result of this test was not as expected. When the media was selected to wipe, WinHex would open that drive. For the experiment a floppy drive was used. The help option was selected to in an attempt to find the solution to the problem. A solution was not found so the file was wiped. Wiping the file worked as expected, to verify that the file was wiped it was viewed in WinHex before and after the procedure. The results were successful.

The next test was to determine if the timestamps on the files are touched when the file is opened in WinHex.

1. An MD5sum was taken of a picture and a screenshot of the files MAC times was taken to determine the timestamps prior to the file being

© SANS Institute 2005 Author retains full rights.

opened in WinHex.
2. The file was then opened in WinHex.
3. The option to take an MD5sum was selected and compared with the MD5sum of the file before it was opened in WinHex.

The results of this test were not as expected. The modified and created time remained the same however, the accessed time was touched. After some consideration it made since that the access time stamp would be updated to the time that WinHex accessed the information in order to open the file. In order to avoid the timestamp information being lost, the original evidence or the backup used for the investigation should never be used. A copy should be created for the specific use in WinHex. To verify that the accessed time was changed, the file was right clicked on and the properties viewed. This also changes the accessed time stamp but it will not matter for this test. A screen shot was taken and then the file was opened in WinHex. The properties of the file were then viewed in WinHex showing a different accessed time than the first. This is shown in the following images.

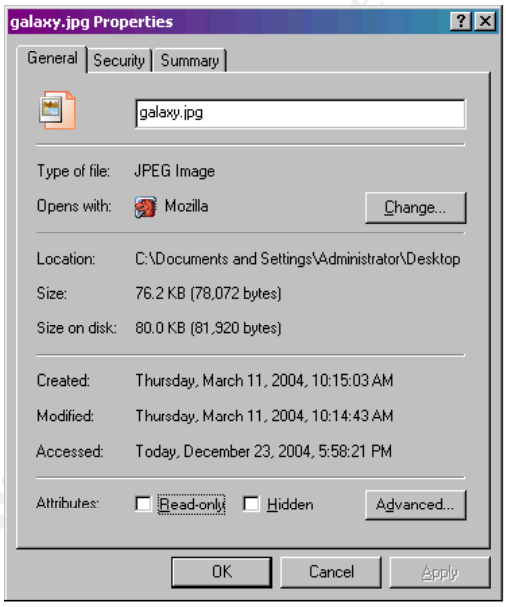


Figure 16 - MAC times file prior to WinHex

© SANS Institute 2005 Author retains full rights.

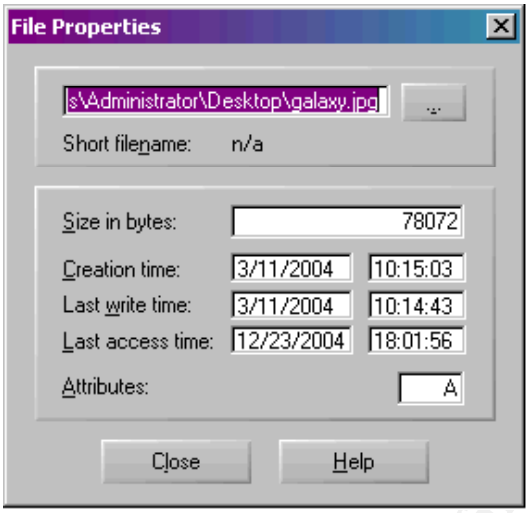


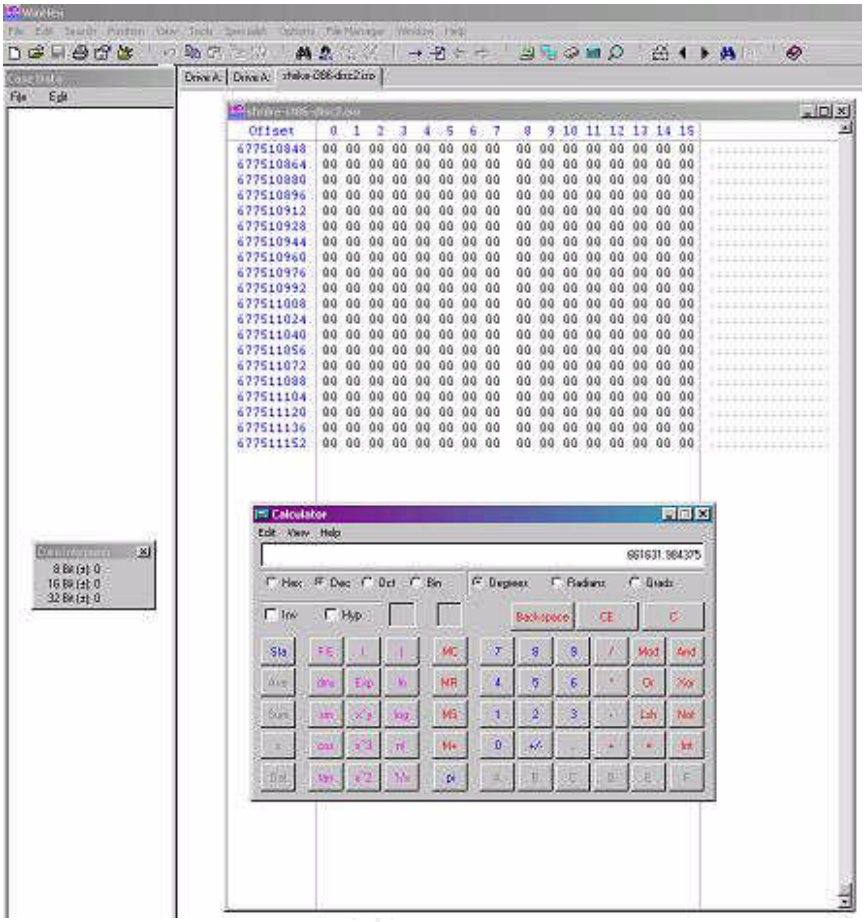
Figure 17 - MAC times of file after WinHex

The final step was to determine how large a file could be opened in WinHex.

1. An ISO image containing a portion of Redhat 9.0 was viewed in windows explorer to determine the size.
2. The ISO was then opened in WinHex.
3. The last offset was viewed. That number was then put into a calculator and divided by 1024. This reveals the size of the file, which verifies the entire image, was opened.

The results of this test did not prove the largest amount that WinHex could open due to a lack of a file large enough to test this. The image that was opened was 661,632 KB. The speed that the image was opened was impressive. The following image shows the last offset of the ISO opened in WinHex. The entire size can be determined by dividing that number by 1024.

© SANS Institute 2005 Author retains full rights.



© SANS Institute 2005

© SANS Institute 2005 Author retains full rights.

Analysis

Most of the information found while testing this tool proved that the designers did a decent job of accurately describing their tool. The only incident that was found was with the wiping utility. This function is not that important to a forensic investigator but could become useful in certain occasions. There could be several explanations for the wiping tool not to work. The first, X-Ways did not portray the functions of their tool correctly or was explained so that it was not clear of the capabilities and was misunderstood. This may also be a function that can only be used by certain licenses.

Another important detail to touch on is the significance of using backups for investigational purposes. It was proved that WinHex touches the access time stamp. However, this information was never found while doing preliminary research of the tool.

Presentation

WinHex does not necessarily have a format to present its output because the tool doesn't output anything. The primary purpose of this tool is a hex editor. It is a way to view what is in a file. The format that it puts the file into could be understood by someone else. The data is converted into hexadecimal and shown along side the contents of the file. If the file contains plain text it would be readable, however, most other file types would look like garbage. The significant information shown is the metadata of the file. The evidence could easily be presented in court. WinHex allows the information shown to be printed out. The format would have to be explained to someone who has never seen a hex editor before. The output of the tool is the exact information in the file or drive. This includes metadata and file signatures. Data that could not normally be seen when the file is opened can be seen using WinHex. For example a text document in Microsoft Word, there is substantial amount of information tagged onto the document, including file name and information about the computer it was opened on. Using WinHex this information can be seen.

© SANS Institute 2005 Author retains full rights.

Conclusion

The tests that were conducted on WinHex attempted to prove that this tool is important to a forensic investigator. The tests also tried to prove the many different capabilities this one tool holds. I have concluded that this tool could be used by a forensic investigator. As states early a backup will need to be made, in order for the access timestamps to remain the untouched. Improvements to this tool could not be found through the tests that were performed. The forensic license of WinHex may have many more capabilities that could be used by a forensic investigator.

© SANS Institute 2005, Author retains full rights.

© SANS Institute 2005 Author retains full rights.

References

Guillermito, (easily) Breaking a (very weak) steganography software: Camouflage. 16 Sept. 2002. 5 Nov. 2004 <http://guillermito2.net/stegano/camouflage/index.html>

Raggo, Michael T., Steganography, Steganalysis, & Cryptanalysis. 2004 24 Nov. 2004. <http://www.blackhat.com/presentations/bh-usa-04/bh-us-04-raggo/bh-us-04-raggo-up.pdf>

Kruse, Warren G and Heiser, Jay G. Computer Forensics Incident Response Essentials. Indianapolis: Pearson. 2003.

© SANS Institute 2005, Author retains full rights.

© SANS Institute 2005 Author retains full rights.