GIAC Certified Forensic Analyst (GCFA)

PRACTICAL ASSIGNMENT

Version 1.3

"Whenever you have excluded the impossible, whatever remains, however improbable, must be the truth."[1]

PART 1 ~ Analyze an Unknown Binary

PART 2 ~ Option 2: Perform Forensic Tool Validation

PART 3 ~ Legal Issues of Incident Handling

Bil Bingham
December 5, 2003

[1] Sir Author Conan Doyle - (The Adventures of Sherlock Holmes - "The Adventures of the Beryl Coronet")

Table of Contents

TABLE OF CONTENTS **1**

ABSTRACT **2**

PART 1 ~ ANALYZE AN UNKNOWN BINARY **3**

1.1 INTRODUCTION 3
1.2 FINDINGS 4
1.3 DETAILED ANALYSIS 7
1.4 LEGAL IMPLICATIONS 23
1.5 INTERVIEW QUESTIONS 24
1.6 ADDITIONAL INFORMATION 25

PART 2 ~ OPTION 2: PERFORM FORENSIC TOOL VALIDATION **27**

2.1 INTRODUCTION 27
2.2 SCOPE 27
2.3 TOOL DESCRIPTION 28
2.4 TEST APPARATUS AND ENVIRONMENTAL CONDITIONS 31
2.5 DESCRIPTION OF THE PROCEDURES 33
2.6 CRITERIA FOR APPROVAL 40
2.7 DATA AND RESULTS 41
2.8 ANALYSIS 47
2.9 PRESENTATION 48
2.10 CONCLUSION 49
2.11 ADDITIONAL INFORMATION 49

PART 3 ~ LEGAL ISSUES OF INCIDENT HANDLING **50**

3.1 INTRODUCTION 50
3.2 THE LAW 51
3.3 ADDITIONAL INFORMATION 54

APPENDIX **55**

A.1 NETWORK CONNECTION CODE 55
A.2 VISION-TEST.VBS 59

Abstract

This document has been prepared as an assignment to qualify as a GIAC Certified Forensics Analyst. The document has three parts that are direct responses to the GIAC practical assignment for Forensic Analysts.

Part 1 is a systematic account of an analysis of an unknown binary. This section describes the process of discovering information about executable code through a variety of processes and software tools.

In part 2, a test is formulated and applied to prove that a new software tool has what it takes to be a part of a forensic analyst's toolbox. Read on to find out if a Foundstone™[2] product called Vision™ v1.0 makes the grade.

Part 3 focuses on the Canadian legal system and electronic forensics. This part will answer a series of questions designed to inform the reader about Canadian law and forensics covering issues such as privacy and electronic evidence.

[2] For more information about Foundstone visit their web site at http://www.foundstone.com/

Part 1 ~ Analyze an Unknown Binary

1.1 Introduction

Global Information Assurance Certification[3] (GIAC) has provided the file target2.exe in a compressed format (binary v1.3.zip) for a forensic analysis. GIAC also provided instructions to analyze the binary file to determine the program's capabilities, purpose, what it was used for and why it was on the system in question. The uncompressed file (target2.exe) is the target of this analysis and no analysis of on the compressed file (binary v1.3.zip) is included with this report.

The binary analysis techniques discussed during the SANS[4] conference in Portland entitled Track 8: System Forensics, Investigation and Response are put into practice for this analysis.

[3] For more information about GIAC visit http://www.giac.org/
[4] SANS (SysAdmin, Auditing, Network, and Security) is a research and education organization. For more information about SANS visit http://www.sans.org/

1.2 Findings

This section of the document presents the findings of the binary analysis for quick reference. The evidence and analysis used to gather this information is discussed in great detail in the next section of this document entitled "Detailed Analysis"

A. Binary Details

The table below shows a summary of the binary details gathered from the first step of the detailed analysis.

Name	target2.exe
Size	26793 bytes
Md5 Checksum	848903a92843895f3ba7fb77f02f9bf1
Last Modified	February 20, 12:45 PM (PST) 2003
File Classification	MS-DOS executable (EXE), OS/2 or MS Windows

Figure 1 - Table of Binary Details

Without access to the original file system, it is not possible to determine the last time this binary was executed.

B. Program Description

The systematic analysis identified several features of target2.exe:

1. The original filename of the file is smsses.exe.

2. This program runs as a service win32 operating systems (Windows NT, Windows 2000, and Windows XP). A service is a process that does not require a user to log in. The UNIX equivalent to a service would be a daemon, which runs in the background without user intervention required to start.

3. The binary contains installing and un-installing subroutines activated by command line parameters.

4. This binary may act as an ICMP[5] backdoor and allow a remote client to talk to this service in a hidden tunnel. This hidden communication has the ability to pass undetected through some network security devices (firewalls, access lists).

[5] ICMP (Internet Control Message Protocol) is defined by RFC 792. It provides a messaging system for IP modules to communicate in certain conditions. For more information visit (http://www.faqs.org/rfcs/rfc792.html)

C. Forensic Details

The forensic details of this binary presented here are in two categories (footprints and leads). The footprints are the unique set of features on the host system that can positively identify that this binary is installed or running. The leads are pieces of information that require further analysis, investigation or research such as a computer and or person's name.

This binary leaves a distinct set of footprints when installed onto a host system. The installation process of this binary creates a windows service called "Local Print Manager". The discrepancy between "Local Print Manager" and Local Partners Access" is the name of the service in the registry is "Local Partners Access" and the name of the service in the services control is "Local Print Manager". The name of the service in the services control is the value of the registry key "HKLM\System\CurrentControlSet\Services\Local Partners Access\DisplayName".

The following registry keys are present on a host with this application installed:

```
HKLM\System\CurrentControlSet\Services\Local Partners Access
HKLM\System\CurrentControlSet\Services\Local Partners Access\Type
HKLM\System\CurrentControlSet\Services\Local Partners Access\Start
HKLM\System\CurrentControlSet\Services\Local Partners Access\ErrorControl
HKLM\System\CurrentControlSet\Services\Local Partners Access\ImagePath
HKLM\System\CurrentControlSet\Services\Local Partners Access\DisplayName
HKLM\System\CurrentControlSet\Services\Local Partners Access\Security\Security
HKLM\System\CurrentControlSet\Services\Local Partners Access\ObjectName
```

When running, the process name that appears in the task list will be smsses.exe.

The forensic analysis provides several hints that provide some more detail on the nature ad the origin of this program. The following strings provide more leads to track down the nature of the program.

a) IP Address 199.107.97.191

The Azusa Pacific University owns this IP address. The host name for this IP address is sbm191.dtc.apu.edu. A quick search on the University's web site reveals that "sbm" is short for School of Business and Management.

b) "Code by Spoof"

The analysis uncovered the programmer's alias is Spoof. An initial Internet search (http://www.google.ca/) on this name yields approximately eight hundred thousand hits. Trying to narrow down the search by using the name in context of the text found did not yield any further information.

c) “Hello from MFC!”

MFC typically stands for Microsoft Foundation Class. In the context that this “MFC” appears, “MFC” is an abbreviation for a name. The third person viewpoint further suggests that it is an abbreviation of a group of people. The Internet search engine, Google™, does not find any relevant information about this text string.

D. Program Identification

Searching the Internet uncovered some similarities with this binary and source code for ICMP backdoors, however no exact match was located for the source code of this binary.

The two pieces of code that appear to be a reference or starting point for this binary are HDoor V0.1[6] and icmp_tunnel.h[7].

[6] The source code for HDoor V0.1 by Lion was found at http://www.cnhonker.net/Files/show.php?id=189
[7] The source code for icmp_tunnel.h by Dark Schneider was found at http://www.s0ftpj.org/bfi/online/bfi7/bfi07-13.html

1.3 Detailed Analysis

This section of the document is a systematic account of the methods, techniques, and actions taken during the course of this analysis. The analysis is broken down into five stages (preparation, binary details, code analysis, run-time analysis, and research). Each stage of the analysis will unearth more information and may effect the actions and/or steps taken in the next stage.

A. Preparation

The preparation stage is required not to analyze the file or gather any information, but to reduce any risks involved with examining a possibly malicious executable file.

The workstations used to directly analyze this file do not have any connections to the production networks to avoid any risk from the unknown binary. For this analysis, VMWare™ sessions containing various operating systems running on a Microsoft™ Windows 2000 Professional host workstation are in place. Also in the toolkit is the Forensic and Incident Response Environment (F.I.R.E.)[8] cd-rom, which contains a host of forensic tools. The F.I.R.E. cd-rom runs from a laptop in the same private network as the forensic workstation. The two workstations (plus the virtual VMWare™ hosts) are on a private network connected to a Cisco™ Pix 501 firewall. In one part of the analysis, a separate Windows 2000 Workstation[1] is used to overcome some icmp reply issues encountered with VMWare™.

[8] F.I.R.E. is a free Forensic and Incident Response Environment on a bootable CD-ROM created by William Salusky. For more information visit http://fire.dmzs.com/

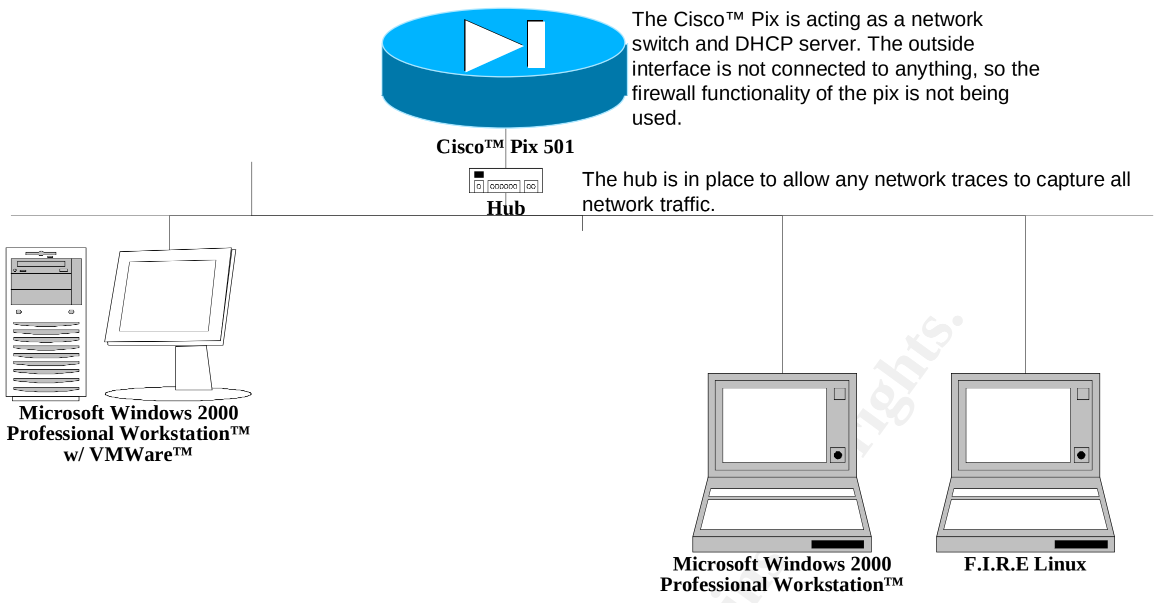
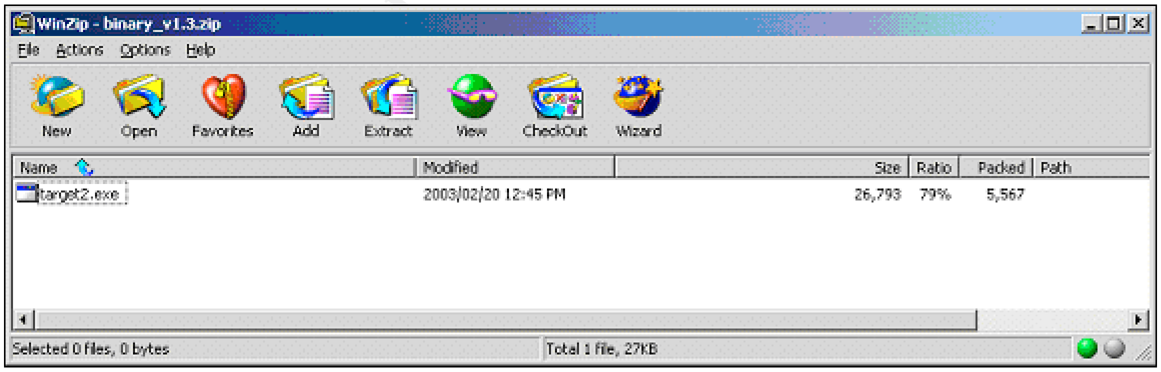

The Cisco™ Pix is acting as a network switch and DHCP server. The outside interface is not connected to anything, so the firewall functionality of the pix is not being used.
Cisco™ Pix 501
The hub is in place to allow any network traces to capture all network traffic.
Hub
Microsoft Windows 2000 Professional Workstation™ w/ VMWare™
Microsoft Windows 2000 Professional Workstation™
F.I.R.E Linux

Figure 2 - Forensic Laboratory Configuration

B. Binary Details

In the binary detail analysis stage of this analysis, facts and details about the binary are collected without opening, or executing the file. This detail will analyze properties of the file such as date/time stamps for access, modification and creation. File ownership will also appear at this stage of the analysis.

WinZip™ (8.0) on a Windows 2000 Professional workstation shows the file name, size and last modified date (see graphic).


WinZip - binary_v1.3.zip
File Actions Options Help
New Open Favorites Add Extract View CheckOut Wizard

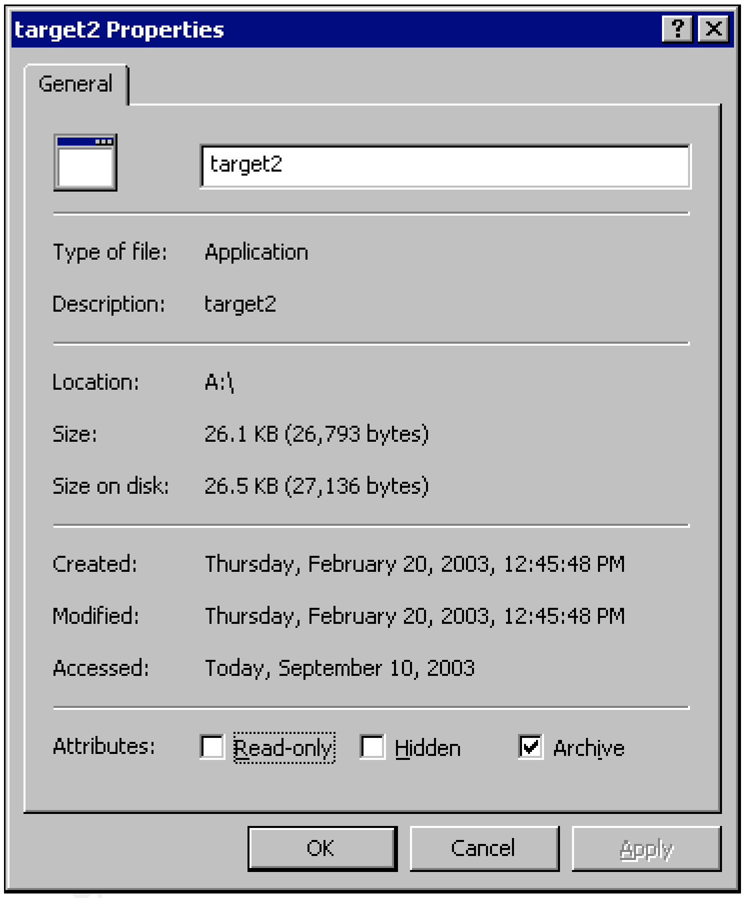
Name	Modified	Size	Ratio	Packed	Path
target2.exe	2003/02/20 12:45 PM	26,793	79%	5,567	

Selected 0 files, 0 bytes   Total 1 file, 27KB

Figure 3 – WinZip™ 8.0 Properties

In NT File Systems (the file system on Windows 2000, Windows NT and Windows XP), file owner information is contained in the file allocation tables on the hard drive not with the file itself. Without access to the original hard drive this file resided on, owner information will not be available.

After file extraction (decompressing) to a floppy drive, displaying the file properties via the operating system (Windows 2000 Professional) verifies the file properties. It also shows the creation date/time and last accessed date/time. Decompressing the file modifies the last access timestamp of the file and is therefore little value to this analysis. The creation date/time is the same as the last modified date/time.



target2

Type of file: Application
Description: target2

Location: A:\
Size: 26.1 KB (26,793 bytes)
Size on disk: 26.5 KB (27,136 bytes)

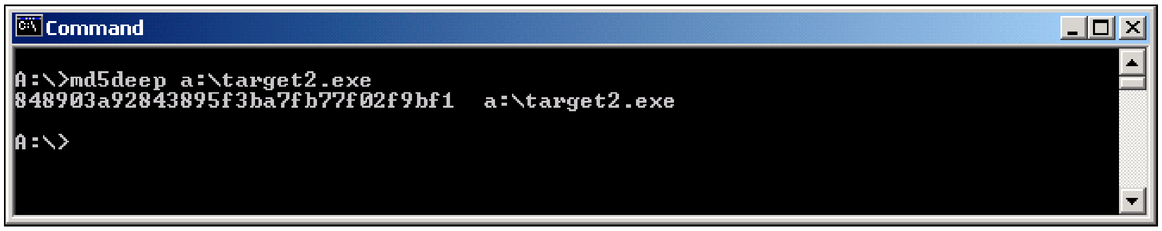
Created: Thursday, February 20, 2003, 12:45:48 PM
Modified: Thursday, February 20, 2003, 12:45:48 PM
Accessed: Today, September 10, 2003

Attributes: ☐ Read-only ☐ Hidden ☑ Archive

Figure 4 - Windows File Properties

Md5Deep[9] calculates a md5[10] message digest on the file. The md5 message digest provides a unique identifier for this file. If any of the file contents change or are tampered with the md5 message digest will change. Calculating the md5 digest is important to prove the evidence's integrity. As the md5 message digest is unique to this file, it proves that the file contents do not change past this point in time. It is compared with itself and any other files found in the course of this analysis to prove if the files are identical. The md5 message digest of this file is *848903a92843895f3ba7fb77f02f9bf1.*

[9] Special Agent Jesse Kornblum of the US Air Force Office of Special Investigations wrote Md5deep. For more information on Md5Deep visit http://md5deep.sourceforge.net
[10] Ronald Rivest defines the md5 algorithm in RFC 1321. For more information about the md5 algorithm visit http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc1321.html



```
A:\>md5deep a:\target2.exe
848903a92843895f3ba7fb77f02f9bf1  a:\target2.exe

A:\>
```

Figure 5 - MD5Deep Message Digest

On most UNIX operating systems, a command called "**file**" exists to help identify files by their contents. The ***file*** command run from a Red Hat™ distribution of Linux VMWare™ session identifies target2.exe as:

target2.exe: MS-DOS executable (EXE), OS/2 or MS Windows

C. Code Analysis

The code analysis will examine and analyze the contents of the file in detail. The code analysis deploys two tools to extract information out of the file. The first program named **strings** extracts ASCII[11] text out of the target file. ACSII codes represent English number and letters in the form of hexadecimal numbers in the binary file. The second tool is a disassembler that will try to reverse engineer the binary file into assembly code, which is easier to analyze. Assembly code is a programming language that is one better than machine language (one's and zero's) that allows for the use of names.

The utility **strings** will extract all ASCII phrases out of target2.exe that are four characters or longer. The output of **strings** gives us a good knowledge of what some of the functions and subroutines called by this binary are. However, the disassembler in the next step of code analysis are much more accurate and detailed in extracting any dependencies. In terms of a program, dependencies are any other files the program requires to execute.

The most interesting evidence to further identify this binary are the text lines:

```
impossibile creare raw ICMP socket
RAW ICMP SendTo:
======================== Icmp BackDoor V0.1
========================
========= Code by Spoof. Enjoy Yourself!
 Your PassWord:
loki
cmd.exe
 Exit OK!
```

These extracted lines give us a good indication of this executable's possible use. Spoof appears to be an alias of the programmer that developed this binary. The references to the icmp raw and socket support the header icmp backdoor. In addition, the reference to loki contains a big hint. As well as being

[11] ASCII is the acronym for Acronym for the American Standard Code for Information Interchange

the Greek god of mischief, loki is the codename for a published icmp backdoor program. Phrack Magazine[12] published several articles including source code to implement loki.

Given a service running on a system such as target2.exe or loki, icmp backdoors can the capability to send commands and files directly to the host machine.

One of the most frightening aspects of icmp backdoors is the potential for bypassing tradition security devices. Although technically icmp is a layer 4 (transport) protocol like TCP and UDP, it is normally classified as layer 3 (network) of the OSI[13] model. This is because it is typically not used as a transport, but for network error messages and connectivity. Due to this nature of icmp, most firewalls do not inspect the payload of icmp and only inspect the packet header. This means that computer hosts compromised with an icmp backdoor may be remotely accessed even behind a firewall.

The other interesting output is the reference to starting and stopping services giving us another clue about the capabilities of target2.exe.

Other strings that require some investigation also show up. “Rich” appears however is embedded in the first 64 bits of the program meaning it is part of the MS-DOS stub. The MS-DOS stub is the part of all portable executable (PE) files that displays “This program cannot be run in DOS mode”. This particular string resides in all PE files followed by “.text”.

After strings, another tool called a dissembler dissects the program into various parts and collects debug information. A demo version of PE Explorer 1.93[14] suited this purpose. PE Explorer is a disassembler specifically for PE files. Microsoft™ designed the portable executable (PE) file format for 32 bit Windows executable files.

PE Explorer gives us the following information to analyze about target2.exe:

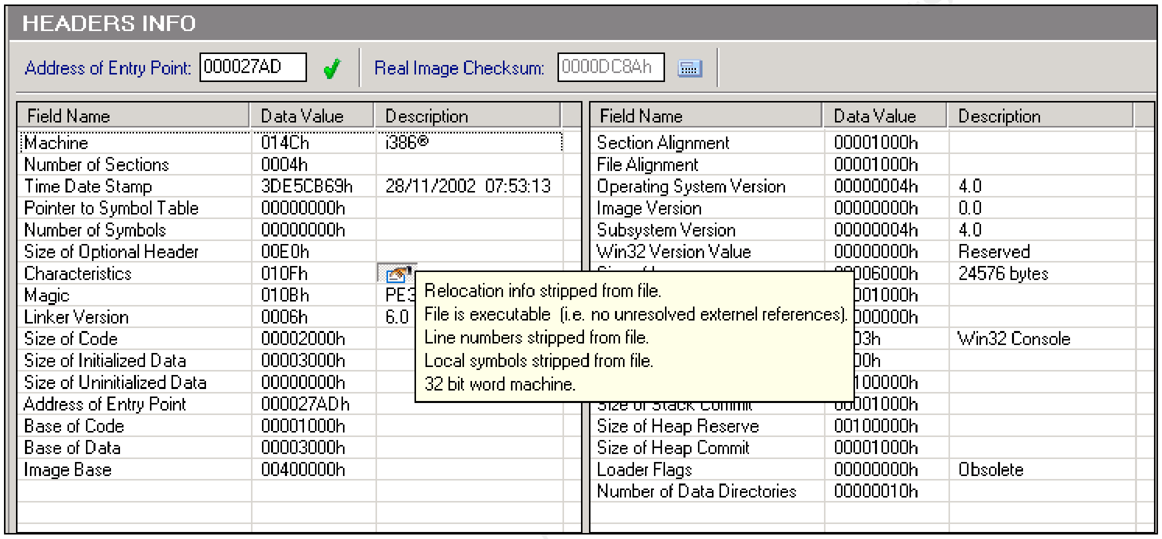
a) Header - Header information found in the file header of target2.exe
b) Names - A list of names of functions and subroutines
c) Resources - Strings table extracted from the resource section of the target2.exe
d) Dependencies - List of external linked libraries required to execute target2.exe
e) Assembly Code - The raw assembly code instructions

[12] http://www.phrack.org/show.php?p=49&a=6 and http://www.phrack.org/show.php?p=51&a=6 are the two articles from Phrack Magazine that discuss and implement icmp backdoors.
[13] OSI is short for Open System Interconnection, a standard for worldwide communication. A chart detailing the seven layers of the OSI model can be found at http://webopedia.internet.com/quick_ref/OSI_Layers.asp
[14] PE Explorer is a product of Heaven Tools software. Their homepage is http://www.heaventools.com/

f) End of File Data - Data found after the end of the program.
g) Strings - Another source of text extracted from target2.exe.

a) Header Information

In a portable executable (PE) file format like all executable formats, the header information contains a collection of fields telling the executing system important information required to run the program.


HEADERS INFO

Address of Entry Point: 000027AD   Real Image Checksum: 0000DC8Ah

Field Name	Data Value	Description
Machine	014Ch	i386®
Number of Sections	0004h	
Time Date Stamp	3DE5CB69h	28/11/2002 07:53:13
Pointer to Symbol Table	00000000h	
Number of Symbols	00000000h	
Size of Optional Header	00E0h	
Characteristics	010Fh	
Magic	010Bh	PE3
Linker Version	0006h	6.0
Size of Code	00002000h	
Size of Initialized Data	00003000h	
Size of Uninitialized Data	00000000h	
Address of Entry Point	000027ADh	
Base of Code	00001000h	
Base of Data	00003000h	
Image Base	00400000h	

Field Name	Data Value	Description
Section Alignment	00001000h	
File Alignment	00001000h	
Operating System Version	00000004h	4.0
Image Version	00000000h	0.0
Subsystem Version	00000004h	4.0
Win32 Version Value	00000000h	Reserved
[illegible]	00006000h	24576 bytes
[illegible]	001000h	
[illegible]	000000h	
[illegible]	03h	Win32 Console
[illegible]	00h	
[illegible]	100000h	
Size of Stack Commit	00001000h	
Size of Heap Reserve	00100000h	
Size of Heap Commit	00001000h	
Loader Flags	00000000h	Obsolete
Number of Data Directories	00000010h	

Relocation info stripped from file.
File is executable (i.e. no unresolved externel references).
Line numbers stripped from file.
Local symbols stripped from file.
32 bit word machine.

Figure 6 - PE Explorer Headers Information

The date/time stamp displayed in the above graphic is the time that this file was compiled (according to the date/time of the compiling computer). In this case, the date/time of compilation is several months off from the last modified date of the file system (Feb 20th 2003).

The decompiler also shows that normal debugging information is not present. The line number and local symbols have been stripped from the file. This will make further analysis harder as all the decompiled assembly language instructions will have memory addresses as names instead of symbolic names.

Note that the information in the version fields of the header is often in error. Several linkers either leave the field blank or enter a wrong value. By the linker version of 6.0 and image base is consistent with program that are compiled by Microsoft C++ 6.0 but is not conclusive.

b) Names

Names in the PE executable file are symbols that refer to specific sections of code. In this binary, all the symbolic names have been removed, and we are left with memory address.

```
00404094: L00404094
00404098: SSZ00404098_RAW_ICMP_SendTo__
004040AC: SSZ004040AC___________________________Icmp_
00404130: SSZ00404130_loki
00404138: L00404138
```

As shown in the above excerpt from the name output, some of the text referenced in the “data” portion of the executable is linked to the memory location where they are called. During the debugging analysis, this information will be necessary to help pinpoint which lines of code are producing a specific output.

c) Resources

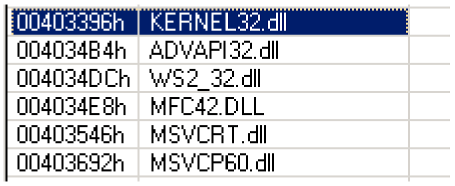
This is information stored in the resources section of the PE file. PE Explorer gives us an approximation of what was in the original “.rc” file.

```
STRINGTABLE
LANGUAGE LANG_CHINESE, SUBLANG_CHINESE_SIMPLIFIED
{
1, "Hello from MFC!"
}
```

The context of this indicates that MFC is an alias or abbreviation of person and/or group. MFC is the common short form of Microsoft Foundation Classes. The second clue that this output reveals is the language used. The “Lang_Chinese” setting may provide some insight to the origins of this program.

d) Dependencies

This list displays all the dynamic linked libraries (dll) that this program calls during execution. Dynamic linked libraries are files that contain functions and data for other programs (executables) to call upon and reference.



00403396h	KERNEL32.dll
004034B4h	ADVAPI32.dll
004034DCh	WS2_32.dll
004034E8h	MFC42.DLL
00403546h	MSVCRT.dll
00403692h	MSVCP60.dll

Figure 7 - Target2.exe imports

Target2.exe links five dll files directly. These direct imports in turn require more dll files to function.

Kernel32 provides an interface into the system to access memory, processes, and resources. Advapi32 provides access to security and the registry. Ws2_32 contains libraries for network connections. Mfc42, Msvcrt, and Msvcp60 are all standard C libraries. Msvcp60 is associated and distributed with most Microsoft visual c++ applications.

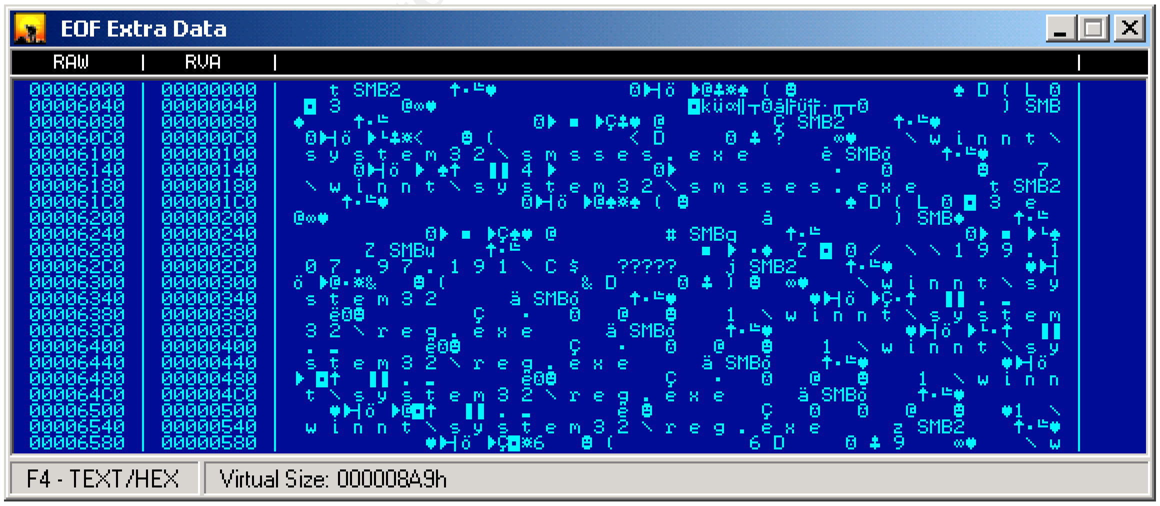
Analyzing the subroutines used within these dll files give a very clear picture of what this program is capable of doing. The ones that seem have the most potential for malicious behaviors are the disk read/write (kernel32), network access (ws2_32), and service management (advapi32). The service management capabilities of advapi32 include start, stop, create and delete.

e) Raw Code

The raw assembly code is a list of all the assembly code processed by the computer as this binary executes. This information will be useful when debugging the program and trying to correlate specific actions with source code and variables.

f) End of File (EOF) Raw Data

After the End of File Marker and before the physical end of the file on disk some interesting leads appear.



```
RAW       | RVA       |
00006000  | 00000000  |   t SMB2    ↑·╚♥          0▶┤ö ▶@♣※♠ ( ☻            ♠ D ( L 0
00006040  | 00000040  |  ◘ 3      @∞♥                 ◘kü∞║┬0ä╟üï·╥┬0         ) SMB
00006080  | 00000080  | ♦    ↑·╚          0▶ ■ ▶Ç♣♥ @        Ç SMB2    ↑·╚♥
000060C0  | 000000C0  |  0▶┤ö ▶└♣※<    ☻ (        < D    0 ♣ ?     ∞♥    \ w i n n t \
00006100  | 00000100  |  s y s t e m 3 2 \ s m s s e s . e x e      è SMBö    ↑·╚♥
00006140  | 00000140  |     0▶┤ö ▶ ♠↑  ▌▌ 4 ▶        0▶          .    0        ☻     7
00006180  | 00000180  |  \ w i n n t \ s y s t e m 3 2 \ s m s s e s . e x e      t SMB2
000061C0  | 000001C0  |    ↑·╚♥              0▶┤ö ▶@♠※♠ ( ☻           ♠ D ( L 0 ◘ 3  e
00006200  | 00000200  | @∞♥                                 á          ) SMB♦     ↑·╚
00006240  | 00000240  |               0▶ ■ ▶Ç♠♥ @        # SMBq     ↑·╚          0▶ ■ ▶└♠
00006280  | 00000280  |      Z SMBw     ↑·╚                 ■ ▶ ·♦  Z ◘ 0 ⁄  \ \ 1 9 9 . 1
000062C0  | 000002C0  |  0 7 . 9 7 . 1 9 1 \ C $   ?????    j SMB2     ↑·╚♥              ♥┤
00006300  | 00000300  | ö ▶@·※&    ☻ (           & D     0 ♣ ) ☻   ∞♥     \ w i n n t \ s y
00006340  | 00000340  |  s t e m 3 2       ä SMBö     ↑·╚♥           ♥┤ö ▶Ç·↑  ▌▌ . ▬
00006380  | 00000380  |   ë0☻        Ç   .    0    @    ☻      1   \ w i n n t \ s y s t e m
000063C0  | 000003C0  |  3 2 \ r e g . e x e       ä SMBö     ↑·╚♥              ♥┤ö ▶└·↑  ▌▌
00006400  | 00000400  |  . ▬            ë0☻        Ç   .    0    @    ☻      1   \ w i n n t \ s y
00006440  | 00000440  |  s t e m 3 2 \ r e g . e x e       ä SMBö     ↑·╚♥             ♥┤ö
00006480  | 00000480  | ▶ ◘↑   ▌▌ . ▬          ë0☻        Ç   .    0    @    ☻      1   \ w i n n
000064C0  | 000004C0  |  t \ s y s t e m 3 2 \ r e g . e x e       ä SMBö     ↑·╚♥
00006500  | 00000500  |    ♥┤ö ▶@◘↑   ▌▌ . ▬         ë ☻        Ç   0    0    @    ☻    ♥1   \
00006540  | 00000540  |  w i n n t \ s y s t e m 3 2 \ r e g . e x e       z SMB2      ↑·╚♥
00006580  | 00000580  |               ♥┤ö ▶Ç◘※6    ☻ (          6 D     0 ♣ 9      ∞♥     \ w
```

F4 - TEXT/HEX | Virtual Size: 000008A9h

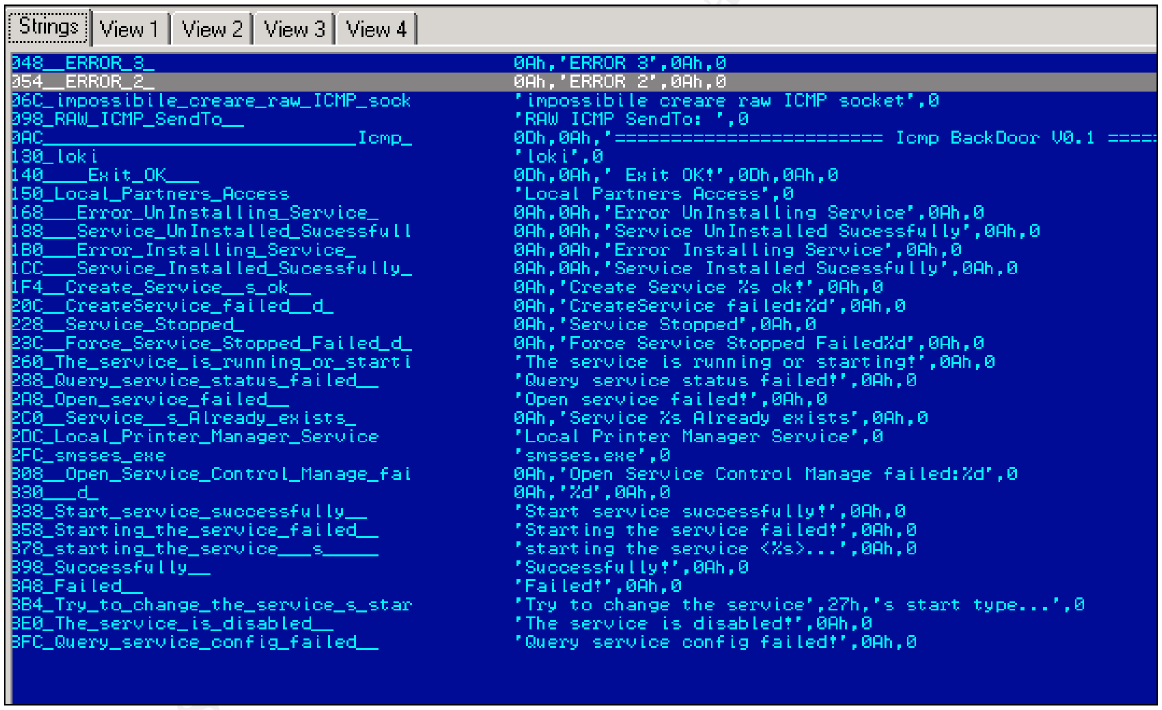
Figure 8 - PE Explorer EOF Raw Data

Some file names and an IP address show up separated by a non-ASCII character (they do not show up from the strings command). These characters could have been extracted with a **strings** utility using a binary instead of am ASCII format.

```
\\199.107.97.191\C$
\Winnt\system32\smsses.exe
\\winnt\system32\reg.exe
```

g) Strings

The disassembler pulls this text out the data portion of the file. These strings are different from the command **strings** utility in that PE Explorer separates these out from specified memory locations instead of searching for text. Some text that appears to be program switches "-i" and "-d" show up. These are not found by the **strings** utility as they are only two ASCII characters long and the default for strings is four.



```
048__ERROR_3_                              0Ah,'ERROR 3',0Ah,0
054__ERROR_2_                              0Ah,'ERROR 2',0Ah,0
06C_impossibile_creare_raw_ICMP_sock       'impossibile creare raw ICMP socket',0
098_RAW_ICMP_SendTo__                      'RAW ICMP SendTo: ',0
0AC_____________________________Icmp_      0Dh,0Ah,'======================== Icmp BackDoor V0.1 ====
130_loki                                   'loki',0
140____Exit_OK___                          0Dh,0Ah,' Exit OK!',0Dh,0Ah,0
150_Local_Partners_Access                  'Local Partners Access',0
168___Error_UnInstalling_Service_          0Ah,0Ah,'Error UnInstalling Service',0Ah,0
188___Service_UnInstalled_Sucessfull       0Ah,0Ah,'Service UnInstalled Sucessfully',0Ah,0
1B0___Error_Installing_Service_            0Ah,0Ah,'Error Installing Service',0Ah,0
1CC___Service_Installed_Sucessfully_       0Ah,0Ah,'Service Installed Sucessfully',0Ah,0
1F4__Create_Service__s_ok__                0Ah,'Create Service %s ok!',0Ah,0
20C__CreateService_failed__d_              0Ah,'CreateService failed:%d',0Ah,0
228__Service_Stopped_                      0Ah,'Service Stopped',0Ah,0
23C__Force_Service_Stopped_Failed_d_       0Ah,'Force Service Stopped Failed%d',0Ah,0
260_The_service_is_running_or_starti       'The service is running or starting!',0Ah,0
288_Query_service_status_failed__          'Query service status failed!',0Ah,0
2A8_Open_service_failed__                  'Open service failed!',0Ah,0
2C0__Service__s_Already_exists_            0Ah,'Service %s Already exists',0Ah,0
2DC_Local_Printer_Manager_Service          'Local Printer Manager Service',0
2FC_smsses_exe                             'smsses.exe',0
308__Open_Service_Control_Manage_fai       0Ah,'Open Service Control Manage failed:%d',0
330___d_                                   0Ah,'%d',0Ah,0
338_Start_service_successfully__           'Start service successfully!',0Ah,0
358_Starting_the_service_failed__          'Starting the service failed!',0Ah,0
378_starting_the_service___s_____          'starting the service <%s>...',0Ah,0
398_Successfully__                         'Successfully!',0Ah,0
3A8_Failed__                               'Failed!',0Ah,0
3B4_Try_to_change_the_service_s_star       'Try to change the service',27h,'s start type...',0
3E0_The_service_is_disabled__              'The service is disabled!',0Ah,0
3FC_Query_service_config_failed__          'Query service config failed!',0Ah,0
```

Figure 9 - PE Explorer String

D. Run-Time Analysis

At the run-time analysis stage, several trusted programs monitor the activity of a lab system while target2.exe runs. The output of these trusted programs determine what the target2.exe program is doing to the host. Another program called a debugger steps through target2.exe binary code line by line and can provide memory states at any step of target2.exe's execution.

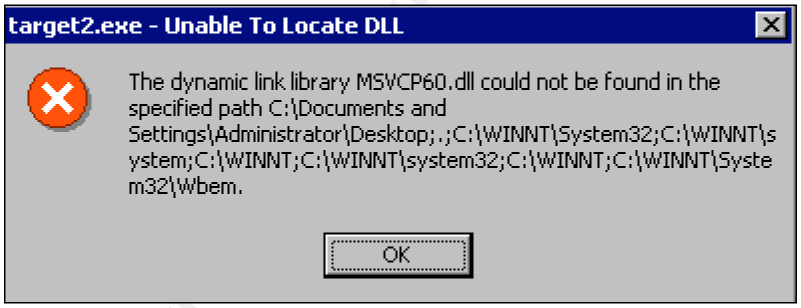
Five programs monitor the state of the test system. The first is regmon[15]. This program monitors the windows registry and provides a list of all transaction to the registry. The Second is filemon[16]. This program is similar to regmon except instead of the registry it monitors changes to the file system.

The third tool used to track system changes is winalysis[17]. This program snapshots the workstation and saves the data to compare against another snapshot (taken after the binary is executed). The version used in this analysis is a trail version downloaded from Winalysis Software^{TM}'s website.

Netstat run with the switch "–an" displays a list of all the tcp and udp sockets that are open. This is a list of all communications on the system. **Netstat** is a utility that comes with Windows operating systems.

The last program deployed is Ethereal[18]. Ethereal is a network sniffer that monitors the network card and captures any network packets coming from or to the test system.

Running target2.exe the first time proves to be a very disappointing exercise. At first, an error about a missing dll file appears.

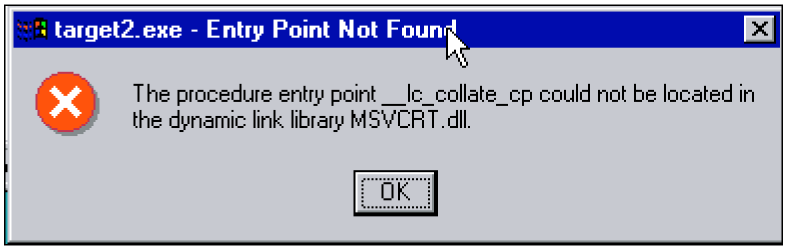

target2.exe - Unable To Locate DLL

```
The dynamic link library MSVCP60.dll could not be found in the
specified path C:\Documents and
Settings\Administrator\Desktop;.;C:\WINNT\System32;C:\WINNT\s
ystem;C:\WINNT;C:\WINNT\system32;C:\WINNT;C:\WINNT\Syste
m32\Wbem.
```

Figure 10 - target2.exe dll error

Most visual C++ applications (including Microsoft Office) install msvcp60.dll. Windows XP platforms have msvcp60.dll installed by default. The three tools did not detect any chances to the host system even after the missing dll problem resolution. Filemon logs indicate that the dependency dll files referenced by this program are accessed, but no system changes took place. The same run-time tests applied to Windows XP also show no system changes. The run-time test on Windows NT produced an error.

[15] Copyright © 1996-2003 Mark Russinovich and Bryce Cogswell
[16] Copyright © 1996-2003 Mark Russinovich and Bryce Cogswell
[17] Winalysis is produced by Winalysis Software Inc. The software is available at http://www.winalysis.com/
[18] Gerald Combs is the original author of the open source program Ethereal. The many contributors to the program can be found at http://www.ethereal.com/introduction.html - authors


target2.exe - Entry Point Not Found

The procedure entry point __lc_collate_cp could not be located in the dynamic link library MSVCRT.dll.

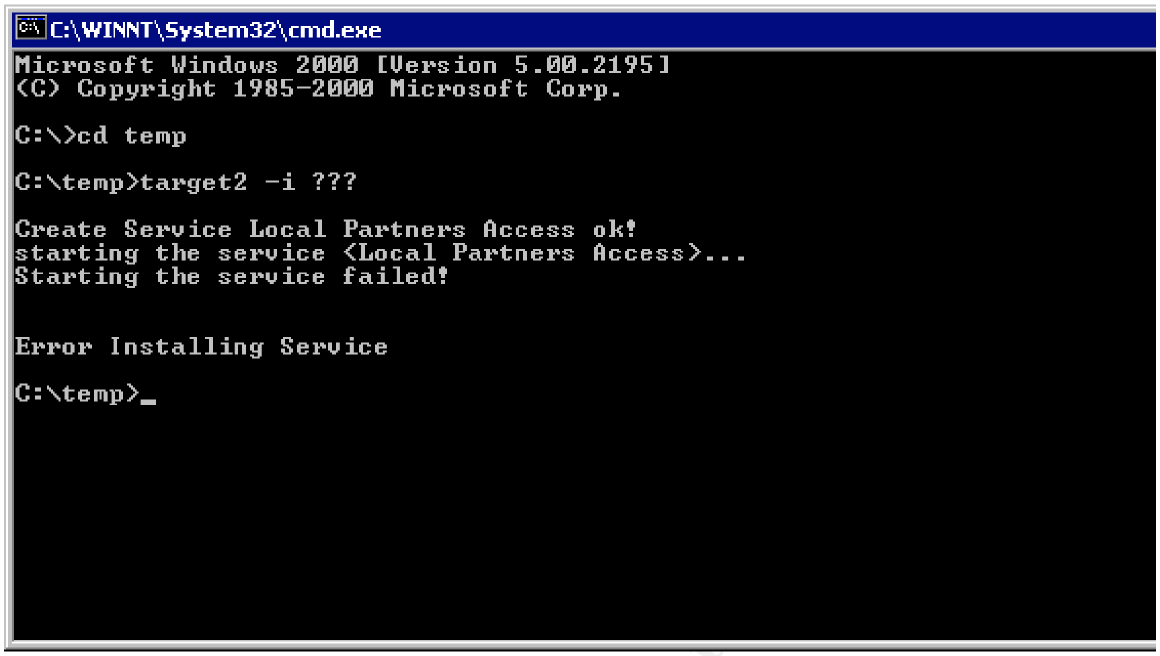
Figure 11 - target2.exe errors in WinNT

In all these tests, target2.exe completes and returns control back to the command shell indicating it has finished. Regmon and Winalysis logs show that the executable reads files but makes no changes to the host system. Ethereal shows no relevant network traffic.

Setting up the run-time tests again with the command line switches "-i" and "-d" yields some results. Through trial and error, it was determined that this combination of two switches initiates the binary.

a) Target2.exe –i anything

This combination of switches installs a new service onto the host machine called "Local Print Manager". The service is created via the advapi32.ddl!createservice subroutine. The service (although this executable tries to start it) fails, as the source binary it is trying to start does not exist (smsses.exe). Regmon, filemon and winalysis all provide an inventory of changes to the system as the binary installs a service. An error appears on the screen, as target2.exe reports that the service failed to start.



```
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\>cd temp

C:\temp>target2 -i ???

Create Service Local Partners Access ok!
starting the service (Local Partners Access)...
Starting the service failed!


Error Installing Service

C:\temp>_
```

Figure 12 target2.exe -i ???

b) Target2.exe –d anything

This combination of switches removes the service "Local Print Manager" if it exists. Otherwise this combination reports an error.

The content of the second switch does not seem to matter only that it is present. Any more than two switches also results in no binary activity.

Renaming target2.exe to smsses.exe and placing it in the path c:\winnt\system32\ produce a running service. When smsses.exe is executed with the parameters –i ??? a new service is created and starts successfully.

The file registry and winalysis results indicate show this executable running, creating, and then starting the service. A second process (smsses.exe) starts up and stays resident in memory as a running service.

The only system changes made running this executable with the "-i" switch are registry changes associated with creating a new service. The process smsses.exe stays resident in memory and continues to run as a service. Ethereal and Netstat both indicate no change or exchange of information on the network.

To debug the binary and try to determine the nature of the second variable and extract any more information about the nature of this executable, a debugger called w32dasm[19] was used. The debugging activity tries to find clues about the nature of smsses.exe were not very successful.

[19] Win32Dasm by URSoft®. The trial software was obtained from http://www.downseek.com/. The official home page of URSoft (http://www.expage.com/w32dasm/) appears to be abandoned at the time of this writing.

Kernel32!Writefile Call 4D1F2A, tracing results in limited success as it is found that no calls past the starting point refer to this section of code.

Trying to determine the significance of the second parameter in the ASM, debugging module only finds the conditional statements that

1) Check parameter 1 against "-i" at relative memory location 4041F0
2) Check for the number of parameters 402888 mov eax; dword ptr (esp+4)
3) Conditional Check to see if the service "Local Print Manager" exists 40233B

The debugging tests are very limited as the scope of the program running at the command prompt (as opposed to a service) is limited to installing and de-installing the service.

To conclude the run-time tests, another test is formulated to attempt to produce a reaction of smsses.exe. A Linux (FIRE Workstation) is used to generate icmp packets with the –p option trying to prove and induce a response from the smsses.exe service. Ethereal immediately picks up some icmp response packet changes. Icmp reply data should be identical to the original icmp request. After testing it is discovered that the changes in the response data are a result of VMWare TCP/IP stack misbehaving and not the binary smsses.exe. After a new workstation is acquired (without VMMare), ICMP reacts as expected and no response or hidden data flow can be detected via Ethereal.

The next logical step in this testing process would be to try and debug the running service and watch the code as input in the form of ICMP input is received. By further research into the nature of windows 32-bit winsock, it is revealed that not all ICMP traffic is passed to the raw socket, and this includes ICMP echo requests that are automatically answered by the TCP/IP driver stack. To revalidate the previous ping tests a packet generator with the capability to form icmp echo replies (which are passed onto to the raw socket) would be required.

By the definition of icmp there are also many possible ways to pass data across it with client/server software such as the date information in a different type of icmp packet, so these test do not indicate that this is not an icmp backdoor.

E. Research

Now all the information is gathered, it is time to follow upon some of the leads and hints provided by the technical portion of this forensic analysis.

a) //199.107.97.191/C$

The IP address can be tracked down as it is an Internet routable IP address. ARIN[20] provides an online utility called **whois** to lookup information about IP addresses. The output of a **whois** search on 199.107.97.191 returns the following information:

```
Search results for: 199.107.97.191
CERFnet NETBLK-CERFNET-CBLK2 (NET-199-105-0-0-1)
199.105.0.0 - 199.108.255.255
CERFnet customer - Azusa Pacific University CERF-AZUSA (NET-199-107-96-0-1)
199.107.96.0 - 199.107.99.255
# ARIN WHOIS database, last updated 2003-06-03 21:05
# Enter ? for additional hints on searching ARIN's WHOIS database.
Search results for: ! NET-199-107-96-0-1
OrgName:    CERFnet customer - Azusa Pacific University
OrgID:      CCAPU-1
Address:    901 E. Alosta Ave.
City:       Azusa
StateProv:  CA
PostalCode: 91702
Country:    US
NetRange:   199.107.96.0 - 199.107.99.255
CIDR:       199.107.96.0/22
NetName:    CERF-AZUSA
NetHandle:  NET-199-107-96-0-1
Parent:     NET-199-105-0-0-1
NetType:    Reassigned
Comment:
RegDate:    1996-08-09
Updated:    1997-10-11
TechHandle: CERF-HM-ARIN
TechName:   AT&T Enhanced Network Services
TechPhone:  +1-858-812-5000
TechEmail:  notify@attens.com
# ARIN WHOIS database, last updated 2003-06-03 21:05
# Enter ? for additional hints on searching ARIN's WHOIS database.
```

The IP address (199.109.97.191) is registered to Azusa Pacific University, and using **nslookup**, the DNS[21] name of this IP address resolves to sbm191.dtc.apu.edu. Nslookup is a utility that The website of the Azusa Pacific University is www.apu.edu. A quick search on their web site reveals that sbm is short for School of Business and Management.

b) /winnt/system32/smsses.exe

This is the file name of the unknown binary that is the focus of this analysis. In the end of file space where this name appears, the word SMB kept appearing as well. SMB stands for System Message Block and is a protocol used for transferring files.

[20] American Registry for Internet Numbers (ARIN) is the authority that assigns IP address. Their web site is http://www.arin.net/
[21] DNS or Domain Name System translates names into IP addresses and vice versa.

c) /winnt/system32/reg.exe

Looking up this file name on Google™ shows up that reg.exe is part of the windows resource kit. This file is a utility to read and update the windows registry.

d) ”Code by Spoof”

The search engine Google™ had far too many hits on this search string as spoof is a very common term. Nothing in context to this analysis appeared.

e) “Hello from MFC!”

Again, many returns on the term MFC but not in the context of a group or as an absolute string were found. MFC commonly stands for Microsoft Foundation Class.

The next step is to determine and try to find a source of this program on the Internet. Various strings, unique identifiers, and descriptions are put into a search engine to try to find a relevant match. Two source codes have some similar characteristics as target2.exe and they may be a reference or a basis for this code. The words that found hints at the source code are “ping backdoor win32” and “immposible to creare raw socket”

a) HDoor[22]

Here is the HDoor v0.1 code, as it would appear without the formatting tags.

```
“====================== Ping BackDoor V0.1 ======================”
“========= Code by Lion. “
” Your PassWord:"
```

Here is the strings extracted from target2.exe:

```
“====================== Icmp BackDoor V0.1 ======================”
“========= Code by Spoof. Enjoy Yourself!”
“ Your PassWord:”
```

Notice how the number of equal signs for both text descriptions is equal in both banners. It is almost as if it was a cut and paste and change the name, and change ping to icmp. Even the space before the “Your password: “ is identical. This is circumstantial evidence however the fact that two suspected icmp backdoor programs use the same banner appears to be more than reasonable doubt.

[22] The source code for HDoor V0.1 by Lion was found at http://www.cnhonker.net/Files/show.php?id=189

b) ICMP_tunnel.h[23]

Again, the similarities are only a small part of the code. “RAW ICMP SendTo:” and "impossibile creare raw ICMP socket" from the bulletin board source code match strings output of target2.exe. These strings are very circumstantial and the only reason I included them in this analysis is that the two executables contain the same two strings in different languages.

Both of these found sources are dissimilar from the binary in that they do not contain any service installing / de-installing code.

[23] The source code for icmp_tunnel.h by Dark Schneider was found at http://www.s0ftpj.org/bfi/online/bfi7/bfi07-13.html

1.4 Legal Implications

No proof was available to determine that this executable had been run on the system, as access to the original system was not available to compare the system state with the footprint of this binary, however should such proof come available, and it could be proven that access to said computer system was fraudulent or unauthorized. Then under Canadian Criminal Code 342.1(1) any offender can be sentenced to a term of up to ten years. Here is the excerpt from the Canadian Criminal Code:

"Every one who, fraudulently and without colour of right,

(a) obtains, directly or indirectly, any computer service,

(b) by means of an electro-magnetic, acoustic, mechanical or other device, intercepts or causes to be intercepted, directly or indirectly, any function of a computer system,

(c) uses or causes to be used, directly or indirectly, a computer system with intent to commit an offence under paragraph (a) or (b) or an offence under section 430 in relation to data or a computer system, or

(d) uses, possesses, traffics in or permits another person to have access to a computer password that would enable a person to commit an offence under paragraph (a), (b) or (c)

is guilty of an indictable offence and liable to imprisonment for a term not exceeding ten years, or is guilty of an offence punishable on summary conviction."

If this executable was found installed and running on a corporate computer, the person(s) responsible would be in direct violation with my corporation's information security policy, section titled configuration control.

1.5 Interview Questions

These interview questions should determine if the interviewee is responsible for the binary analyzed:

- Do/Did you attend the university of Azusa Pacific?
- Did you install and/or execute this program?
- What does MFC stand for?
- Who is Spoof?
- What does the executable smsses.exe do?
- Did you install the “Local Printer Manager” service?
- Are you authorized by the company to use this computer?
- What is your username?
- When did you have access to this computer?

1.6 Additional Information

Reference

Loki in Norse Mythology
http://www.pantheon.org/articles/l/loki.html

Peering Inside the PE: A Tour of the Win32 Portable Executable File Format
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndebug/html/msdn_peeringpe.asp
by Matt Pietrek

An In-Depth Look into the Win32 Portable Executable File Format, Part 2
http://msdn.microsoft.com/msdnmag/issues/02/03/PE2/default.aspx
by Matt Pietrek

Alien Autopsy: Reverse Engineering Win32 Trojans on Linux
http://www.lurhq.com/alien.pdf
by Joe Stewart, GCIH

Covert Shells
http://gray-world.net/papers/covertshells.txt
by J. Christian Smith

Project Loki: ICMP Tunneling
http://www.phrack.org/show.php?p=49&a=06
by Alhambra and daemon9

LOKI2 (the implementation)
http://www.phrack.org/show.php?p=51&a=6
by daemon9

Inside NT Utilities - Regmon
http://www.winnetmag.com/Articles/Index.cfm?ArticleID=4795&pg=3
by Mark Russinovich

Section 342.1 Criminal Code of Canada - Unauthorized Use of a Computer System
http://www.catalaw.com/logic/docs/ds-3421.htm
by Daniel Shap

Department of Justice Canada – Criminal Code
http://laws.justice.gc.ca/en/C-46/41491.html

TCP/IP Illustrated, Volume 1
The Protocols
By W. Richard Stephens

Just What is SMB ?
http://samba.anu.edu.au/cifs/docs/what-is-smb.html
by Richard Sharpe

Internet Utilities

http://www.google.ca/	Internet search engine
http://www.webopedia.com/	Online dictionary
http://www.liutilities.com/	Database of dynamic linked libraries

Software

http://www.winalsyis.com	Winalysis
http://www.heaventools.com	PE Explorer
http://fire.dmzs.com/	F.I.R.E. boot cd-rom
http://www.geocities.com/govest/	GoVest Debugger
http://www.expage.com/w32dasm/	Win32Dasm Debugger
http://www.sysinternals.com	RegMon and FileMon
http://md5deep.sourceforge.net	MD5Deep
http://www.microsoft.com	Windows 2000, Windows XP, Windows NT, Windows 98
http://www.redhat.com	Red Hat's Distibution of Linux
http://www.winzip.com	WinZIp
http://www.vmware.com	VMWare

Part 2 ~ Option 2: Perform Forensic Tool Validation

2.1 Introduction

During this part of the CGFA assignment, a series of tests confirm whether a tool is viable as a forensic tool. The tests will validate the tool as reliable, accurate and what impact it has on a system.

The first line of testing revolves around the program footprint. The results help reduce and eliminate the original footprint. The second test validates the accuracy and reliability of the tool.

2.2 Scope

To communicate effectively on a TCP/IP network, an application or process needs a layer four protocol (transport) to effectively send or receive network data. The most commonly used protocols for this are Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). These protocols are defined in RFC 1700[24] as protocol 2 and protocol 17.

Each of these protocols can be used on a variety of ports from 1 to 16555. The tests in this part will focus on measuring the accuracy and reliability of a program that lists these open port and protocol pairs, and then associates them with a running process.

For the purposes of this validation, the scope of tests will be limited to Windows 2000™.

[24] A copy of this RFC can be found at http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc1700.html

2.3 Tool Description

A. Description

The product tested is Vision™ v1.0 from Foundstone[25]. Vision™ is a free downloadable software package obtained from http://www.foundstone.com/. Once at Foundstone's home page click resources and then free tools, then forensics. Now click Vision™ v1.0 to start a download.

This software program's main function is to list and associate TCP and UDP network connections with running processes. In addition, the software allows the user to send the socket a string of information and kill the socket. This software lists running applications, processes, services, device drivers, and basic system information. The output of the log file is the list of open network connections and the owning application. The list of running processes mapped against network ports provides an excellent source of evidence to prove that an executable is running. Vision™ v1.0's output combined with a network sniff provides can prove beyond a reasonable doubt exactly what transpired on a given system.

B. Footprint

The same three tools that determined the foot of the unknown binary will determine the footprint of this forensic tool. These tools are regmon, filemon and Winalysis™.

Using those three tools to footprint the install process results in one hundred and thirty seven thousand three hundreds seventy-three file accesses and ten thousand two hundred and two registry accesses. The install process called **visionsetup.exe** has a very large impact on the target system.

To try and decrease this footprint, several steps are taken

1) Gather a list of all the installed files as determined by the output of filemon and winalysis.

```
C:\Program Files\Foundstone\Vision\DelsL1.isu
C:\Program Files\Foundstone\Vision\License.txt
C:\Program Files\Foundstone\Vision\Readme.txt
C:\Program Files\Foundstone\Vision\Vision.cnt
C:\Program Files\Foundstone\Vision\Vision.exe
C:\Program Files\Foundstone\Vision\VISION.HLP
C:\Program Files\Foundstone\Vision\_DEISREG.ISR
C:\Program Files\Foundstone\Vision\_ISREG32.DLL
```

[25] Copyright © 1999 - 2003, Foundstone, Inc. All rights reserved worldwide.

2) Copy the identified files to a network share
3) Reset the VMWare™ session to prior to Vision™ install
4) Copy the files back to the host
5) Try to launch Vision™

The program launches successfully without the install portion from the off-line files. The file and registry access counts are now five hundred and twenty six and seven hundred and two to run the program.

To try to reduce these further, all the dynamic linked library files that **vision.exe** requires are relocated into the off-line vision directory. The following dll files are copied into the **C:\Program Files\Foundstone\Vision\** directory:

```
C:\WINNT\System32\ACTIVEDS.DLL        C:\WINNT\System32\rnr20.dll
C:\WINNT\System32\ADSLDPC.DLL         C:\WINNT\System32\RTUTILS.DLL
C:\WINNT\System32\DHCPCSVC.DLL        C:\WINNT\System32\SAMLIB.DLL
C:\WINNT\System32\DNSAPI.dll          C:\WINNT\System32\Secur32.dll
C:\WINNT\System32\ICMP.DLL            C:\WINNT\System32\SETUPAPI.DLL
C:\WINNT\System32\inetmib1.dll        C:\WINNT\System32\snmpapi.dll
C:\WINNT\System32\iphlpapi.dll        C:\WINNT\System32\TAPI32.DLL
C:\WINNT\System32\MPRAPI.DLL          C:\WINNT\System32\USERENV.DLL
C:\WINNT\System32\NETAPI32.DLL        C:\WINNT\System32\winrnr.dll
C:\WINNT\System32\NETRAP.dll          C:\WINNT\System32\WINSPOOL.DRV
C:\WINNT\System32\PSAPI.DLL           C:\WINNT\System32\WS2_32.dll
C:\WINNT\System32\RASAPI32.DLL        C:\WINNT\System32\WS2HELP.DLL
C:\WINNT\System32\RASMAN.DLL          C:\WINNT\System32\WSOCK32.dll
```

Now that the file access has been minimized, all the vision files and dll files that are in the off-line folder are moved to a cd-rom. We are down to one hundred and eighty-eight file accesses. Of these accesses the only files accessed by vision.exe directly are:

```
C:\WINNT\System32\drivers\etc\services     C:\WINNT\System32\winrnr.dll
C:\WINNT\System32\rnr20.dll
```

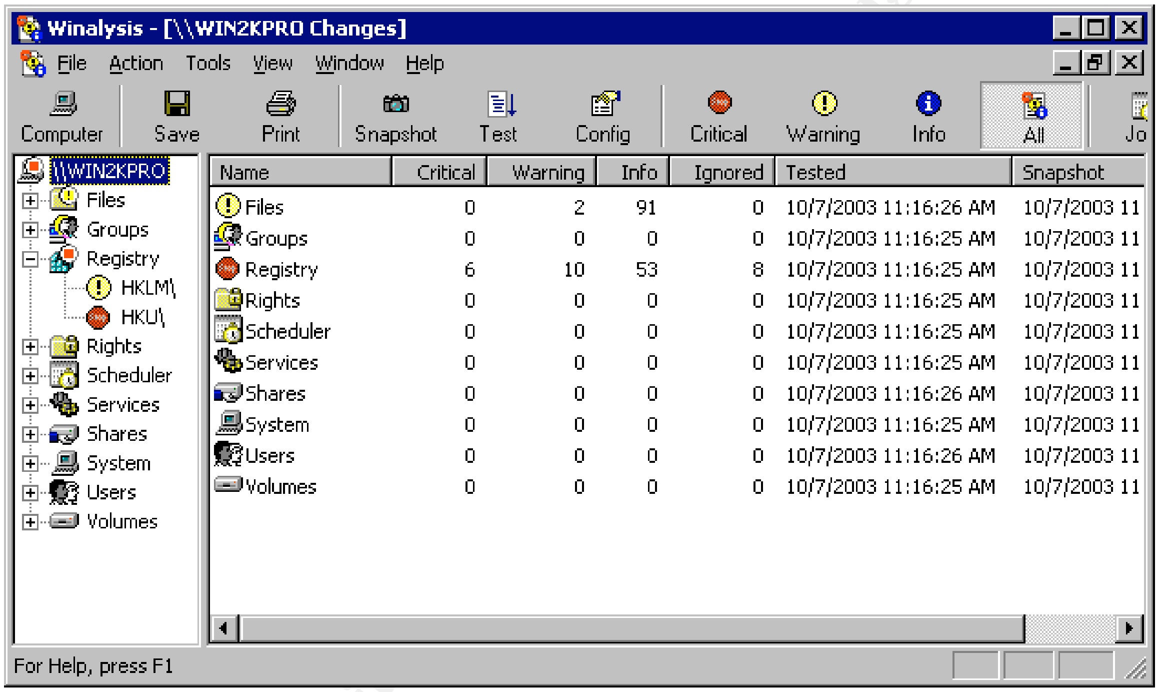
Two of the files that we moved into the application path (winrnr.dll and rnr20.dll) are still accessed by **vision.exe**. The files indirectly accessed by running vision.exe are:

```
C:\Documents and Settings\Administrator\Application Data\Microsoft\Internet Explorer
C:\Documents and Settings\Administrator\Application Data\Microsoft\Internet Explorer\Quick Launch
C:\Documents and Settings\Administrator\ntuser.dat.LOG
C:\PROGRA~1\COMMON~1\MICROS~1\WEBFOL~1\MSONSEXT.DLL
C:\Documents and Settings\Administrator\NTUSER.DAT
C:\Documents and Settings\Administrator\ntuser.dat.LOG
C:\Program Files\                         C:\WINNT\AppPatch\sysmain.sdb
C:\Program Files\desktop.ini              C:\WINNT\system32\cabinet.dll
C:\WINNT\AppPatch\LayerStorage.dat        C:\WINNT\system32\drivers\etc\services
C:\$LogFile                               C:\WINNT\system32\sdbapiu.dll
C:\pagefile.sys                           C:\WINNT\system32\sfc.dll
C:\Program Files\desktop.ini              C:\WINNT\win.ini
C:\WINNT\AppPatch\msimain.sdb             C:\
```

The following registry keys are created by running vision™ the first time on a system:

```
HKLM\SOFTWARE\Microsoft\RFC1156Agent\CurrentVersion
HKLM\SOFTWARE\Microsoft\RFC1156Agent\CurrentVersion\Parameters
HKLM\SOFTWARE\Microsoft\RFC1156Agent\CurrentVersion\Parameters\TrapPollTimeMilliSecs
HKLM\SOFTWARE\Classes\TypeLib\{1EA4DBF0-3C3B-11CF-810C-00AA00389B71}\1.1\0\win32
HKLM\SOFTWARE\Microsoft                          HKLM\SOFTWARE\Microsoft\RFC1156Agent
```

The two file warnings listed in the Winalysis™ graphic directly relate to saving regmon and filemon results to disk. Some of the registry access listed in regmon output is a direct result of the winalysis tool. Several registry keys are created and depending how the program is launched, the "most recently used" lists are modified.



Name	Critical	Warning	Info	Ignored	Tested	Snapshot
Files	0	2	91	0	10/7/2003 11:16:26 AM	10/7/2003 11
Groups	0	0	0	0	10/7/2003 11:16:25 AM	10/7/2003 11
Registry	6	10	53	8	10/7/2003 11:16:25 AM	10/7/2003 11
Rights	0	0	0	0	10/7/2003 11:16:25 AM	10/7/2003 11
Scheduler	0	0	0	0	10/7/2003 11:16:25 AM	10/7/2003 11
Services	0	0	0	0	10/7/2003 11:16:25 AM	10/7/2003 11
Shares	0	0	0	0	10/7/2003 11:16:26 AM	10/7/2003 11
System	0	0	0	0	10/7/2003 11:16:25 AM	10/7/2003 11
Users	0	0	0	0	10/7/2003 11:16:26 AM	10/7/2003 11
Volumes	0	0	0	0	10/7/2003 11:16:25 AM	10/7/2003 11

Figure 13 - Winalysis Detected System Changes.

The footprint has now been minimized and calculated, running Vision™ changes the access time of 23 files, creates 6 registry keys, and modifies other registry key depending on how Vision™ was launched.

2.4 Test Apparatus and Environmental Conditions

Two hosts connected to a private isolated network generate test tcp and udp connections for Vision™ to record. These hosts are connected to a private isolated network.

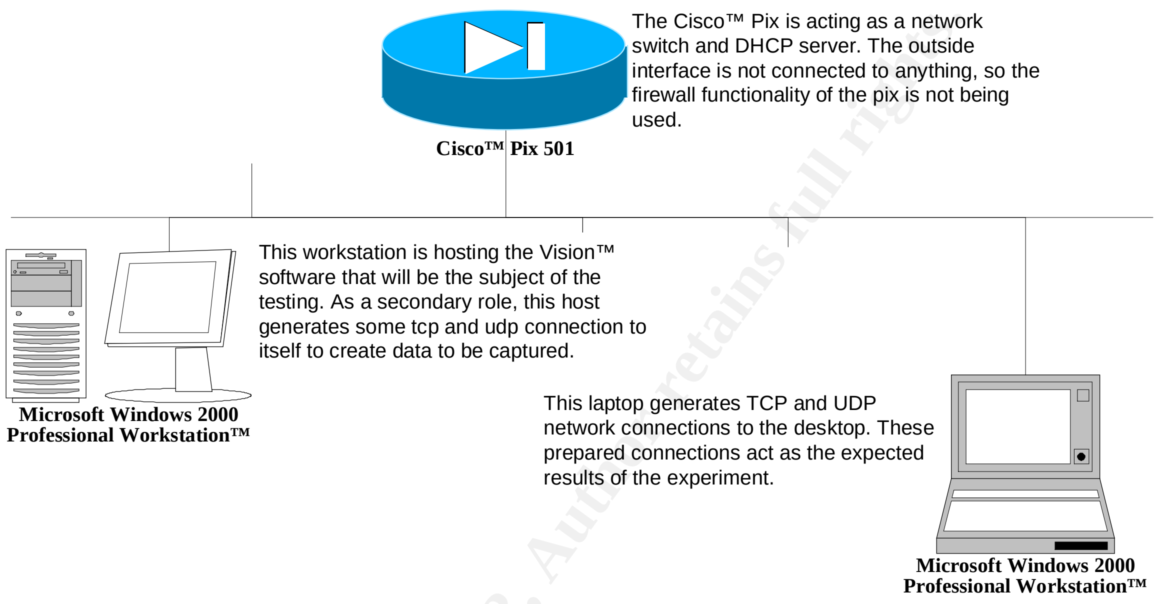

The Cisco™ Pix is acting as a network switch and DHCP server. The outside interface is not connected to anything, so the firewall functionality of the pix is not being used.
Cisco™ Pix 501
This workstation is hosting the Vision™ software that will be the subject of the testing. As a secondary role, this host generates some tcp and udp connection to itself to create data to be captured.
Microsoft Windows 2000 Professional Workstation™
This laptop generates TCP and UDP network connections to the desktop. These prepared connections act as the expected results of the experiment.
Microsoft Windows 2000 Professional Workstation™

Figure 14 - Experiment Environment

For clarity, the two hosts on this test network will be called Vision™ Host and test laptop for the rest of this document. Both test platforms are running Microsoft Windows 2000 Professional Workstation™, Version 5.00.2195 Service Pack 4.

The windows hosts in the experiment may generate NetBIOS traffic. Windows networking is known to be chatty. The broadcasts and internal windows 2000 NetBIOS traffic may interfere with the results. Initially a VMWare™ session was used as the tasting platform, however due to the shared aspect (with the host) of the networking layers, this option was not used to avoid skewing the test results.

FPORT is another utility from Foundstone™. It is the command line predecessor of Vision™ and will be used for two purposes. One is to check the accuracy of Vision™ and the second is to ensure the test environment does not change.

Netstat is a command that is bundled with Windows 2000 Professional Workstation™. This command list the UDP and TCP ports open without displaying applications that correspond with the ports.

TGrab™ v1.4 is a screen capture program that will be used to capture the data from vision. Vision does have a log file, however the log file output is restricted to active connections. TGrab™ is available from http://www.doldersum.com/. For this tool validation a trail copy of TGrab™ is used.

2.5 Description of the Procedures

This chapter of the document describes in detail the testing procedures used in this experiment. The output of these procedures will be analyzed to determine the feasibility of using Vision™ as a forensic tool.

The test will be performed as follows:

1) Launch Vision
2) Get Vision results from screen with TGrab™. The screenshot will have to be renamed as TGrab™ will overwrite it the next time it is launched.
3) Get Vision results from output log. The file will have to be renamed, as Vision will append any new information to the same file. To retain the integrity of the data and associate it with a particular test. The log file will be renamed every time Vision™ is executed.
4) Get Fport output
5) Get Netstat –an output
6) Close Vision™

The test will be repeated 1005 times to determine the accuracy and consistency of the data output. Once complete, md5sum will calculate a message digest on all the files. The message digests will be the basis for the analysis and any variations indicate a change in the test environment or inconsistent results of Vision™.

A. Setup

Two windows 2000 hosts will create a static networking environment by utilizing C# (c-sharp) scripts to generate UDP and TCP connections.

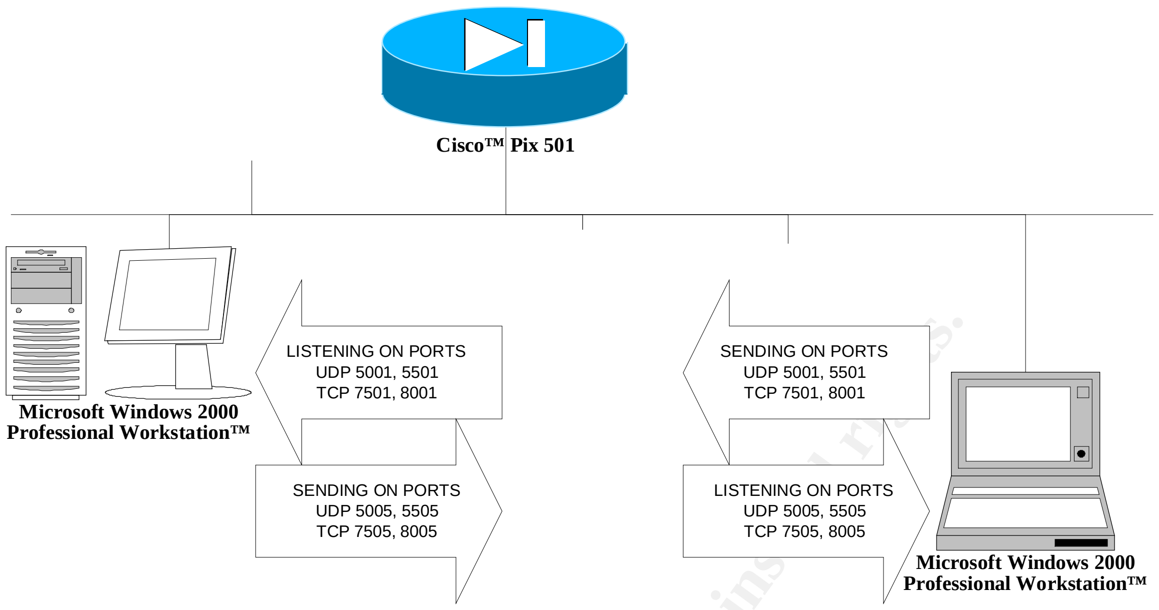

Cisco™ Pix 501
Microsoft Windows 2000 Professional Workstation™
LISTENING ON PORTS UDP 5001, 5501 TCP 7501, 8001
SENDING ON PORTS UDP 5005, 5505 TCP 7505, 8005
SENDING ON PORTS UDP 5001, 5501 TCP 7501, 8001
LISTENING ON PORTS UDP 5005, 5505 TCP 7505, 8005
Microsoft Windows 2000 Professional Workstation™

Figure 15 - Network Connections

The script source codes are displayed in the appendix. These scripts have been compiled for each port for ease of identification in the log files. Had the c# script had the port number as a parameter instead of being hard coded it might not be apparent which application was which.

Originally, the test was designed with 10 ports for each TCP and UDP (20 in total), however two issues arose out of that. The first was that 20 ports would not fit onto one screen of data so the screenshot would not have all the data. Even with four ports, the screen resolution had to be increased to capture all the GUI data in one screenshot. The other issue was system performance. With 20 applications tossing raw data around in an endless loop, the test platforms were not very responsive.

Here is a table of which application is sending or receiving on the appropriate port.

Host	Application Name	Action	Protocol	Port
Test Laptop	TX_UDP_5001	Send	UDP	5001
Test Laptop	TX_UDP_5501	Send	UDP	5501
Test Laptop	TX_TCP_7501	Send	TCP	7501
Test Laptop	TX_TCP_8001	Send	TCP	8001
Test Laptop	RX_UDP_5005	Receive	UDP	5005
Test Laptop	RX_UDP_5505	Receive	UDP	5505
Test Laptop	RX_TCP_7505	Receive	TCP	7505
Test Laptop	RX_TCP_8005	Receive	TCP	8005
Vision™ Host	TX_UDP_5005	Send	UDP	5005
Vision™ Host	TX_UDP_5505	Send	UDP	5505

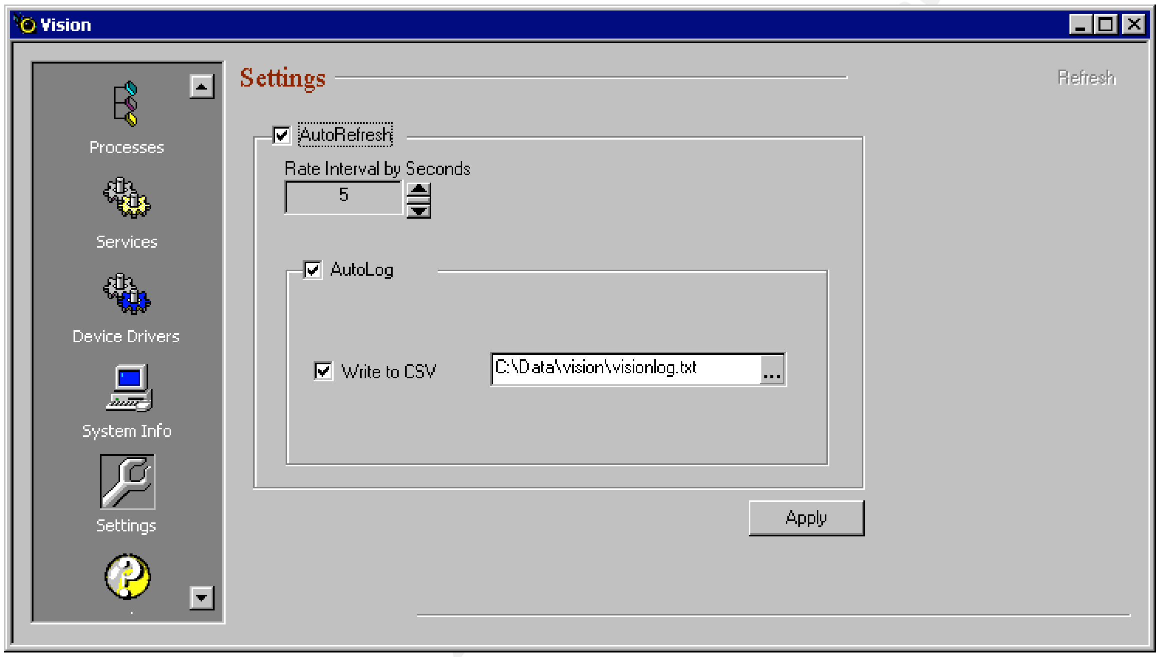
Vision™ Host	TX_TCP_7505	Send	TCP	7505
Vision™ Host	TX_TCP_8005	Send	TCP	8005
Vision™ Host	RX_UDP_5001	Receive	UDP	5001
Vision™ Host	RX_UDP_5501	Receive	UDP	5501
Vision™ Host	RX_TCP_7501	Receive	TCP	7501
Vision™ Host	RX_TCP_8001	Receive	TCP	8001

To conduct the procedure as outlined, a visual basic script was written to facilitate doing the test 1005 times. The vbs code listed below launches TGrab™, Vision™, Fport, and Netstat. The script relies on the setup on TGrab™ to get an automatic screenshot. After a delay, the script renames the log files and terminates vision. Once it has completed this cycle 1005 times, it then calculates an md5 message digest on all the files in the four data directories. The output of md5sum is placed into a text file in each directory called md5sum.txt.

```
StopAt=1005
NETSTATFolder="C:\DATA\NETSTAT"
FPORTFolder="C:\DATA\FPORT"
TGRABFolder="C:\DATA\SCREENSHOTS"
VISIONFolder="C:\DATA\VISION"
VISIONLog ="C:\DATA\VISION\Visionlog.txt"
Set wshshell = createobject("wscript.shell")
Set objWMIService = GetObject("winmgmts:"  & "{impersonationLevel=impersonate}!\\.\root\cimv2")
Done = False
i=1
Do While Not(Done )
        wshshell.run "C:\Progra~1\TGrab\TGrab.exe"
        Wscript.sleep 2000
        wshshell.SendKeys "~"
        Wscript.sleep 1000
        wshshell.SendKeys "%b"
        wshshell.run "C:\Progra~1\Foundstone\Vision\Vision.exe"
        Wscript.sleep 1000
        Wshshell.SendKeys "%"
        Wshshell.SendKeys " "
        Wshshell.SendKeys "x"
        Wshshell.run "cmd.exe /c fport > c:\data\fport\fport_" & i & ".txt",7,FALSE
        Wshshell.run "cmd.exe /c netstat -an > c:\data\netstat\netstat_" & i & ".txt",7,FALSE
        Wscript.sleep 4000
        Set colProcessList = objWMIService.ExecQuery ("Select * from Win32_Process Where Name = 'vision.exe'")
        For Each objProcess in colProcessList
          objProcess.Terminate()
        Next
        Wscript.sleep 2000
        Wshshell.run "cmd.exe /c rename c:\data\screenshots\screenshot1.jpg ss_" & i & ".jpg",7,FALSE
        Wshshell.run "cmd.exe /c rename " & VISIONlog & " Visionlog_" & i & ".txt",7,FALSE
        IF (i=StopAT) THEN
                Done=true
        ELSE
                i=i+1
        END IF
Loop
Wscript.sleep 5000
Set FS_Obj = CreateObject("Scripting.FileSystemObject")
Dim DataFolders(3)
DataFolders(0)="C:\DATA\NETSTAT"
DataFolders(1)="C:\DATA\FPORT"
DataFolders(2)="C:\DATA\SCREENSHOTS"
DataFolders(3)="C:\DATA\VISION"
i=0
do While not(i=4)
        Set Folder_H = FS_Obj.GetFolder(DataFolders(i))
```

```
            Set Files_H = Folder_H.files
            For Each f1 in Files_H
                        Wshshell.run "cmd.exe /c md5sum " & DataFolders(i) & "\" & f1.name & " >>" & DataFolders(i) &
"\md5sum.txt",7,TRUE
            Next
            i=i+1
loop
```

To complement the script, Vision™ and TGrab™ will be configured to work within the time constraints in the script. Vision™ will update it's log file every five seconds.



Settings

AutoRefresh ✓

Rate Interval by Seconds 5

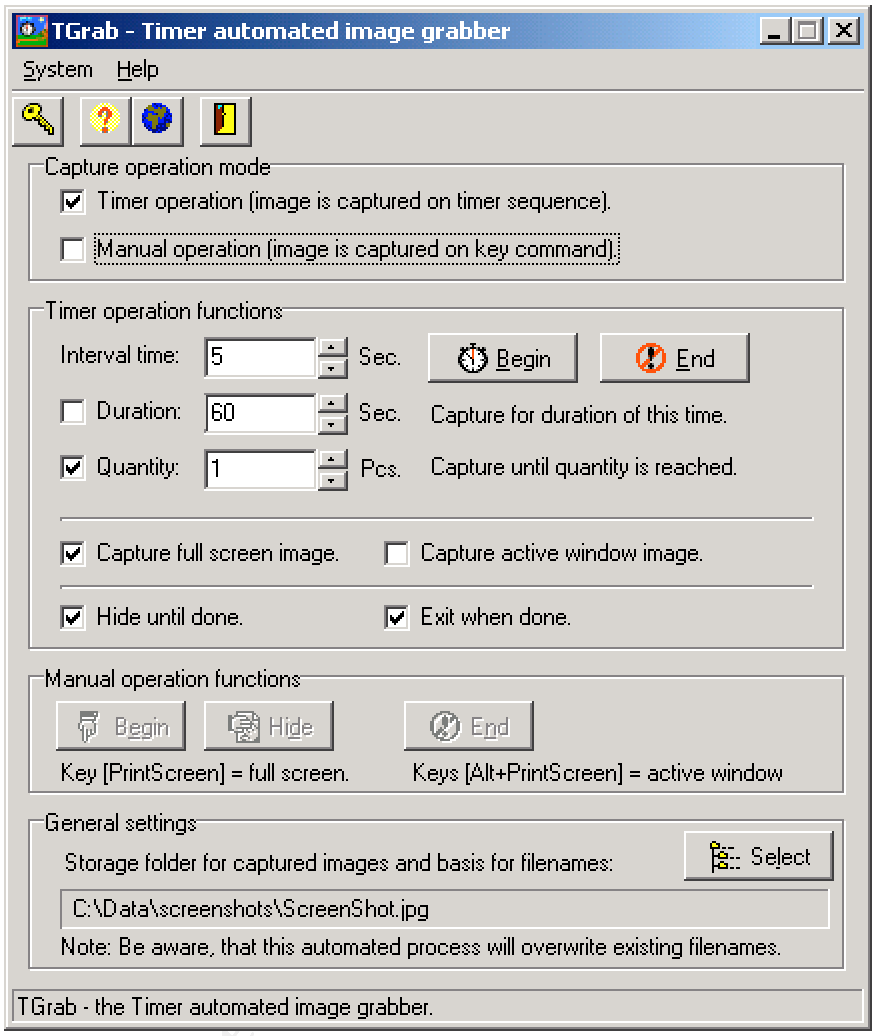
AutoLog ✓

Write to CSV ✓ C:\Data\vision\visionlog.txt

Figure 16 - Vision™ Setup

After vision™ is launched, there are two pauses in the script adding up to five seconds to allow for the screenshot. In this duration Vision™ will only write to it's log file once based on this configuration,

TGrab™ is configured to take one screenshot in 5 seconds and then terminate. Notice the option "Capture Full Screen image" is selected. When only taking one picture this setting is inverse, so the program is really taking a screen shot of the active application. The other setting of importance it the filename, when TGrab write the file it appends a number to the filename indicating which picture it is. (IE, if we took five screenshots the filenames would have the numbers 1 through 5 appended to them)



Capture operation mode

[✓] Timer operation (image is captured on timer sequence).

[ ] Manual operation (image is captured on key command).

Timer operation functions

Interval time: 5 Sec.

[ ] Duration: 60 Sec. Capture for duration of this time.

[✓] Quantity: 1 Pcs. Capture until quantity is reached.

[✓] Capture full screen image. [ ] Capture active window image.

[✓] Hide until done. [✓] Exit when done.

Manual operation functions

Key [PrintScreen] = full screen. Keys [Alt+PrintScreen] = active window

General settings

Storage folder for captured images and basis for filenames:

C:\Data\screenshots\ScreenShot.jpg

Note: Be aware, that this automated process will overwrite existing filenames.

TGrab - the Timer automated image grabber.

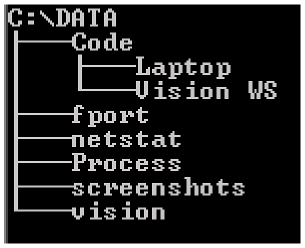
Figure 17 - TGrab Setup

B. Preflight Checks

1) Check all eight network applications are running and sending/receiving data.
2) Verify no data files exist in the four data directories.
    a. C:\DATA\FPORT\
    b. C:\DATA\NETSTAT\
    c. C:\DATA\VISION\
    d. C:\DATA\SCREENSHOTS\
3) Verify the application settings for Vision™ and TGrab™

C. Documentation Requirements

Four data directories will be the repositories of testing. The "code" directories are to contain the source code and compiled code for testing. The process directory is there for manual screenshots of the setup.



```
C:\DATA
├───Code
│   ├───Laptop
│   └───Vision WS
├───fport
├───netstat
├───Process
├───screenshots
└───vision
```

Figure 18 - Data Structure

C:\DATA\FPORT\

This directory will contain the output of fport. The naming convention for the files will be fport_#.txt where # is the iteration of the test that produced this result. (E.G, fport1005.txt will be the filename of the 1005th test)

C:\DATA\NETSTAT

This directory will contain the output of the "**Netstat –an**" command. The naming convention will be Netstat_#.txt where # is the iteration of the test.

C:\DATA\SCREENSHOTS

TGrab™ creates a file screenshot1.jpg when it captures the screen. After that is complete the vbs script will rename this file to ss_#.jpg where # is the iteration of the procedures the test is currently on.

C:\DATA\VISION

Vision™ is configured to automatically log to c:\data\vision\visionnlog.txt every five seconds. The script will rename this to visionlog_#.txt where # is the iteration of the procedures the test is currently working on.

D. Data Integrity

Prior to starting the test the mouse pointer will be move to the bottom of the screen to the task bar where an "eye" from running vision™ will appear. When Vision terminates, the "eye" icon in the taskbar remains until it is highlighted with the mouse. Because the program will be launched 1005 times the mouse pointer is placed here to avoid any memory leaks associated with 1005 "eye" icons on the taskbar. The pointer must be placed at the bottom of where the eye icon will appear. As the mouse hovers over the "eye", a popup appears saying "Vision". The mouse must be low enough that the popup does not appear in the active area of vision, otherwise the screenshots may vary and using a md5 checksum to compare the screenshots will not be feasible.

Once the mouse pointer is perfectly in place, the mouse cord is unplugged to prevent any unintentional movement. The keyboard will be unplugged once the test has been started as well.

Two of the programs (Fport and **netstat**) will be used to ensure the testing environment remains constant and no unintentional network connections are made. In the event of a change in the network, Fport and netstat can be used to qualify any changes in Vision™ data so the overall test will not be unusable.

E. Test Results

As long as the baseline results of Fport and netstat are constant, the expectations of the Vision™ output are:

- Each folder will have 1006 files. Any deviation from this means an error in testing and will be analyzed to determine the extent of the problem.
- There will be only one unique md5 message digest for each of the data folders with the exception of the vision text log. Any deviation from this must be accounted for.
- The vision text log will contain multiple md5 message digests as each log file includes date/time information. The expectation of this log file is that there will only be four unique TCP connections.
- The port and application information extracted from the Vision™ output (including screenshot) will be the same as Fport.

2.6 Criteria for Approval

There are three criteria for approval

1) Repeatable
   The results from test to test must be the same with the only provision for a discrepancy being that something on the system changes and the other tools running have the same discrepancy as vision™ v1.0.

2) Accurate
   The GUI[26] output of vision 1.0 must match up with the output of Fport. The text logging output of Vision™ must match the port list of netstat. Any discrepancy must have a valid, documented, technical explanation. Although the accuracy does not disqualify the output as evidence, it does make the product less of a useful tool.

3) Minimized Footprint
   This condition is the least tangible of the criteria in that the measurements used will be personal judgment of whether the footprint that vision™ 1.0 makes while running justifies the forensics information vision™ provides.

[26] Graphical User Interface

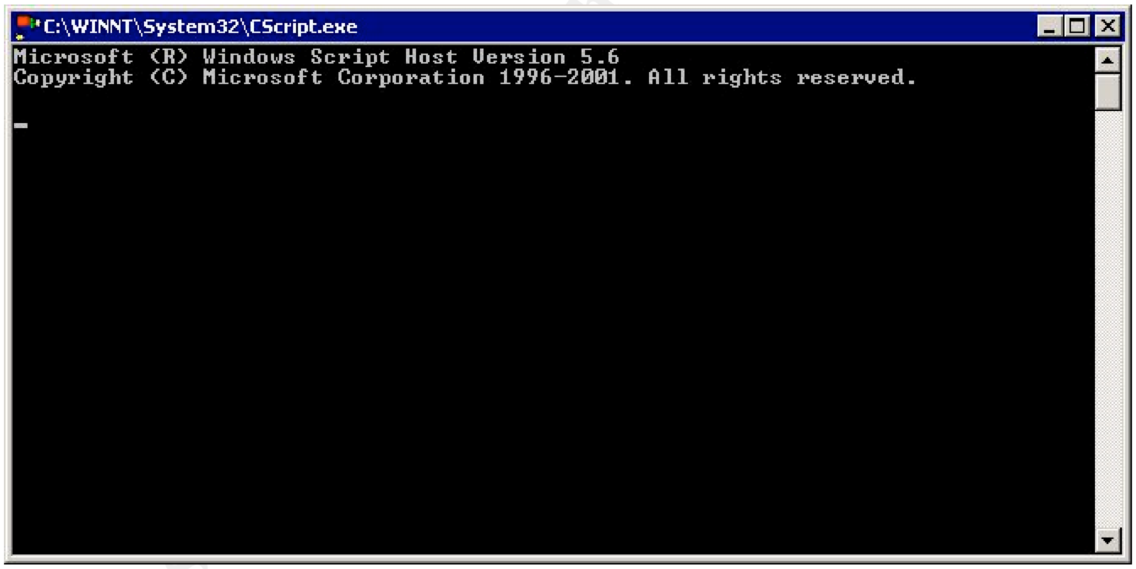
2.7 Data and Results

Before analyzing the data, any inconsistencies must be addressed. The data presented by this set of tests has several issues that must be addresses or removed from consideration.

A. Data Validation

The output of Fport™ has one unique md5 digest and the output of **netstat** has one unique md5 digest. These two facts make the testing environment clean and indicate that no network changes were introduced during the course of the test.

There are two unique md5 digests for the vision screenshots. Two screenshots have a digest of 4818373619c159c7a9e2f246285acd19 and the rest have a digest of 5f05d381797d05bb45dcca50db4df7cc. The two inconsistent screenshots are from iteration 12 and 925. In addition the vision™ log output only has 1004 files not 1006, the two missing iterations are 12 and 925.



```
Microsoft (R) Windows Script Host Version 5.6
Copyright (C) Microsoft Corporation 1996-2001. All rights reserved.

_
```

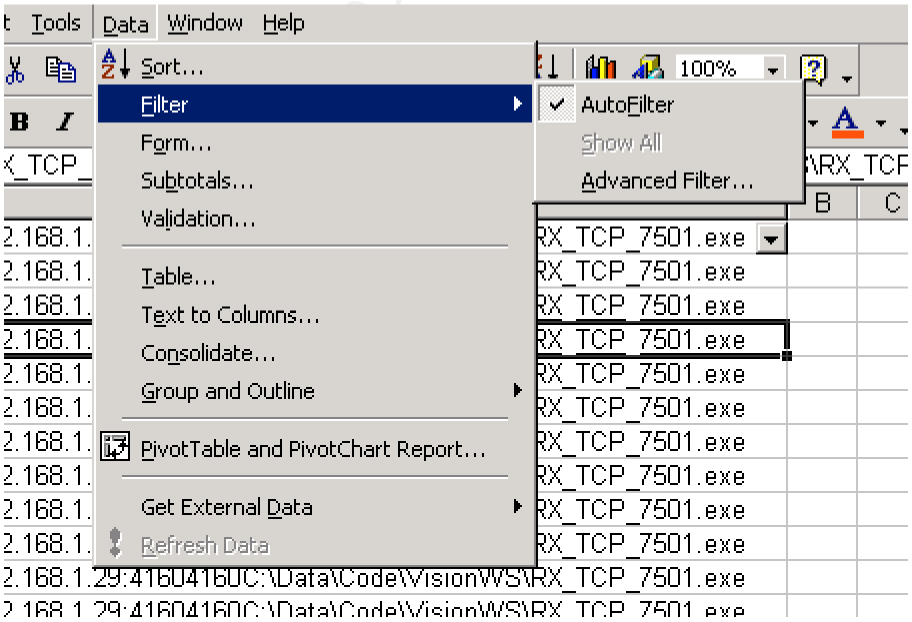
Figure 19 - ss_12.jpg and ss_925.jpg

The screenshots and total lack of log output for these two iterations seem to indicate the vbs script failed to start Vision™. These two iterations will be removed from consideration, as they do not show that Vision™ was inconsistent or inaccurate. This only shows that Vision™ failed to execute.

B. Data Analysis (Repeatable)

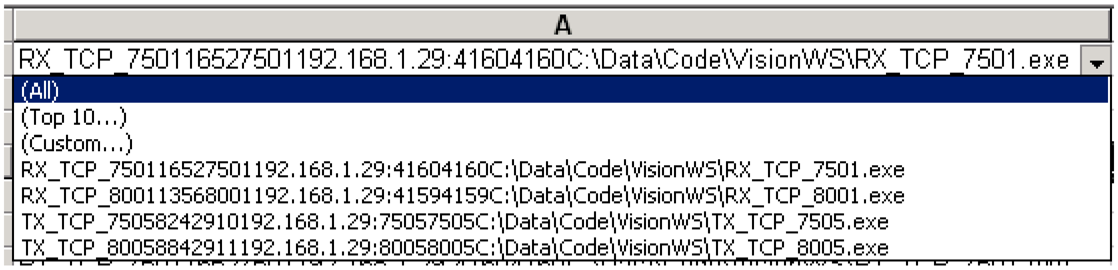
With the two exceptions out of the way, the remainder of the data shows that for 1003 iterations, the screenshot data is repeatable and reproducible. The md5 digest of the vision log data is inconclusive as the vision log data contains date and time information. Each file has a unique md5 digest. Using the command **dir** shows that 81 of the files have a size of 1020 bytes while the remaining 922 files have a file size of 510 bytes. Examining the contents shows that the larger files have the same information twice. It appears that the test was not consistently timed and that in some instances Vision™ logged twice. The setup for Vision should have set a much longer time that the 5 seconds allotted in the setup phase.

To use the information gathered constructively to prove the repeatability of Vision™, we will analyze the data contained within. Once compiled the data should only have four unique TCP connections. (Remember, the vision log file only contains active TCP connections). After piping all the data into one comma delimited file, the total amount of records is 4336. The result of 4336 divided by 4 is 1084 so this file may be 1084 iterations of the four TCP connections. To prove this, the data is imported into Microsoft Excel™. The date and time information is removed. All the fields are run together so they form one run-on field. Then using the auto filter function in Microsoft Excel™ the filter bar show 4 unique records.



```
2.168.1.29:4160416OC:\Data\Code\VisionWS\RX_TCP_7501.exe
2.168.1.29:4160416OC:\Data\Code\VisionWS\RX_TCP_7501.exe
```

Figure 20 – Microsoft™ Excel Auto Filter



```
A
RX_TCP_750116527501192.168.1.29:41604160C:\Data\Code\VisionWS\RX_TCP_7501.exe
(All)
(Top 10...)
(Custom...)
RX_TCP_750116527501192.168.1.29:41604160C:\Data\Code\VisionWS\RX_TCP_7501.exe
RX_TCP_800113568001192.168.1.29:41594159C:\Data\Code\VisionWS\RX_TCP_8001.exe
TX_TCP_75058242910192.168.1.29:75057505C:\Data\Code\VisionWS\TX_TCP_7505.exe
TX_TCP_80058842911192.168.1.29:80058005C:\Data\Code\VisionWS\TX_TCP_8005.exe
```

Figure 21 - Microsoft™ Excel Autofilter Results

The four unique records goes to prove that the data retrieved from Vision™ is repeatable and reproducible.

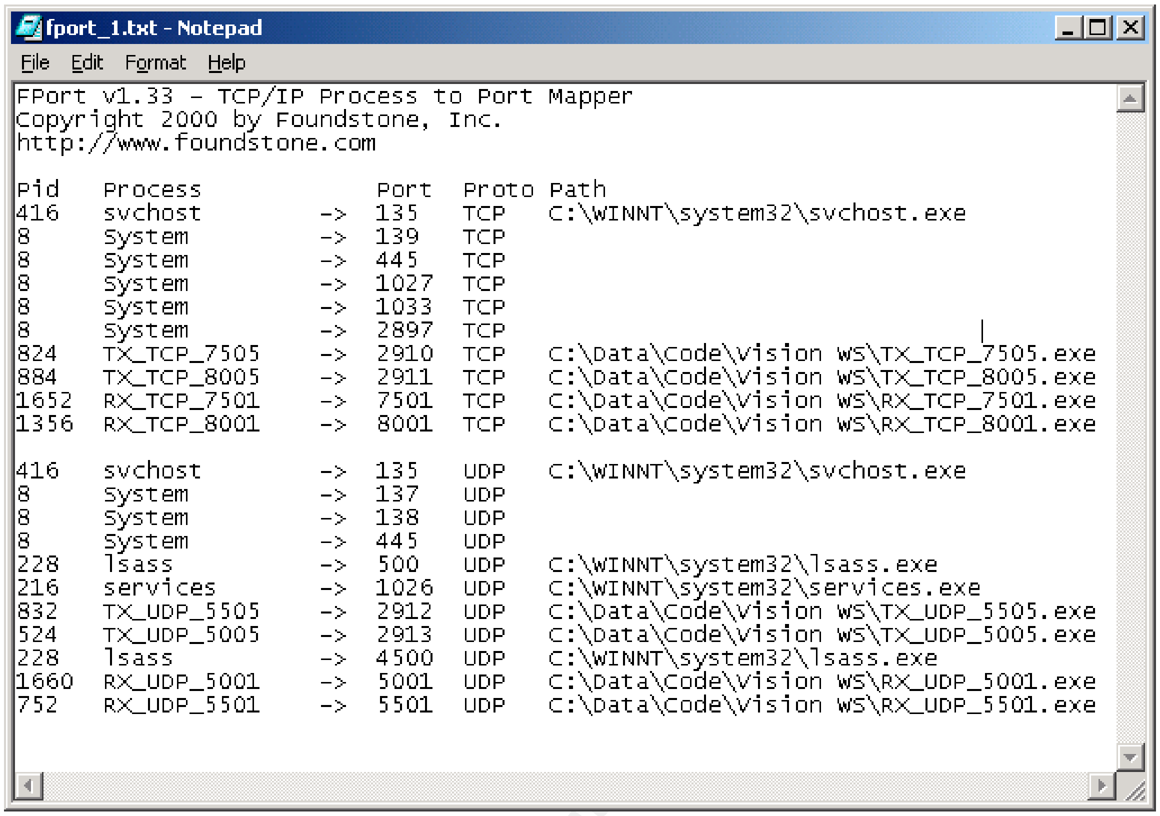
C. Data Analysis (Accurate)

The log file of Vision™ contains all the active TCP ports that Fport and **netstat** show. It does not however contain all the UDP TCP port information and without the screenshot of the GUI should not be considered an accurate representation of the UDP/TCP connections.

The GUI screen of Vision™ is a direct reflection of Fport and Netstat. In the output of netstat, it appears that there are more ports showing up. This is because netstat uses the source IP address as well. This mean that one application listening on port 137, will show up multiple times in netstat depending on how many ip addresses it has. The same port only shows up once in Fport and Vision™ as UDP port 137.

D. Data Results

The following figures contain screenshots of the unique results gathered by this test. The screenshots have been cropped for readability.

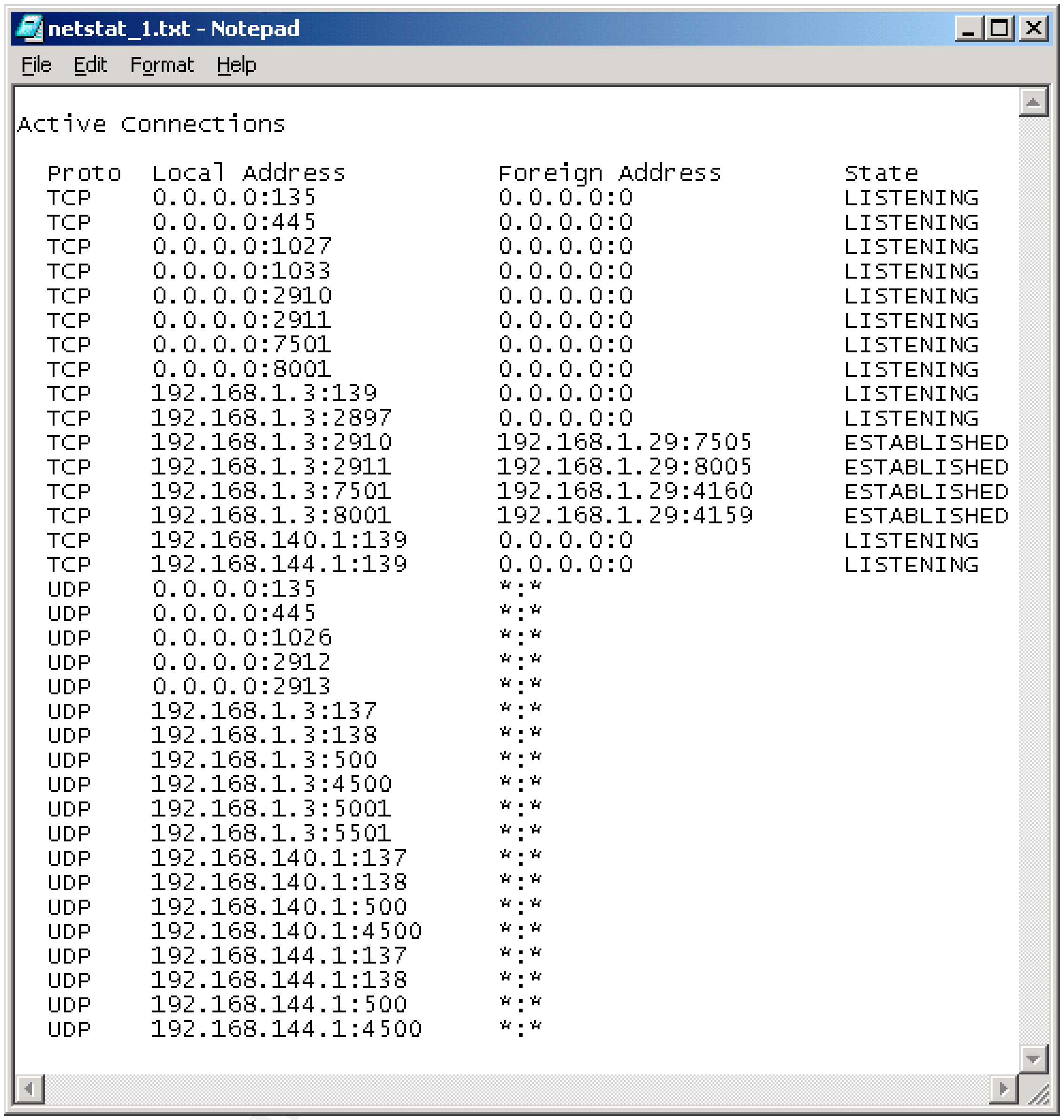


```
FPort v1.33 - TCP/IP Process to Port Mapper
Copyright 2000 by Foundstone, Inc.
http://www.foundstone.com

Pid   Process            Port  Proto Path
416   svchost         -> 135   TCP   C:\WINNT\system32\svchost.exe
8     System          -> 139   TCP
8     System          -> 445   TCP
8     System          -> 1027  TCP
8     System          -> 1033  TCP
8     System          -> 2897  TCP
824   TX_TCP_7505     -> 2910  TCP   C:\Data\Code\Vision WS\TX_TCP_7505.exe
884   TX_TCP_8005     -> 2911  TCP   C:\Data\Code\Vision WS\TX_TCP_8005.exe
1652  RX_TCP_7501     -> 7501  TCP   C:\Data\Code\Vision WS\RX_TCP_7501.exe
1356  RX_TCP_8001     -> 8001  TCP   C:\Data\Code\Vision WS\RX_TCP_8001.exe

416   svchost         -> 135   UDP   C:\WINNT\system32\svchost.exe
8     System          -> 137   UDP
8     System          -> 138   UDP
8     System          -> 445   UDP
228   lsass           -> 500   UDP   C:\WINNT\system32\lsass.exe
216   services        -> 1026  UDP   C:\WINNT\system32\services.exe
832   TX_UDP_5505     -> 2912  UDP   C:\Data\Code\Vision WS\TX_UDP_5505.exe
524   TX_UDP_5005     -> 2913  UDP   C:\Data\Code\Vision WS\TX_UDP_5005.exe
228   lsass           -> 4500  UDP   C:\WINNT\system32\lsass.exe
1660  RX_UDP_5001     -> 5001  UDP   C:\Data\Code\Vision WS\RX_UDP_5001.exe
752   RX_UDP_5501     -> 5501  UDP   C:\Data\Code\Vision WS\RX_UDP_5501.exe
```

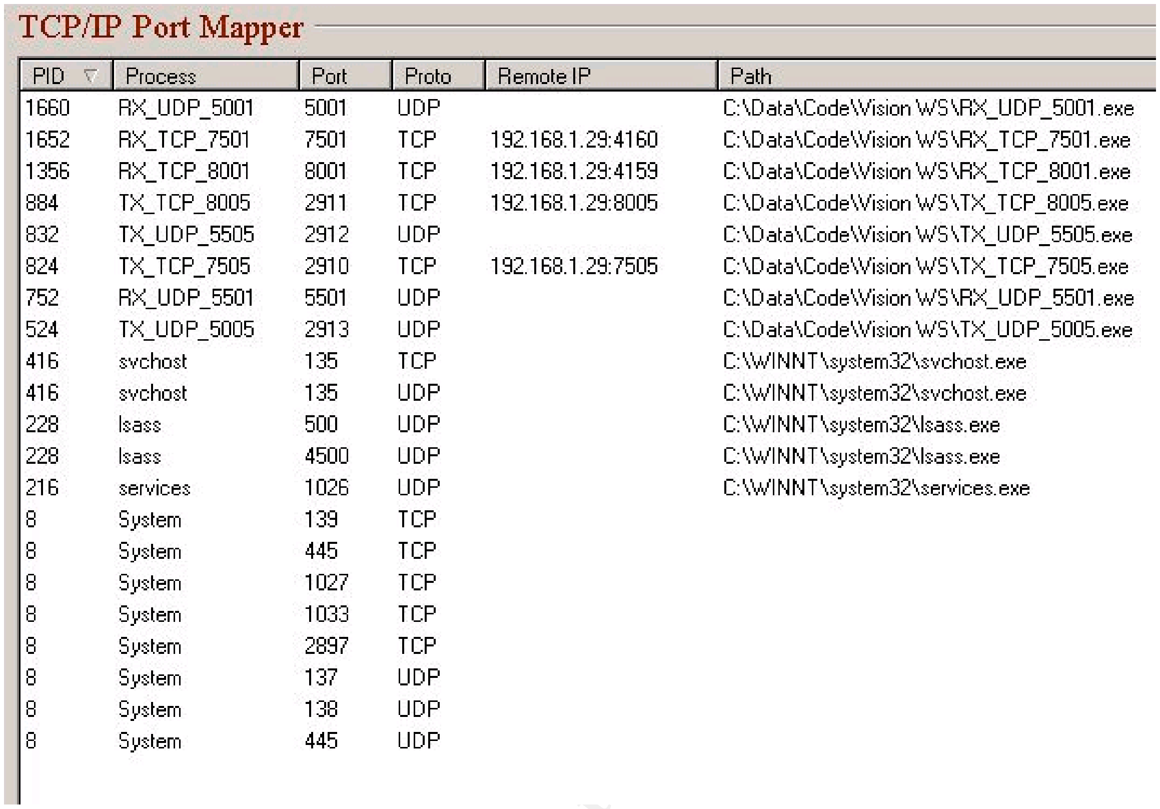
Figure 22 - FPort Results



```
Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1027           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1033           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:2910           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:2911           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:7501           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:8001           0.0.0.0:0              LISTENING
  TCP    192.168.1.3:139        0.0.0.0:0              LISTENING
  TCP    192.168.1.3:2897       0.0.0.0:0              LISTENING
  TCP    192.168.1.3:2910       192.168.1.29:7505      ESTABLISHED
  TCP    192.168.1.3:2911       192.168.1.29:8005      ESTABLISHED
  TCP    192.168.1.3:7501       192.168.1.29:4160      ESTABLISHED
  TCP    192.168.1.3:8001       192.168.1.29:4159      ESTABLISHED
  TCP    192.168.140.1:139      0.0.0.0:0              LISTENING
  TCP    192.168.144.1:139      0.0.0.0:0              LISTENING
  UDP    0.0.0.0:135            *:*
  UDP    0.0.0.0:445            *:*
  UDP    0.0.0.0:1026           *:*
  UDP    0.0.0.0:2912           *:*
  UDP    0.0.0.0:2913           *:*
  UDP    192.168.1.3:137        *:*
  UDP    192.168.1.3:138        *:*
  UDP    192.168.1.3:500        *:*
  UDP    192.168.1.3:4500       *:*
  UDP    192.168.1.3:5001       *:*
  UDP    192.168.1.3:5501       *:*
  UDP    192.168.140.1:137      *:*
  UDP    192.168.140.1:138      *:*
  UDP    192.168.140.1:500      *:*
  UDP    192.168.140.1:4500     *:*
  UDP    192.168.144.1:137      *:*
  UDP    192.168.144.1:138      *:*
  UDP    192.168.144.1:500      *:*
  UDP    192.168.144.1:4500     *:*
```

Figure 23 - Netstat -an Results


TCP/IP Port Mapper

PID	Process	Port	Proto	Remote IP	Path
1660	RX_UDP_5001	5001	UDP		C:\Data\Code\Vision WS\RX_UDP_5001.exe
1652	RX_TCP_7501	7501	TCP	192.168.1.29:4160	C:\Data\Code\Vision WS\RX_TCP_7501.exe
1356	RX_TCP_8001	8001	TCP	192.168.1.29:4159	C:\Data\Code\Vision WS\RX_TCP_8001.exe
884	TX_TCP_8005	2911	TCP	192.168.1.29:8005	C:\Data\Code\Vision WS\TX_TCP_8005.exe
832	TX_UDP_5505	2912	UDP		C:\Data\Code\Vision WS\TX_UDP_5505.exe
824	TX_TCP_7505	2910	TCP	192.168.1.29:7505	C:\Data\Code\Vision WS\TX_TCP_7505.exe
752	RX_UDP_5501	5501	UDP		C:\Data\Code\Vision WS\RX_UDP_5501.exe
524	TX_UDP_5005	2913	UDP		C:\Data\Code\Vision WS\TX_UDP_5005.exe
416	svchost	135	TCP		C:\WINNT\system32\svchost.exe
416	svchost	135	UDP		C:\WINNT\system32\svchost.exe
228	lsass	500	UDP		C:\WINNT\system32\lsass.exe
228	lsass	4500	UDP		C:\WINNT\system32\lsass.exe
216	services	1026	UDP		C:\WINNT\system32\services.exe
8	System	139	TCP		
8	System	445	TCP		
8	System	1027	TCP		
8	System	1033	TCP		
8	System	2897	TCP		
8	System	137	UDP		
8	System	138	UDP		
8	System	445	UDP		

Figure 24 - Vision™ GUI Results (cropped for readability)

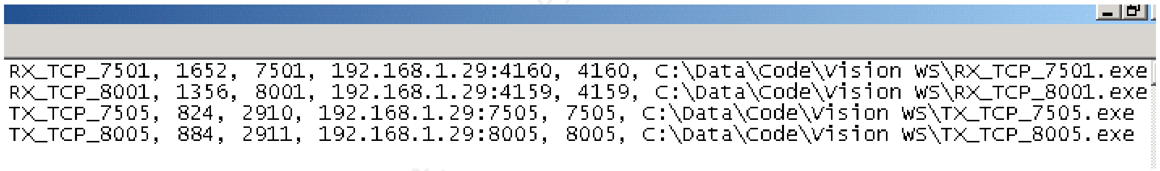

```
RX_TCP_7501, 1652, 7501, 192.168.1.29:4160, 4160, C:\Data\Code\Vision WS\RX_TCP_7501.exe
RX_TCP_8001, 1356, 8001, 192.168.1.29:4159, 4159, C:\Data\Code\Vision WS\RX_TCP_8001.exe
TX_TCP_7505, 824, 2910, 192.168.1.29:7505, 7505, C:\Data\Code\Vision WS\TX_TCP_7505.exe
TX_TCP_8005, 884, 2911, 192.168.1.29:8005, 8005, C:\Data\Code\Vision WS\TX_TCP_8005.exe
```

Figure 25 - Vision™ Log File Results

2.8 Analysis

The data collected by Vision™ has a couple of uses to a forensic investigator. In the case of finding a suspect machine running, Vision™ can provide proof that an executable was running and communicating on the network. The data will also reveal an ip address that the host is connected to for TCP connections.

Vision™ also determines what application corresponds with a network port. The output can pinpoint what application received or transmitted specific data if the forensic analyst has network traces to lead him to a specific Microsoft™ NT or 2000 host. This tool would be used in scenarios where it is expected or possible for a backdoor program to be installed.

2.9 Presentation

Vision™ outputs data in two ways. The comma delimited log file (although nice in that is can log over a period of time) contains only a portion of the data and until this changes, the presentation of Vision™ data is a screenshot. The log file can be used in the case where an active TCP connection is involved.

The output of the tool is a list of six fields of data as shown

A. PID

This is the process identification number. These numbers are assigned by the operating system sequentially so can be used to determine what sequence the processes were executed.

B. Process

This is the name of the process.

C. Port

The port number the process is communicating with. This number can be anywhere in the range from 1 to 65535.

D. Protocol

This field will be either TCP or UDP.

E. Remote IP

This will display an IP address of an active TCP connection

F. Path

This displays the full path to the program that is communicating on this port.

2.10 Conclusion

This validation was only partly successful. It has proven that Vision™ by Foundstone™ can provide accurate, repeatable and reproducible results. The footprint of running this product is significant enough that it should not be used in an incidence response scenario. This is reinforced when compared to Fport, which has no footprint except for the MRU (Most Recently Used). Vision™ can be used forensically where the evidence has already been imaged and evidence containment is not an issue. (Such as on a copy of the data)

Recommendations

The text based logging should include all the port/application data collected by Vision™ and not just the active TCP connections. This would remove the need for screenshots as a presentation mechanism. In addition, this would allow this program to be used in an ongoing investigation to collect connection information over time. In this context, the footprint may be a worthwhile price to pay for the function.

Reduce the footprint. This program references several dll files from the system drive that could be access from a cd-rom. The other suggestion along these lines is too remove the need to make registry entries.

2.11 Additional Information

Reference

A Simple Multi-threaded TCP/UDP Server and Client v2 **By Patrick Lam** http://www.c-sharpcorner.com/Network/simpleTcpUdpServerClientPL2.asp

The Wizard's Hot Keys/Shortcuts http://www.wizardscave.com/hotkeys.html

Windows Scripting Host http://www.devguru.com/Technologies/wsh/quickref/wsh_objects.html

Microsoft Windows Script http://msdn.microsoft.com/scripting

Software

http://www.go-mono.com/	Mono (Win32 C# Compiler)
http://md5deep.sourceforge.net	MD5Deep
http://www.microsoft.com	Windows 2000, Windows XP, Windows NT, Windows 98
http://www.foundstone.com	Fport
http://www.foundstone.com	Vision v1.0
http://www.doldersum.com/	TGrab v1.4
http://www.sysinternals.com	Regmon and Filemon

Part 3 ~ Legal Issues of Incident Handling

3.1 Introduction

This part of the assignment will delve into the depths of Canadian law and provide some insight of the law and how it relates to electronic documents and computer networks.

The three main acts that deal with this subject are the Canada Criminal Code, Canada Evidence Act, and Canada Personal Information Protection and Electronic Documents Act. Canada has very recently introduced federal laws regarding electronic evidence and information. Some provinces have further outlined the federal regulations, however British Columbia has not yet adopted any provincial regulations regarding electronic information.

Throughout this part, I will be personifying a system administrator working for an Internet service provider. Sgt Boggs is a fictional character that will be used to portray some aspects of Canadian Law. For purposes of the following scenarios, I know Sgt Boggs as law enforcement officer and there is no need to verify his identity.

Phone:	*"Brrring. Brrring."*
Bil the sysadmin:	*"Hello"*
Sgt Boggs[27]:	*"Hello, this is Sgt Boggs from the Royal Canadian Mounted Police, I am calling to inform you of criminal activity coming from an account on your systems. It appears that an account has hacked into a government computer. Can you review your activity logs and determine if this activity came from your systems or from a downstream provider? The timeline in question is yesterday from three to four pm."*

[27] Sgt Boggs as depicted in this paper is a purely fictional character and any resemblance to any real person is unintentional.

3.2 The Law

A. Disclosure of Information

Bil The sysadmin:	*"After reviewing my logs, the only thing I can find is ISPUSER_01 was logged in during that period. "*
Sgt Boggs:	*"Great, tell me everything you know about ISPUSER_01."*
Bil The sysadmin:	*"Ok, just a sec, I will look it up."*

According to the Canada Personal Information Protection and Electronic Documents Act[28], I can provide the law enforcement officer any information I have on ISPUSER_01 as the request was made from an official source for the purpose of investigating a crime. I can do this without the consent of the user in question under these circumstances. This scenario also assumes that the information collected by the ISP about ISPUSER_01 is legal, which is a different law.

B. Electronic Evidence

Bil the sysadmin:	*"… And that's all the info we have on ISPUSER_01"*
Sgt Boggs:	*" Ok, I have all that written down now. Can you make a copy of the logs files in question and get a md5 checksum on them?"*

In Canadian Federal Evidence Act, there are two special stipulations for electronic evidence. The normal laws and regulations on admissibility do apply except for these two stipulations:

a) Authenticity

The person(s) bringing the electronic record must also have evidence that the document is what they claim it is[29]. This evidence is typically verbal and in the case of this scenario would be, "This is a log of users logged in."

b) Integrity

To be admissible into legal proceedings, the electronic document must have a certain degree of integrity. This is some degree of certainty that the document has not changed from the original. Given this situation the integrity of the logs could be proven under three clauses of the Canada Evidence Act.

[28] Canada Personal Information Protection and Electronic Documents Act 7(3)(h.2)
[29] Canada Evidence Act 31.1

1) The best evidence rule states, “on proof of the integrity of the electronic documents system by or in which the electronic document was recorded or stored”[30]. This clause could be satisfied by an md5[31] checksum on the drive and/or copy of file at the point in time in question.

2) Further to the best evidence rule, a presumption of integrity exists if the system the electronic document is taken from is working reliably as per the Evidence act[32].

3) Another clause in the presumption of integrity states, “if it is established that the electronic document was recorded or stored in the usual and ordinary course of business by a person who is not a party and who did not record or store it under the control of the party seeking to introduce it”[33]. In this case the logs I was reviewing were the user access logs recorded to calculate billing information.

C. Legal Authority

Sgt Boggs: *“Great I will come by a and pick up that log file on disk later on “*

As discussed in the answer to question A, the Canada Personal Information Protection and Electronic Documents Act has a clause for law enforcement access to private information collected in the normal course of business. Sgt Boggs in this case does not need any further legal authority for me to send him/her the logs.

D. Further Investigation

Sgt Boggs: *“Thank you for all you cooperation. If you can find out anything more, let me know”*

Another part of the Personal Protection and Electronic Documents Act[34] allows me to review and search for any recorded personal information without the knowledge or consent from the suspect, as I know that it may be used in a criminal investigation.

[30] Canada Evidence Act 31.2(1)(a)
[31] md5 message digest as outlined by Ronald Rivest in RFC 1521.
[32] Canada Evidence Act 31.3(a)
[33] Canada Evidence Act 31.3(c)
[34] Personal Protection and Electronic Documents Act 7.2(a)

The criminal code[35] would prevent me from eavesdropping on the wire for his activity until such a time as the law enforcement officer provided a court order. I could however review any intrusion detection systems or network sniffer logs that are already in place to protect the ISP under a provision of the criminal code[36]. The distinction here is I cannot put a new sniffer or modify the current eavesdropping equipment to specifically look at the suspect's activity.

E. Evidence Collection

Bil the sysadmin:	*"Oh, I have new information based on the IDS logs, ISPUSER_01 was hacked into using the Ijustmadethisup vulnerability. Another account was created called ISPHACK_01 and that account was also used to hack into that government system"*
Sgt Boggs:	*"…"*

The law enforcement officer at that point in time can obtain a search warrant[37] and seize the computer equipment in question. The other alternatives for the law enforcement officer are to obtain the logs/data[38] or have me obtain the logs, data and/or image. [39] The alternatives listed here are the ones available by law, I am unsure of the standard practices (if there is one) the RCMP[40] deploy in these scenarios.

[35] Canada Criminal Code Part IV Invasion of Privacy 184. (1)
[36] Canada Criminal Code Part IV Invasion of Privacy 184(2)(c)(iii)
[37] Canada Criminal Code Part XV Special Procedure and Powers 487(1)(d)
[38] Canada Criminal Code Part XV Special Procedure and Powers 487 (2.1)
[39] Canada Criminal Code Part XV Special Procedure and Powers 487 (2.2)
[40] Royal Canadian Mounted Police

3.3 Additional Information

References

Privacy Commissioner of Canada
http://www.privcom.gc.ca/

Canada's Parliament
http://www.parl.gc.ca/

Department of Justice Canada
http://laws.justice.gc.ca/

APPENDIX

A.1 Network Connection Code

This c-sharpe code was designed based on Patrick Lam's UDP/TCP examples[41]. For brevity, only four programs are shown here. The remaining code are these with the name and ports changed accordingly.

A. RX_TCP_7501.cs

The c# code will open up a TCP listening socket on port 7501.

```
namespace RX_TCP_7501
{
	using System;
	using System.Net;
	using System.Net.Sockets;
	using System.Threading;

	public class RX_TCP_7501
	{
		private const int Port = 7501;
		public static void Main(String[] argv)
		{
			TcpListener tcpListener = new TcpListener(Port);
			try
			{
				tcpListener.Start();
				Socket soTcp = tcpListener.AcceptSocket();
				while (true)
				{
					Byte[] received = new Byte[512];
					int bytesReceived = soTcp.Receive(received, received.Length, 0);
					String dataReceived = 
System.Text.Encoding.ASCII.GetString(received);
					Console.WriteLine(dataReceived);
					String returningString = "The Server got your message through TCP: " 
+ dataReceived;
					Byte[] returningByte = 
System.Text.Encoding.ASCII.GetBytes(returningString.ToCharArray());
					soTcp.Send(returningByte, returningByte.Length, 0);
				}
				tcpListener.Stop();
			}
			catch (SocketException se)
			{
				Console.WriteLine("A Socket Exception has occurred!" + se.ToString());
			}
		}
	}
}
```

B. RX_UDP_5001.cs

This c# code will listen on port 5001 for UDP packets.

```
namespace RX_UDP_5001
{
	using System;
```

[41] The article on this can be viewed at http://www.c-sharpcorner.com/Network/simpleTcpUdpServerClientPL2.asp

```
	using System.Net;
	using System.Net.Sockets;
	using System.Threading;

	public class RX_UDP_5001
	{
		private const int Port = 5001;
		public Thread UdpThread;
		public static void Main(String[] argv)
		{
			IPHostEntry localHostEntry;
			try
			{
				//Create a UDP socket.
				Socket soUdp = new Socket(AddressFamily.InterNetwork, SocketType.Dgram,
ProtocolType.Udp);
				try
				{
					localHostEntry = Dns.GetHostByName(Dns.GetHostName());
				}
				catch(Exception)
				{
					Console.WriteLine("Local Host not found"); // fail
					return ;
				}
				IPEndPoint localIpEndPoint = new IPEndPoint(localHostEntry.AddressList[0],
Port);
				soUdp.Bind(localIpEndPoint);
				while (true)
				{
					Byte[] received = new Byte[256];
					IPEndPoint tmpIpEndPoint = new
IPEndPoint(localHostEntry.AddressList[0],Port);
					EndPoint remoteEP = (tmpIpEndPoint);
					int bytesReceived = soUdp.ReceiveFrom(received, ref remoteEP);
					String dataReceived =
System.Text.Encoding.ASCII.GetString(received);
					Console.WriteLine(dataReceived);
					String returningString = "The Server got your message through UDP:"
+ dataReceived;
					Byte[] returningByte =
System.Text.Encoding.ASCII.GetBytes(returningString.ToCharArray());
					soUdp.SendTo(returningByte, remoteEP);
				}
			}
			catch (SocketException se)
			{
				Console.WriteLine("A Socket Exception has occurred!" + se.ToString());
			}
		}
	}
}
```

C. TX_TCP_7501.cs

This c# code will send TCP packets to the ip address given as a parameter on port 7501.

```
namespace TX_TCP_7501
{
	using System;
	using System.Net;
	using System.Net.Sockets;
	using System.Threading;
	public class TX_TCP_7501
	{
		private const int Port = 7501;
		public static void Main(String[] argv)
		{
			if (argv.Length == 1)
			{
				try
				{
					TcpClient tcpClient = new TcpClient(argv[0],Port);
					NetworkStream tcpStream = tcpClient.GetStream();
					int i = 1;
					while(true)
					{
						bool DONE = false;
						if (tcpStream.CanWrite)
						{
							String StringtobeSent = i + ": Testing TCP On Port
" + Port;
							Byte[] ASCIIToBeSent =
System.Text.Encoding.ASCII.GetBytes(StringtobeSent.ToCharArray());
							tcpStream.Write(ASCIIToBeSent, 0,
ASCIIToBeSent.Length);
							tcpStream.Flush();
						}
						while (tcpStream.CanRead && !DONE)
						{
							if (tcpStream.DataAvailable)
							{
								Byte[] received = new Byte[512];
								int nBytesReceived =
tcpStream.Read(received, 0, received.Length);
								String dataReceived =
System.Text.Encoding.ASCII.GetString(received);
								Console.WriteLine(dataReceived);
								DONE = true;
							}
						}
						i=i+1;
					}
					tcpStream.Close();

				}
				catch (Exception e)
				{
					Console.WriteLine("An Exception has occurred.");
					Console.WriteLine(e.ToString());
				}
			}
			else
			{
				Console.WriteLine("Usage: TX_TCP_7501 a.b.c.d");
			}
		}
	}
}
```

D. TX_UDP_5001.cs

This c# code will send UDP packets to the ip address given as a parameter on port 5001.

```
namespace TX_UDP_5001
{
	using System;
	using System.Net;
	using System.Net.Sockets;
	using System.Threading;
	public class TX_UDP_5001
	{
		private const int Port = 5001;
		public static void Main(String[] args)
		{
					if (args.Length == 1)
					{
						try
						{
							UdpClient UDPStream = new UdpClient(args[0], Port);
							Byte[] ASCIIText = new Byte[256];
							IPHostEntry remoteHost = Dns.GetHostByName(args[0]);
							IPEndPoint remoteIp = new
IPEndPoint(remoteHost.AddressList[0],Port);
							int i = 1;
							while(true)
							{
								string StringText = i + ": Testing UDP On Port " + Port;
								ASCIIText =
System.Text.Encoding.ASCII.GetBytes(StringText.ToCharArray());
								int nBytesTX = UDPStream.Send(ASCIIText,
ASCIIText.Length);
								Byte[] RX = new Byte[512];
								RX = UDPStream.Receive(ref remoteIp);
								String dataRX =
System.Text.Encoding.ASCII.GetString(RX);
								Console.WriteLine(dataRX);
								i=i+1;
							}
							UDPStream.Close();
						}
						catch (Exception e)
						{
							Console.WriteLine("An Exception Occurred!");
							Console.WriteLine(e.ToString());
						}
					}
					else
					{
						Console.WriteLine("Usage: TX_UDP_5001 a.b.c.d");
					}

				}
		}
}
```

A.2 Vision-Test.vbs

This script was used to generate 1005 iteration of Vision™. During the testing, two of sets of data were invalid. If you intend on using this script or a variation of it, you will have to adjust the wscript.sleep commands to allow enough time for the operation to complete. This will vary based on applications and computer speed.

```
StopAt=1005
NETSTATFolder="C:\DATA\NETSTAT"
FPORTFolder="C:\DATA\FPORT"
TGRABFolder="C:\DATA\SCREENSHOTS"
VISIONFolder="C:\DATA\VISION"
VISIONLog ="C:\DATA\VISION\Visionlog.txt"
set wshshell = createobject("wscript.shell")
Set objWMIService = GetObject("winmgmts:"  & "{impersonationLevel=impersonate}!\\.\root\cimv2")
Done = False
i=1
Do While Not(Done )
        wshshell.run "C:\Progra~1\TGrab\TGrab.exe"
        Wscript.sleep 2000
        wshshell.SendKeys "~"
        Wscript.sleep 1000
        wshshell.SendKeys "%b"
        wshshell.run "C:\Progra~1\Foundstone\Vision\Vision.exe"
        Wscript.sleep 1000
        wshshell.SendKeys "%"
        wshshell.SendKeys " "
        wshshell.SendKeys "x"
        wshshell.run "cmd.exe /c fport > c:\data\fport\fport_" & i & ".txt",7,FALSE
        wshshell.run "cmd.exe /c netstat -an > c:\data\netstat\netstat_" & i & ".txt",7,FALSE
        Wscript.sleep 4000
        Set colProcessList = objWMIService.ExecQuery ("Select * from Win32_Process Where Name = 'vision.exe'")
        For Each objProcess in colProcessList
          objProcess.Terminate()
        Next
        Wscript.sleep 2000
        wshshell.run "cmd.exe /c rename c:\data\screenshots\screenshot1.jpg ss_" & i & ".jpg",7,FALSE
        wshshell.run "cmd.exe /c rename " & VISIONlog & " Visionlog_" & i & ".txt",7,FALSE
        IF (i=StopAT) THEN
                Done=true
        ELSE
                i=i+1
        END IF
Loop
Wscript.sleep 5000
Set FS_Obj = CreateObject("Scripting.FileSystemObject")
Dim DataFolders(3)
DataFolders(0)="C:\DATA\NETSTAT"
DataFolders(1)="C:\DATA\FPORT"
DataFolders(2)="C:\DATA\SCREENSHOTS"
DataFolders(3)="C:\DATA\VISION"
i=0
do While not(i=4)
        Set Folder_H = FS_Obj.GetFolder(DataFolders(i))
        Set Files_H = Folder_H.files
        For Each f1 in Files_H
                wshshell.run "cmd.exe /c md5sum " & DataFolders(i) & "\" & f1.name & " >>" & DataFolders(i) &
"\md5sum.txt",7,TRUE
        Next
        i=i+1
loop
```