



Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

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

Interested in learning more?

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

Analyze an Unknown Binary, Perform Sfind Tool Validation and Legal Issue of Incident Handling in Japan

GCFA (GIAC Certified Forensic Analyst) Practical Assignment
Version 1.4

Akiteru Kamoshida

Submitted August 5, 2004

Attended CDIWest2004 at SanDiego
January 26-31, 2004

Table of contents

ABSTRACT	1
1. ANALYZE AN UNKNOWN BINARY	2
1.1. BINARY DETAILS	2
1.2. PROGRAM DESCRIPTION	5
1.3. FORENSIC DETAILS	7
1.4. PROGRAM IDENTIFICATION	12
1.5. LEGAL IMPLICATIONS	16
1.6. INTERVIEW QUESTIONS.....	17
1.7. CASE INFORMATION	17
1.8. ADDITIONAL INFORMATION	20
2. PERFORM FORENSIC TOOL VALIDATION.....	21
2.1. SCOPE	21
2.2. TOOL DESCRIPTION	23
2.3. TEST APPARATUS	25
2.4. ENVIRONMENTAL CONDITIONS	26
2.5. DESCRIPTION OF THE PROCEDURES.....	26
2.6. CRITERIA FOR APPROVAL	33
2.7. DATA AND RESULTS	34
2.8. ANALYSIS	45
2.9. PRESENTATION.....	47
2.10. CONCLUSION	48
2.11. ADDITIONAL INFORMATION	49
3. LEGAL ISSUES OF INCIDENT HANDLING	50
3.1. QUESTION A	50
3.2. QUESTION B	50
3.3. QUESTION C	51
3.4. QUESTION D.....	51
3.5. ADDITIONAL INFORMATION	52

Abstract

This paper provides 3 different matters about computer forensics.

The first chapter is about assignment part 1. It provides analysis of an unknown binary included in the floppy disk image file downloaded from GIAC web site.

The second chapter is about option 2 of assignment part 2. It provides validation of Sfind that scans and lists files hidden in alternative data stream of NTFS file system.

The third chapter is about assignment part 3. It provides legal issues of incident handling based on laws in Japan.

© SANS Institute 2004, Author retains full rights.

1. Analyze an Unknown Binary

The purpose of this chapter is to analyze an unknown binary included in the floppy disk image file downloaded from GIAC web site. Environment used for analysis is shown below.

OS: Red Hat Linux 9 (Japanese localized) Kernel 2.4.20-8

Hardware: Panasonic Let's note CF-R2

Machine Name: kamo-r2

CPU: Intel Centrino 1.0 GHz

Memory: 512 MB

1.1. Binary Details

The floppy disk image file is downloaded from GIAC web site. The image file is unzipped and checked as shown below. The purpose of this check is to ensure that the image file is neither damaged nor altered. The consistency of MD5 hash value of the image file and .md5 file included zip archive means that the image file is neither damaged nor altered.

```
[root@kamo-r2 gcfa_assignment]# unzip binary_v1_4.zip
Archive:  binary_v1_4.zip
GCFA binary analysis
  inflating: fl-160703-jp1.dd.gz
  extracting: fl-160703-jp1.dd.gz.md5
  extracting: prog.md5
[root@kamo-r2 gcfa_assignment]# md5sum fl-160703-jp1.dd.gz
4b680767a2aed974cec5fbcbf84cc97a  fl-160703-jp1.dd.gz
[root@kamo-r2 gcfa_assignment]# cat fl-160703-jp1.dd.gz.md5
4b680767a2aed974cec5fbcbf84cc97a  fl-160703-jp1.dd.gz
[root@kamo-r2 gcfa_assignment]# gunzip fl-160703-jp1.dd.gz
```

The image file is mounted on the mount point (/mnt/forensic) as shown below. The command option “ro” means that the image file is mounted with read only mode. The command option “noatime” means that access time of inode isn’t updated. These options are important for forensic investigation because these

protect evidence from being broken.

```
[root@kamo-r2 gcfa_assignment]# mount -o ro,noatime,loop fl-160703-jp1.dd
/mnt/forensic/
[root@kamo-r2 gcfa_assignment]# ls /mnt/forensic
Docs  John  May03  lost+found  nc-1.10-16.i386.rpm..rpm  prog
```

Some directories and files are found in the floppy disk image. To determine their file type, file command is run on the mount point (/mnt/forensic).

```
[root@kamo-r2 gcfa_assignment]# file /mnt/forensic/*
/mnt/forensic/Docs:                directory
/mnt/forensic/John:                directory
/mnt/forensic/May03:               directory
/mnt/forensic/lost+found:           directory
/mnt/forensic/nc-1.10-16.i386.rpm..rpm: RPM v3 bin i386 nc-1.10-16
/mnt/forensic/prog:                 ELF 32-bit LSB executable, Intel 80386,
version 1 (SYSV), for GNU/Linux 2.2.5, statically linked, stripped
[root@kamo-r2 gcfa_assignment]# file /mnt/forensic/Docs/*
/mnt/forensic/Docs/DVD-Playing-HOWTO-html.tar: POSIX tar archive
/mnt/forensic/Docs/Kernel-HOWTO-html.tar.gz:  gzip compressed data, was
"Kernel-HOWTO-html.tar", from Unix
/mnt/forensic/Docs/Letter.doc:       Microsoft Office Document
/mnt/forensic/Docs/MP3-HOWTO-html.tar.gz:  gzip compressed data, was
"MP3-HOWTO-html.tar", from Unix
/mnt/forensic/Docs/Mikemsg.doc:      Microsoft Office Document
/mnt/forensic/Docs/Sound-HOWTO-html.tar.gz:  gzip compressed data, was
"Sound-HOWTO-html.tar", from Unix
[root@kamo-r2 gcfa_assignment]# file /mnt/forensic/John/*
/mnt/forensic/John/sect-num.gif: GIF image data, version 87a, 145 x 145
/mnt/forensic/John/sectors.gif: GIF image data, version 87a, 282 x 131
[root@kamo-r2 gcfa_assignment]# file /mnt/forensic/May03/*
/mnt/forensic/May03/ebay300.jpg: JPEG image data, JFIF standard 1.01, resolution
(DPI), 96 x 96
[root@kamo-r2 gcfa_assignment]# file /mnt/forensic/lost+found/*
/mnt/forensic/lost+found/*: can't stat `/mnt/forensic/lost+found/*' (No such file
or directory).
```

The output shows that there is an executable binary file that named as “prog”. The binary is ELF 32-bit LSB executable file that runs on Intel 80386 based Linux platform and uses statically linked libraries. As a result of further investigation, detail of the binary is shown below.

Name	prog
Mac Time	Modified at 2003-07-14 23:24:00 Accessed at 2003-07-16 15:12:45 Changed at 2003-07-16 15:05:33 (MAC Time shown above is represented in Japanese local time.)
File owner	an user whose user-id/group-id is 502/502
File size	487476 bites
MD5 hash	7b80d9aff486c6aa6aa3efa63cc56880
Key words	1.0.20 (07/15/03) newt extract a copy from the raw device use block-list knowledge to perform special operations on files prog bmap_get_block_size bmap_map_block bmap_raw_open mode slack size: %d

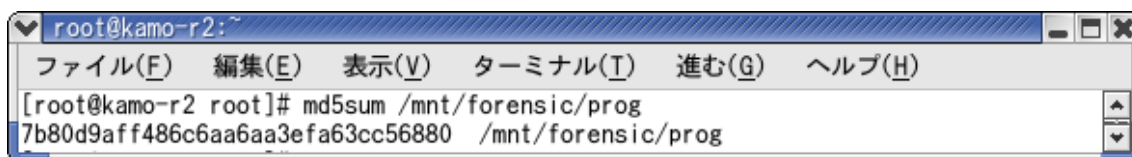
The investigation method is shown below.

1. To investigate detail of the binary, stat command is run on the binary file.
The result of stat command shows mac time, file owner and files size of the binary. Uid and Gid of file owner are both shown as "502 / UNKNOWN". This is because that Uid 502 and Gid 502 aren't exist on the "/etc/passwd" file of the analysis machine. Uid 502 and Gid 502 may exist on the machine that is used for creating the floppy disk.

```
[root@kamo-r2 root]# stat /mnt/forensic/prog
  File: `/mnt/forensic/prog'
  Size: 487476      Blocks: 960      IO Block: 4096   Regular File
Device: 700h/1792d  Inode: 18       Links: 1
Access: (0755/-rwxr-xr-x)  Uid: ( 502/ UNKNOWN)   Gid: ( 502/ UNKNOWN)
Access: 2003-07-16 15:12:45.000000000 +0900
Modify: 2003-07-14 23:24:00.000000000 +0900
Change: 2003-07-16 15:05:33.000000000 +0900
```

2. To calculate MD5 hash value, md5sum command is run on the binary

file.



```
root@kamo-r2:~  
ファイル(F) 編集(E) 表示(V) ターミナル(T) 進む(G) ヘルプ(H)  
[root@kamo-r2 root]# md5sum /mnt/forensic/prog  
7b80d9aff486c6aa6aa3efa63cc56880 /mnt/forensic/prog
```

3. To find key words in the file, strings command is run on the binary file. There are many interesting words within the binary. Specially, information about version number, author, help messages, internal function names, command options are listed as above.

```
[root@kamo-r2 root]# strings /mnt/forensic/prog
```

1.2. Program Description

Using Internet search engine (google) with keywords found in the binary as shown below, a tool that named “bmap” is found.

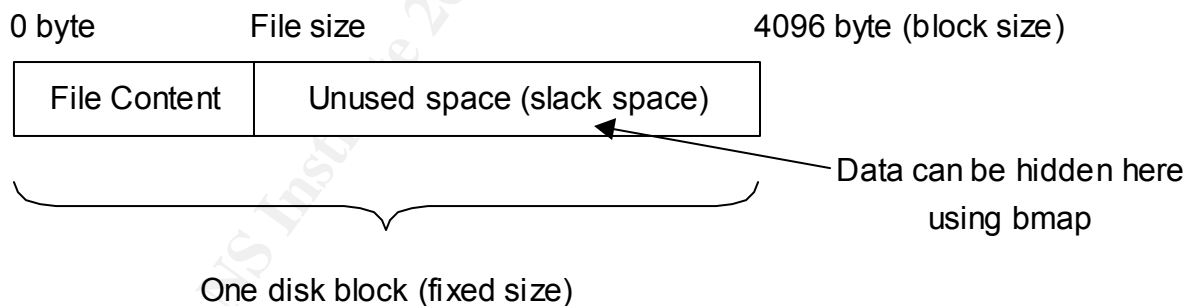


Searching with Internet search engine, further information about bmap is found.

According to the article that posted to Security Focus [1], this tool seems to be a kind of file system manipulation tool. This matches keywords found in the binary such as “display fragmentation information for the file” or “extract a copy from the raw device”.

According to the article that posted to linuxsecurity.com [2], this tool hides/retrieves data into/from slack space that exists in file system. In this article, sample command “bmap --mode slack /etc/passwd” is included. This matches key words within the binary such as “mode” and “slack”.

How bmap works on slack space is well explained in the article [3]. Slack space is a kind of unused disk space. When a file is written to a disk block, rest of disk space is unused. This is slack space. The size of the slack space is difference between block size and file size. Usually, slack space isn’t used. So data can be hidden into slack space. Bmap hides any data into slack space.



These indicate that the binary is a data-hiding tool named “bmap”. Program description is shown below.

What type of program is it?	This program is a data-hiding tool.
What is it used for?	It is used for hiding and retrieving data into/from slack space that exists in file system.
When was the last time it was used?	MAC time tells that last time this program accessed is 2003-07-16 15:12:45 (represented in Japanese local time).

1.3. Forensic Details

One of the forensic footprints is date of compiled. The version number of the program indicates date of compiled. The binary “prog” has version number as shown below.

```
1.0.20 (07/15/03)
```

So, it seems to be compiled on July 15, 2003. It may be represented in the locale of the machine that is used for compiling the binary. So, it may not be represented in Japanese local time. The date of installed may be same.

To determine what libraries the binary uses, ldd command is run on the binary.

```
[root@kamo-r2 root]# ldd /mnt/forensic/prog
not a dynamic executable
```

As a result of ldd command, the binary doesn't need any dynamic library. So the binary doesn't change inode access time of library.

To determine whether or not the binary changes inode access time of host file, stat command is run before and after the binary is executed. This verification process is shown below. In this verification process, the binary hides text data into slack space of a file “/tmp/test.txt” and shows the hidden text data from same file. It's a typical usage of bmap.

```

[root@kamo-r2 forensic]# stat /tmp/test.txt
  File: `/tmp/test.txt'
  Size: 9             Blocks: 8             IO Block: 4096   Regular File
Device: 303h/771d     Inode: 213120          Links: 1
Access: (0644/-rw-r--r--)  Uid: (  0/   root)   Gid: (  0/   root)
Access: 2004-08-01 13:41:45.000000000 +0900
Modify: 2004-08-01 13:37:52.000000000 +0900
Change: 2004-08-01 13:37:52.000000000 +0900

[root@kamo-r2 forensic]# echo "hidden text" | ./prog --mode p /tmp/test.txt
stuffing block 438337
file size was: 9
slack size: 4087
block size: 4096

[root@kamo-r2 forensic]# stat /tmp/test.txt
  File: `/tmp/test.txt'
  Size: 9             Blocks: 8             IO Block: 4096   Regular File
Device: 303h/771d     Inode: 213120          Links: 1
Access: (0644/-rw-r--r--)  Uid: (  0/   root)   Gid: (  0/   root)
Access: 2004-08-01 13:41:45.000000000 +0900
Modify: 2004-08-01 13:37:52.000000000 +0900
Change: 2004-08-01 13:37:52.000000000 +0900

[root@kamo-r2 forensic]# ./prog --mode s /tmp/test.txt
getting from block 438337
file size was: 9
slack size: 4087
block size: 4096
hidden text

[root@kamo-r2 forensic]# stat /tmp/test.txt
  File: `/tmp/test.txt'
  Size: 9             Blocks: 8             IO Block: 4096   Regular File
Device: 303h/771d     Inode: 213120          Links: 1
Access: (0644/-rw-r--r--)  Uid: (  0/   root)   Gid: (  0/   root)
Access: 2004-08-01 13:41:45.000000000 +0900
Modify: 2004-08-01 13:37:52.000000000 +0900
Change: 2004-08-01 13:37:52.000000000 +0900

```

As a result of stat command output comparison, the binary doesn't change inode access time of the host file. So execution of the binary doesn't leave any evidence to inode.

If the binary was executed to hide data into slack space, some meaningful data may found in the floppy disk image. Using bmap utilities compiled on the

analysis environment (described at section 1.4), some files that have slack space are found.

```
[root@kamo-r2 bmap-1.0.20]# ./slacker /mnt/forensic
examining /mnt/forensic/lost+found
examining /mnt/forensic/John
examining /mnt/forensic/John/sect-num.gif
slack bytes: 368
examining /mnt/forensic/John/sectors.gif
slack bytes: 824
examining /mnt/forensic/prog
slack bytes: 972
examining /mnt/forensic/May03
examining /mnt/forensic/May03/ebay300.jpg
slack bytes: 849
examining /mnt/forensic/Docs
examining /mnt/forensic/Docs/Letter.doc
slack bytes: 0
examining /mnt/forensic/Docs/Mikemsg.doc
slack bytes: 0
examining /mnt/forensic/Docs/Kernel-HOWTO-html.tar.gz
slack bytes: 218
examining /mnt/forensic/Docs/MP3-HOWTO-html.tar.gz
slack bytes: 107
examining /mnt/forensic/Docs/Sound-HOWTO-html.tar.gz
slack bytes: 805
examining /mnt/forensic/Docs/DVD-Playing-HOWTO-html.tar
slack bytes: 512
examining /mnt/forensic/nc-1.10-16.i386.rpm..rpm
slack bytes: 394
examining /mnt/forensic/.~5456g.tmp
slack bytes: 480
unformatted capacity: 5529
formatted capacity: 5449
unformatted free: 4724
formatted free: 4652
```

Running bmap command with “--mode slack” option on the all files on the floppy disk image, a file that have meaningful data in its slack space is found. As a result of running file command on the data retrieved from the slack space, it’s identified as a gzip compressed data.

```
[root@kamo-r2 bmap-1.0.20]# ./bmap --mode slack --outfile slack.dat /mnt/forensi
c/Docs/Sound-HOWTO-html.tar.gz
```

```
getting from block 190
file size was: 26843
slack size: 805
block size: 1024
[root@kamo-r2 bmap-1.0.20]# file slack.dat
slack.dat: gzip compressed data, was "downloads", from Unix
```

Uncompressing the data using zcat command, an interesting text is retrieved as shown below.

```
[root@kamo-r2 bmap-1.0.20]# zcat slack.dat
Ripped MP3s - latest releases:

www.fileshares.org/
www.convenience-city.net/main/pub/index.htm
emmpeethrees.com/hidden/index.htm
ripped.net/down/secret.htm

***NOT FOR DISTRIBUTION***
```

This hidden text data is a footprint that proves execution of the binary.

To determine if there is any network connection started by bmap, tcpdump command is run on the analysis machine.

```
[root@kamo-r2 gcfa_assignment]# tcpdump
tcpdump: listening on eth0

0 packets received by filter
0 packets dropped by kernel
```

During executing tcpdump command, the binary is run on the analysis machine as shown below. In this verification process, the binary hides text data into slack space of a file "/tmp/test.txt" and shows the hidden text data from same file. And also it shows help message of its command option. It's a typical usage of bmap.

```
[root@kamo-r2 forensic]# echo "hidden text" | ./prog --mode p /tmp/test.txt
stuffing block 438337
file size was: 9
slack size: 4087
```

```

block size: 4096
[root@kamo-r2 forensic]# ./prog --mode s /tmp/test.txt
getting from block 438337
file size was: 9
slack size: 4087
block size: 4096
hidden text
[root@kamo-r2 forensic]# ./prog --mode w /tmp/test.txt
stuffing block 438337
file size was: 9
slack size: 4087
block size: 4096
write error
write error
write error
[root@kamo-r2 forensic]# ./prog --help
prog:1.0.20 (07/15/03) newt
Usage: prog [OPTION]... [<target-filename>]
use block-list knowledge to perform special operations on files

--doc VALUE
  where VALUE is one of:
  version  display version and exit
  help     display options and exit
  man      generate man page and exit
  sgml     generate SGML invocation info
--mode VALUE
  where VALUE is one of:
  m  list sector numbers
  c  extract a copy from the raw device
  s  display data
  p  place data
  w  wipe
  chk test (returns 0 if exist)
  sb  print number of bytes available
  wipe wipe the file from the raw device
  frag display fragmentation information for the file
  checkfrag test for fragmentation (returns 0 if file is fragmented)
--outfile <filename> write output to ...
--label useless bogus option
--name useless bogus option
--verbose          be verbose
--log-thresh <none | fatal | error | info | branch | progress | entryexit> logging
threshold ...
--target <filename> operate on ...

```

As a result of tcpdump command, the binary doesn't start any network command. To copy data from other network-connected machine to slack space or from slack space to network-connected machine, it's needed to use other network related command such as netcat. It could be why that an rpm file of netcat is found in the floppy disk image. Examples that data is copied from/to slack space using netcat are shown below.

Copy data from other network-connected machine to slack space

```
nc -l -p <port> | prog -p <file name>
```

Copy data from slack space to other network-connected machine

```
prog -s <file name> | nc <ip address> <port>
```

As a result of string command (described at section 1.1), any "leads" that could be pulled out of the file for further investigation (e.g. IP address, user information, etc.) aren't found. It is because that bmap works on only local file system and doesn't need any network connection.

1.4. Program Identification

According to the article [2], distribution site of bmap is shown below.

ftp://ftp.scyld.com/pub/forensic_computing/bmap/

But this distribution site cannot be accessed at the time of writing this report. Searching with Internet search engine, copy of the source code archive of bmap is found. The source code of bmap version 1.0.20 is downloaded from URL shown below.

<http://ftp.cfu.net/mirrors/garchive.cs.uni.edu/garchive/bmap-1.0.20/>

Compiling and installing bmap is simple. The source code archive file is extracted and make command is run at the extracted directory.

```
# tar jxvf bmap-1.0.20.tar.bz2
# cd bmap-1.0.20
```

```
# make
```

The bmap utilities are compiled at the analysis environment, and works as expected. It hides and retrieves data into/from slack space as shown below.

```
[root@kamo-r2 bmap-1.0.20]# ./bmap --mode putslack README < /etc/hosts
stuffing block 2263151
file size was: 6639
slack size: 1553
block size: 4096
[root@kamo-r2 bmap-1.0.20]# ./bmap --mode slack README
getting from block 2263151
file size was: 6639
slack size: 1553
block size: 4096
# Do not remove the following line, or various programs
# that require network functionality will fail.
127.0.0.1      kamo-r2 localhost.localdomain  localhost
```

But, its file size and MD5 hash value don't match the binary file found in the floppy disk image (prog).

```
[root@kamo-r2 bmap-1.0.20]# stat bmap
  File: `bmap'
  Size: 220337      Blocks: 440      IO Block: 4096   Regular File
Device: 303h/771d  Inode: 1125201   Links: 1
Access: (0755/-rwxr-xr-x)  Uid: (  0/   root)   Gid: (  0/   root)
Access: 2004-06-27 01:25:45.000000000 +0900
Modify: 2004-06-27 01:25:43.000000000 +0900
Change: 2004-06-27 01:25:43.000000000 +0900

[root@kamo-r2 bmap-1.0.20]# md5sum bmap
2eb921235e283a5459d155ea497bea0a  bmap
[root@kamo-r2 bmap-1.0.20]# md5sum /mnt/forensic/prog
7b80d9aff486c6aa6aa3efa63cc56880 /mnt/forensic/prog
```

To find key words in the bmap binary file, strings command is run on the binary file compiled on the analysis environment.

```
[root@kamo-r2 bmap-1.0.20]# strings bmap
```

As a result of comparing keywords, unknown binary file is certainly bmap

program.

Key words in the unknown binary file	Key words in the locally compiled bmap program
1.0.20 (07/15/03)	1.0.20 (06/27/04)
newt	newt@scyld.com
prog	bmap
extract a copy from the raw device	extract a copy from the raw device
use block-list knowledge to perform special operations on files	use block-list knowledge to perform special operations on files
bmap_get_block_size	bmap_get_block_size
bmap_map_block	bmap_map_block
bmap_raw_open	bmap_raw_open
mode	mode
slack size: %d	slack size: %d

One of the reasons that they have difference, compile options may be different. The “Makefile” file on the analysis environment is shown below. Program name, version, patch level, build date and author are given by “Makefile” settings.

Extraction from Makefile

```
versioning information
#
PKG_NAME = "bmap"
VERSION = 1
PATCHLEVEL = 0.20
BUILD_DATE = $(shell date +%D)
AUTHOR = "newt@scyld.com"
```

According to the key words, the binary found in the floppy disk image may be compiled with options shown below. And compile option “BUILD_DATE” is automatically set to current date.

```
PKG_NAME = "prog"
AUTHOR = newt
```

Command options are customized in the case of the binary found in the floppy disk image as shown below. It also makes difference between two hash values.

Command options of the binary “prog”

```
[root@kamo-r2 forensic]# ./prog --help
prog:1.0.20 (07/15/03) newt
Usage: prog [OPTION]... [<target-filename>]
use block-list knowledge to perform special operations on files

--doc VALUE
  where VALUE is one of:
  version  display version and exit
  help     display options and exit
  man      generate man page and exit
  sgml     generate SGML invocation info
--mode VALUE
  where VALUE is one of:
  m  list sector numbers
  c  extract a copy from the raw device
  s  display data
  p  place data
  w  wipe
  chk test (returns 0 if exist)
  sb  print number of bytes available
  wipe wipe the file from the raw device
  frag display fragmentation information for the file
  checkfrag test for fragmentation (returns 0 if file is fragmented)
--outfile <filename> write output to ...
--label useless bogus option
--name useless bogus option
--verbose          be verbose
--log-thresh <none | fatal | error | info | branch | progress | entryexit> logging
threshold ...
--target <filename> operate on ...
```

Command options of bmap (locally compiled)

```
[root@kamo-r2 bmap-1.0.20]# ./bmap --help
bmap:1.0.20 (06/27/04) newt@scyld.com
Usage: bmap [OPTION]... [<target-filename>]
use block-list knowledge to perform special operations on files

--doc VALUE
  where VALUE is one of:
```

```
version  display version and exit
help      display options and exit
man       generate man page and exit
sgml      generate SGML invocation info
--mode VALUE
  where VALUE is one of:
  map      list sector numbers
  carve    extract a copy from the raw device
  slack    display data in slack space
  putslack place data into slack
  wipslack wipe slack
  checkslack test for slack (returns 0 if file has slack)
  slackbytes print number of slack bytes available
  wipe     wipe the file from the raw device
  frag     display fragmentation information for the file
  checkfrag test for fragmentation (returns 0 if file is fragmented)
--outfile <filename> write output to ...
--label useless bogus option
--name    useless bogus option
--verbose      be verbose
--log-thresh <none | fatal | error | info | branch | progress | entryexit> logging
threshold ...
--target <filename> operate on ...
```

The other reason is difference of linked libraries' version. And the bmap program is built using dynamic libraries, but the binary found in the floppy disk image is built using static libraries. It's because that bmap utilities are built using dynamic libraries by default.

1.5. Legal Implications

Execution of data-hiding tool such as bmap is legal in Japan. But if he distributed copyrighted material, he violated copyright law[4].

According to copyright law, a person who distributes copyrighted material shall be punished by imprisonment for a term not exceeding one year or a fine not exceeding one million Yen (Article 121bis).

Additional information about Japanese law and distribution of copyrighted material is described at chapter 3.

1.6. Interview Questions

In the interview with John Price, it's necessary to confirm 2 matters. These are about text data hidden in floppy disk and his distribution of copyrighted material.

1. What is your role in this organization? Are you a system administrator?
2. Did you use Linux machine on your job? Did you have administrator right on it?
3. How well do you operate Linux machine?
4. Do you like music? Have you ever downloaded mp3 file from Internet?
5. What did you do at around 15:00 of July 16, 2003 (Japanese local time)?
6. A floppy disk is found on your machine. Does it belong to you? If not, from whom did you get it?
7. Did you know about the program on the floppy disk? Did you install or use it?
8. Did you make the text data hidden in the floppy disk? If not, from whom did you receive it?
9. What can you get from the URLs written in the text data? And did you upload these?
10. Did you give/take information about copyrighted material to/from other person. If so, who is he/she?
11. An rpm file of netcat is found in the floppy disk. Did you install or use netcat? If so, how and why did you use it?
12. Did you store or transport copyrighted material in/to other computer system?

1.7. Case Information

Some advice shown below should be provided to System administrators for detecting if John Price used other computer resources for distributing copyrighted material.

1. To detect whether the binary is in use or has been used on the other machines, it is a simple solution that retrieve slack data using bmap and validate it using file command. For example, file command run on the output of running bmap as shown below. Making shell script may help systems administrators to validate all files in the system.

```
bmap --mode slack --outfile slack.dat <file name>
file slack.dat
```

2. An rpm file of netcat is found in the floppy disk image. There is a possibility that he use netcat to transfer copyrighted material to other systems. To validate netcat command is in use, ps command is run on the system as shown below.

```
[root@kamo-r2 root]# ps -aux | grep nc
root      3571  0.0  0.0 1428  504 pts/2    S   15:32   0:00 nc -l -p 4321
root      3639  0.0  0.1 4576  656 pts/3    S   15:35   0:00 grep nc
```

From some evidence shown below, it's certainly that John Price distributed copyrighted material using organizations computing resources.

1. Text data show below is found in slack space of file "Docs/Sound-HOWTO-html.tar.gz" in the floppy disk image. This text is titled as "Ripped MP3s – latest releases:" and includes some URLs. MP3 files may be downloadable at the URLs included in the text.

The text data hidden in slack space of the floppy disk

```
Ripped MP3s - latest releases:

www.fileshares.org/
www.convenience-city.net/main/pub/index.htm
emmpeethrees.com/hidden/index.htm
ripped.net/down/secret.htm

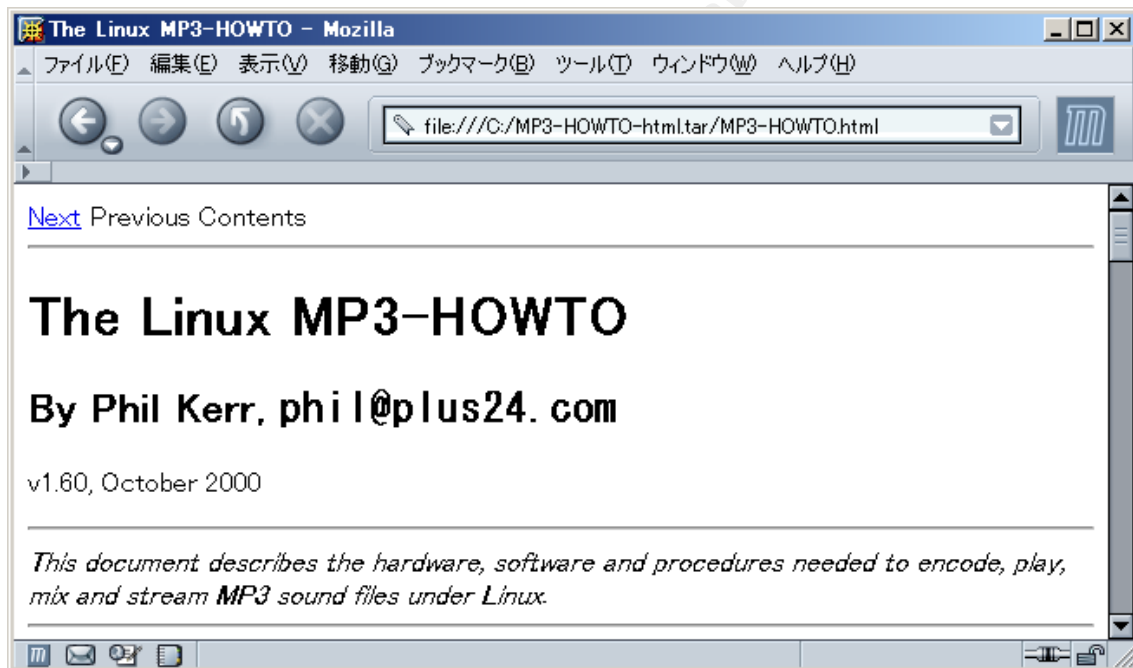
***NOT FOR DISTRIBUTION***
```

2. A Microsoft Word file that has content shown below is found. The signature "JP" at the bottom of the content may mean "John Price". From the content, John Price sent some files to the person who called "Mike".

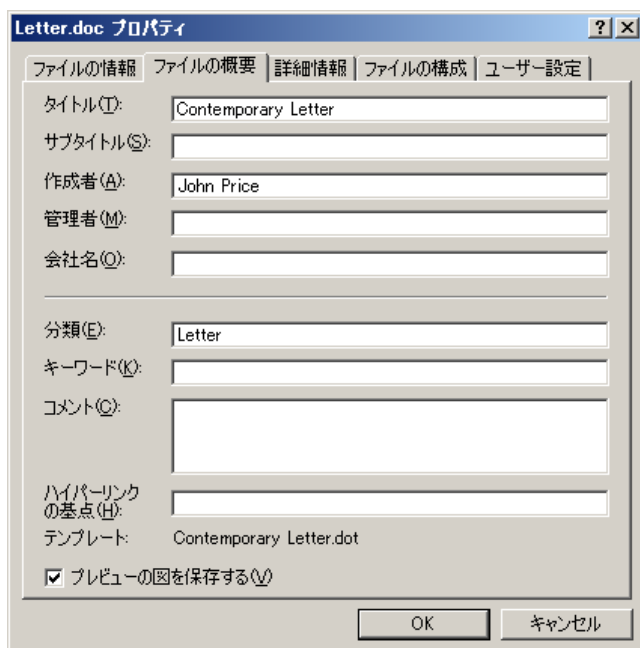
Content of the file named “Docs/Mikemsg.doc”

```
Hey Mike,  
  
I received the latest batch of files last night and I'm ready to rock-n-roll (ha-ha) .  
  
I have some advance orders for the next run. Call me soon.  
  
JP
```

3. An archive file named “Docs/Sound-HOWTO-html.tar.gz” is found. It contains documents that describe how to encode MP3 file on Linux system. John Price may encode MP3 files.



4. The signature “JP” is found in the file named “Docs/Mikemsg.doc”. The directory named “John” is found. And document property of Microsoft Word file named “Docs/Letter.doc” tells that author of the document is “John Price”. These are evidence that the floppy disk belongs to him.



1.8. Additional Information

- [1] Daniel Ridge, “bmap 1.0.17” <http://www.securityfocus.com/tools/1359>
- [2] Anton Chuvakin, “Linux Data Hiding and Recovery”
http://www.linuxsecurity.com/feature_stories/data-hiding-forensics.html
- [3] Jonathan C. Busey, “Data Hiding and Recovery”
<http://www.cs.fsu.edu/~yasinsac/group/slides/busey4.pdf>
- [4] Copyright Research and Information Center, “Copyright Law of Japan”
http://www.cric.or.jp/cric_e/clj/clj.html

2. Perform Forensic Tool Validation

The purpose of this chapter is to validate a forensic tool. The tool named “Sfind” is selected for validation. It’s included in The Forensic Toolkit published by FoundStone, Inc. It’s a forensic tool that scans and lists files hidden in alternative data stream of NTFS file system.

2.1. Scope

The Forensic Toolkit is a collection of file analyzing tools for Windows NTFS file system. It consists of Afind, Hfind, File Stat, Hunt and Sfind. Sfind is a tool that scans and lists data hidden in file system.

NTFS file system has a function called “Alternate Data Stream”[5]. Files can be hidden using this function. For example, an executable file can be hidden into a normal text file “test.txt”. The hidden file can be executed using start command.

```
C:\temp>type c:\windows\system32\ftp.exe > test.txt:stream.exe  
C:\temp>start c:\temp\test.txt:stream.exe
```

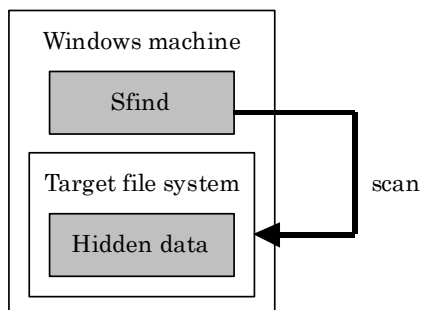
Also, simple text data can be hidden into a normal text file “test2.txt”. The hidden text can be printed using “more” command.

```
C:\temp>echo hidden text > test2.txt:hidden.txt  
C:\temp>more < test2.txt:hidden.txt  
hidden text
```

In this validation, Sfind is run under 4 cases shown below.

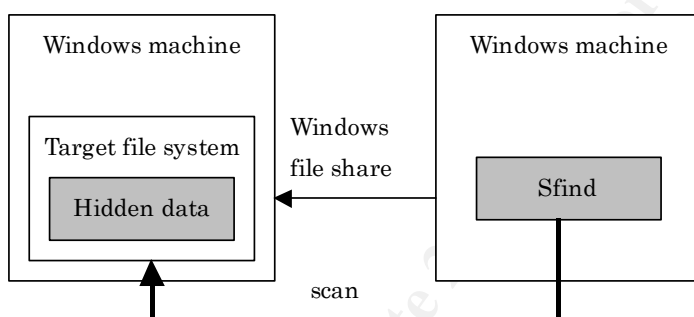
Case 1:

Sfind is run on the Windows machine that has the target file system.



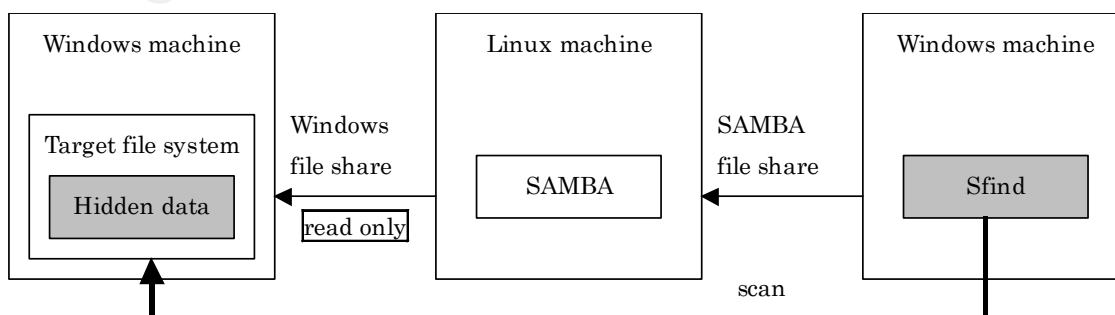
Case 2:

The Windows machine that has target file system has Windows file share. Sfind is run on the other Windows machine.



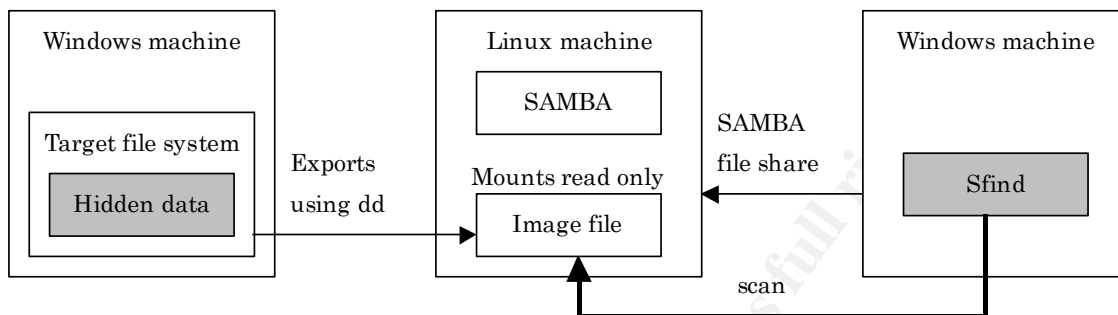
Case 3:

The Windows machine that has target file system has Windows file share. Linux machine mounts the target file system using Windows file share and provides SAMBA file share. Sfind is run on the other Windows machine against Linux machine.



Case 4:

The target file system is exported using dd command as a disk image file. Linux machine mounts the disk image file and provides samba file share. Sfind is run on the other Windows machine against Linux machine.



The purpose of validation held in this chapter is to evaluate ability of this tool. The evaluation points are shown below.

1. Can Sfind find files hidden not only in files but also in directories?
2. Can Sfind files hidden in files or directories if their attributes are set to read-only, archive, system or hidden?
3. Can Sfind find hidden files under 4 cases shown above.
4. Does Sfind make any change to file system?

2.2. Tool Description

Current version of The Forensic Toolkit is v2.0. It was updated at October 31, 2000. It can be downloaded at [6]. It's published by Foundstone, Inc.

Installing Sfind is simple. It's just extracting archive file of The ForensicToolkit and copying binary file "sfind.exe" to specific directory. Other files such as dll, ini, bat, etc. aren't needed.

By default, Sfind scans current folder recursively. If a folder is specified, Sfind scans it. It's also changeable scanning recursively or not. If files are found in alternative data stream, Sfind lists file names of hidden files and host files.

Sfind has very simple command line options shown below.

```
C:\temp>Sfind /?
Sfind v2.0 - Copyright(c) 1998, Foundstone, Inc.
Alternate Data Stream Finder
    Usage - Sfind [path] /ns
    [dirpath]      Directory to search - none equals current
    -ns           Skip sub-directories
    - or /        Either switch statement can be used
    -?           Help
COMMAND PROMPT MUST HAVE A MINIMUM WIDTH OF 80 CHARACTERS
See http://www.foundstone.com for updates/fixes
```

“-ns” is an only option used for skipping sub directories. For example, scanning only “c:\temp” directory (not recursively) is done as shown below.

```
C:\temp>Sfind c:\temp
Searching...
C:\temp
  test.txt:stream.exe Size: 40448
  test2.txt:hidden.txt Size: 14
Finished
```

2.3. Test Apparatus

The validation needs 2 Windows and 1 Linux machines.

Name	OS	Description
target	Windows 2000 Professional Service Pack 4	It has hidden file on its NTFS file system. In case 1, Sfind command is run on it.
forensic	Windows 2000 Professional Service Pack 4	In case 2-4, Sfind command is run on it.
airgap	RedHat Linux 9 Kernel 2.4.20-8 with NTFS support SAMBA 2.2.7a	In case 3, it mounts target's file system using Windows file share. In case 4, it mounts disk image file that has hidden file.
analysis	Windows XP Professional Service Pack 1 Cygwin version 2.416	Standalone computer.

And 1 network switch and some CAT5 cables are needed.

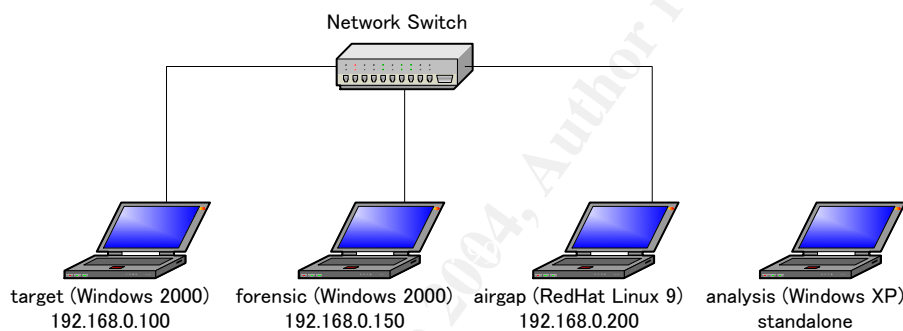
These machines are network connected except for the "analysis" machine. Network status of those machines is shown below.

Machine	Type	Detail
target	Windows file share	Local directory "c:\forensic" is shared as "forensic".
forensic	Windows file share	Shared directory "forensic" on the "target" machine is mounted as "P:\".
		Shared directory "airgap" on the "airgap" machine is mounted as "Q:\".
airgap	Network file system mount using "mount"	Shared directory "forensic" on the "target" machine is mounted as "/mnt/forensic".
	SAMBA file share	Local directory "/mnt/forensic" is shared as "airgap"
analysis	No network connection	

2.4. Environmental conditions

The validation preformed at a local area network (LAN) using a network switch. The LAN doesn't connect to other networks. There is no filtering at the network switch. IP addresses of each machine are statically configured. There is no DNS and WINS server on the LAN.

Name	IP address	Network mask
target	192.168.0.100	255.255.255.0
forensic	192.168.0.150	
airgap	192.168.0.200	
analysis	No network connection	No network connection



2.5. Description of the procedures

To create the target file system with hidden files, automation batch program is created. It makes 5 files and 5 directories, and hides secret text into the files and directories using echo command. Then it sets different attributes (no attribute, read only, archive, system and hidden) to the files and directories. At last, it displays attributes of files and directories just created. This batch program (prepare.bat) is shown below.

prepare.bat

```
mkdir forensic
cd forensic

echo forensic tool validation > test.txt
```

```

echo hidden text > test.txt:hidden.txt

echo forensic tool validation > test_r.txt
echo hidden text > test_r.txt:hidden.txt
attrib +R test_r.txt

echo forensic tool validation > test_a.txt
echo hidden text > test_a.txt:hidden.txt
attrib -A test_a.txt

echo forensic tool validation > test_s.txt
echo hidden text > test_s.txt:hidden.txt
attrib +S test_s.txt

echo forensic tool validation > test_h.txt
echo hidden text > test_h.txt:hidden.txt
attrib +H test_h.txt

mkdir testdir
echo hidden text > testdir:hidden.txt

mkdir testdir_r
echo hidden text > testdir_r:hidden.txt
attrib +R testdir_r

mkdir testdir_a
echo hidden text > testdir_a:hidden.txt
attrib -A testdir_a

mkdir testdir_s
echo hidden text > testdir_s:hidden.txt
attrib +S testdir_s

mkdir testdir_h
echo hidden text > testdir_h:hidden.txt
attrib +H testdir_h

attrib /S /D

```

After running this batch program on the “target” machine, the target file system is created at “C:\forensic”. According to the result shown below, files and directories are created successfully, and secret texts are hidden without error messages. And also, the result of attrib command indicates that attributes of files and directories are correctly set.

```
C:¥>prepare.bat

C:¥>mkdir forensic

C:¥>cd forensic

C:¥forensic>echo forensic tool validation 1>test.txt

C:¥forensic>echo hidden text 1>test.txt:hidden.txt

C:¥forensic>echo forensic tool validation 1>test_r.txt

C:¥forensic>echo hidden text 1>test_r.txt:hidden.txt

C:¥forensic>attrib +R test_r.txt

C:¥forensic>echo forensic tool validation 1>test_a.txt

C:¥forensic>echo hidden text 1>test_a.txt:hidden.txt

C:¥forensic>attrib -A test_a.txt

C:¥forensic>echo forensic tool validation 1>test_s.txt

C:¥forensic>echo hidden text 1>test_s.txt:hidden.txt

C:¥forensic>attrib +S test_s.txt

C:¥forensic>echo forensic tool validation 1>test_h.txt

C:¥forensic>echo hidden text 1>test_h.txt:hidden.txt

C:¥forensic>attrib +H test_h.txt

C:¥forensic>mkdir testdir

C:¥forensic>echo hidden text 1>testdir:hidden.txt

C:¥forensic>mkdir testdir_r

C:¥forensic>echo hidden text 1>testdir_r:hidden.txt

C:¥forensic>attrib +R testdir_r

C:¥forensic>mkdir testdir_a
```

```

C:\¥forensic>echo hidden text 1>testdir_a:hidden.txt

C:\¥forensic>attrib -A testdir_a

C:\¥forensic>mkdir testdir_s

C:\¥forensic>echo hidden text 1>testdir_s:hidden.txt

C:\¥forensic>attrib +S testdir_s

C:\¥forensic>mkdir testdir_h

C:\¥forensic>echo hidden text 1>testdir_h:hidden.txt

C:\¥forensic>attrib +H testdir_h

C:\¥forensic>attrib /S /D
A          C:\¥forensic¥test.txt
A          C:\¥forensic¥testdir
          C:\¥forensic¥testdir_a
A  H       C:\¥forensic¥testdir_h
A  R       C:\¥forensic¥testdir_r
A  S       C:\¥forensic¥testdir_s
          C:\¥forensic¥test_a.txt
A  H       C:\¥forensic¥test_h.txt
A  R       C:\¥forensic¥test_r.txt
A  S       C:\¥forensic¥test_s.txt

```

Case 1:

To evaluate ability of Sfind, a validation batch program is created. The batch program (validation.bat) is shown below. It scans current directory using Sfind command. Sfind command is run with various options such as no option (recursive), “-ns” option (not recursive) and specifying each subdirectories. Then hidden files are retrieved using more command. In the case that Sfind cannot find hidden files and more command can retrieve them, it’s concluded that Sfind have limitation of its scanning ability in that case. On the other hand, in the case that Sfind cannot find hidden files and more command also cannot retrieve them, it isn’t because Sfind have limitation but there is other reason (for example, limitation of SAMBA file share).

Sfind and this batch program are installed on the “target” machine. After

changing current directory to “c:\forensic”, the batch program is run.

validation.bat

```
Sfind
Sfind /ns
Sfind testdir
Sfind testdir_r
Sfind testdir_a
Sfind testdir_s
Sfind testdir_h
more < test.txt:hidden.txt
more < test_r.txt:hidden.txt
more < test_a.txt:hidden.txt
more < test_s.txt:hidden.txt
more < test_h.txt:hidden.txt
more < testdir:hidden.txt
more < testdir_r:hidden.txt
more < testdir_a:hidden.txt
more < testdir_s:hidden.txt
more < testdir_h:hidden.txt
```

Case 2:

Windows file share “forensic” is created on the “target” machine using net share command as shown below.

```
C:\forensic>net share forensic=c:\forensic
```

Using net use command, the shared directory “forensic” is mounted on the “forensic” machine as “p:” as shown below.

```
C:\>net use P: \\192.168.0.100\forensic /user:administrator *****
```

Note: Administrator’s password is sanitized.

After changing current directory to “p:”, the validation batch program (validation.bat) is run on the “forensic” machine.

Case 3:

Using mount command, the shared folder “forensic” is mounted on the “airgap” machine as “/mnt/forensic” as shown below.

```
[root@airgap /]# mount -t smbfs -o  
ro,noexec,noatime,username=administrator,password=*****  
//192.168.0.100/forensic /mnt/forensic
```

Note: Administrator's password is sanitized.

To validate that the hidden files can be retrieved on Linux system, a validation shell script is created. The shell script (validation.sh) is shown below. It shows all hidden data using more command.

validation.sh

```
#!/bin/sh  
more < /mnt/forensic/test.txt:hidden.txt  
more < /mnt/forensic/test_r.txt:hidden.txt  
more < /mnt/forensic/test_a.txt:hidden.txt  
more < /mnt/forensic/test_s.txt:hidden.txt  
more < /mnt/forensic/test_h.txt:hidden.txt  
more < /mnt/forensic/testdir:hidden.txt  
more < /mnt/forensic/testdir_r:hidden.txt  
more < /mnt/forensic/testdir_a:hidden.txt  
more < /mnt/forensic/testdir_s:hidden.txt  
more < /mnt/forensic/testdir_h:hidden.txt
```

SAMBA shared directory "airgap" is created by adding the description shown below to SAMBA configuration file (/etc/samba/smb.conf).

```
[airgap]  
    comment = airgap  
    path = /mnt/forensic
```

Using net use command, the shared directory "airgap" is mounted on the "forensic" machine as "q:¥" as below.

```
C:¥>net use Q: ¥¥192.168.0.200¥airgap /user:smbuser *****
```

Note: SAMBA user's password is sanitized.

After changing current directory to "q:¥", the validation batch program (validation.bat) is run on the "forensic" machine.

Case 4:

Using dd command that is included in Forensic Acquisition Utilities[7], a disk image file of the target file system is created. The “target” machine has a small disk partition “E:”. Running batch program “prepare.bat” on “E:\forensic” directory, target files/directories are created and messages are hidden in those and attributes of the files/directories are changed. After creating the target file system on E: drive, dd command is run on the “target” machine as shown below.

```
dd if=¥¥.¥E: of=forensic.dd
```

The image file “forensic.dd” is copied to the “airgap” machine and mounted using mount command as shown below. The command option “ro” means that the image file is mounted with read only mode. The command option “noatime” means that access time of inode isn’t updated. These options are important for forensic investigation because these protect evidence from being broken. The command option “noexec” means that binary files aren’t executed. It protects analysis system from miss-execution of binary files on image file.

```
[root@airgap root]# mount -o ro,noexec,noatime,loop,uid=500,gid=500 forensic.dd  
/mnt/forensic/
```

To validate that the hidden files can be retrieved on Linux system, validation shell script is created. The shell script (validation2.sh) is shown below. The difference between validation.sh and validation2.sh is only directory path of target files and directories.

validation2.sh

```
#!/bin/sh  
more < /mnt/forensic/forensic/test.txt:hidden.txt  
more < /mnt/forensic/forensic/test_r.txt:hidden.txt  
more < /mnt/forensic/forensic/test_a.txt:hidden.txt  
more < /mnt/forensic/forensic/test_s.txt:hidden.txt  
more < /mnt/forensic/forensic/test_h.txt:hidden.txt  
more < /mnt/forensic/forensic/testdir:hidden.txt  
more < /mnt/forensic/forensic/testdir_r:hidden.txt  
more < /mnt/forensic/forensic/testdir_a:hidden.txt  
more < /mnt/forensic/forensic/testdir_s:hidden.txt  
more < /mnt/forensic/forensic/testdir_h:hidden.txt
```

SAMBA file share is already created and mounted in Case 3. After

changing current directory to “q:¥”, the validation batch program (validation.bat) is run on the “forensic” machine.

Mac time verification:

This test verifies whether or not Sfind makes any changes to file system. To determine what libraries Sfind use, objdump command is run on the Sfind binary on the “analysis” machine as shown below. Objdump command is a command that is usually used on UNIX platform. Objdump command for Windows platform is included in Cygwin binutil package. Command option “-x” means display contents of all headers. Using grep, information related to dynamic link library (dll) is retrieved.

```
objdump -x /cygdrive/c/temp/Sfind.exe | grep -i dll
```

To determine whether or not Sfind changes mactime of target files/directories, stat command is run on the target files/directories before and after execution of Sfind. The batch program (macvalidate.bat) is shown below. Stat and sleep command used in the batch program is usually used on UNIX platform. These commands can be executed on Cygwin platform. After changing current directory to “c:¥temp”, the batch program is run on the “analysis” machine.

macvalidate.bat

```
echo forensic tool validation > test.txt
echo hidden text > test.txt:hidden.txt
mkdir testdir
echo hidden text > testdir:hidden.txt
stat test.txt
stat testdir
sleep 60
Sfind
stat test.txt
stat testdir
```

2.6. Criteria for approval

The expected result of running Sfind is finding all hidden files regardless

attributes of files/directories in all cases. List of hidden files that should be found is shown below.

File or Directory	Name	Attribute	Hidden File Name
File	test.txt		hidden.txt
	test_r.txt	Read Only	hidden.txt
	test_a.txt	Archive	hidden.txt
	test_s.txt	System	hidden.txt
	test_h.txt	Hidden	hidden.txt
Directory	testdir		hidden.txt
	testdir_r	Read Only	hidden.txt
	testdir_a	Archive	hidden.txt
	testdir_s	System	hidden.txt
	testdir_h	Hidden	hidden.txt

From a forensic point of view, forensic tools should not break evidence left on target file system. Specially, attention should be paid not to change mac time of files and directories on target file system. So Sfind also should not change mac time of files and directories. If Sfind uses some dynamic linked libraries (dll), it may change access time of the dlls. If mac times of target files and directories are different before and after execution of Sfind, it means that Sfind breaks evidence left on target file system.

2.7. Data and Results

Case 1:

The result of running the validation batch program (validation.bat) on the “target” machine is shown below. In this case, all files hidden in the files are found regardless their attributes. But, all files hidden in the directories are not found.

```
C:\forensic>validation.bat

C:\forensic>Sfind
Searching...
C:\forensic
```

```
test.txt:hidden.txt Size: 14
C:\forensic
test_a.txt:hidden.txt Size: 14
test_h.txt:hidden.txt Size: 14
test_r.txt:hidden.txt Size: 14
test_s.txt:hidden.txt Size: 14
Finished
C:\forensic>Sfind /ns
Searching...
C:\forensic
test.txt:hidden.txt Size: 14
test_a.txt:hidden.txt Size: 14
test_h.txt:hidden.txt Size: 14
test_r.txt:hidden.txt Size: 14
test_s.txt:hidden.txt Size: 14
Finished
C:\forensic>Sfind testdir
Searching...
Finished
C:\forensic>Sfind testdir_r
Searching...
Finished
C:\forensic>Sfind testdir_a
Searching...
Finished
C:\forensic>Sfind testdir_s
Searching...
Finished
C:\forensic>Sfind testdir_h
Searching...
Finished
C:\forensic>more 0<test.txt:hidden.txt
hidden text

C:\forensic>more 0<test_r.txt:hidden.txt
hidden text

C:\forensic>more 0<test_a.txt:hidden.txt
hidden text

C:\forensic>more 0<test_s.txt:hidden.txt
hidden text

C:\forensic>more 0<test_h.txt:hidden.txt
hidden text
```

```
C:\¥forensic>more 0<testdir:hidden.txt
hidden text

C:\¥forensic>more 0<testdir_r:hidden.txt
hidden text

C:\¥forensic>more 0<testdir_a:hidden.txt
hidden text

C:\¥forensic>more 0<testdir_s:hidden.txt
hidden text

C:\¥forensic>more 0<testdir_h:hidden.txt
hidden text
```

Case 2:

The result of running the validation batch program (validation.bat) on the “forensic” machine is shown below. In this case, all files hidden in the files are found regardless their attributes. But, all files hidden in the directories are not found.

```
C:\¥>p:

P:\¥>validation.bat

P:\¥>Sfind
Searching...
P:\¥
test.txt:hidden.txt Size: 14
P:\¥
test_a.txt:hidden.txt Size: 14
test_h.txt:hidden.txt Size: 14
test_r.txt:hidden.txt Size: 14
test_s.txt:hidden.txt Size: 14
Finished
P:\¥>Sfind /ns
Searching...
P:\¥
test.txt:hidden.txt Size: 14
test_a.txt:hidden.txt Size: 14
test_h.txt:hidden.txt Size: 14
test_r.txt:hidden.txt Size: 14
test_s.txt:hidden.txt Size: 14
Finished
```

```
P:¥>Sfind testdir
Searching...
Finished
P:¥>Sfind testdir_r
Searching...
Finished
P:¥>Sfind testdir_a
Searching...
Finished
P:¥>Sfind testdir_s
Searching...
Finished
P:¥>Sfind testdir_h
Searching...
Finished
P:¥>more 0<test.txt:hidden.txt
hidden text

P:¥>more 0<test_r.txt:hidden.txt
hidden text

P:¥>more 0<test_a.txt:hidden.txt
hidden text

P:¥>more 0<test_s.txt:hidden.txt
hidden text

P:¥>more 0<test_h.txt:hidden.txt
hidden text

P:¥>more 0<testdir:hidden.txt
hidden text

P:¥>more 0<testdir_r:hidden.txt
hidden text

P:¥>more 0<testdir_a:hidden.txt
hidden text

P:¥>more 0<testdir_s:hidden.txt
hidden text

P:¥>more 0<testdir_h:hidden.txt
hidden text
```

Case 3:

The result of running the validation shell script (validation.sh) on the “airgap” machine is shown below. In this case, all files hidden in the files and directories are retrieved successfully.

```
[root@airgap]# ./validation.sh
hidden text
hidden text
hidden text
hidden text
hidden text
hidden text
hidden text
hidden text
hidden text
hidden text
hidden text
```

The result of running the validation batch program (validation.bat) on the “forensic” machine is shown below. In this case, all files hidden in the files and directories are not found.

```
c:\¥temp>q:
Q:¥>validation.bat

Q:¥>Sfind
Searching...
Finished
Q:¥>Sfind /ns
Searching...
Finished
Q:¥>Sfind testdir
Searching...
Finished
Q:¥>Sfind testdir_r
Searching...
Finished
Q:¥>Sfind testdir_a
Searching...
Finished
Q:¥>Sfind testdir_s
Searching...
Finished
Q:¥>Sfind testdir_h
Searching...
```

```
Finished
Q:¥>more 0<test.txt:hidden.txt
指定されたパスが見つかりません。

Q:¥>more 0<test_r.txt:hidden.txt
指定されたパスが見つかりません。

Q:¥>more 0<test_a.txt:hidden.txt
指定されたパスが見つかりません。

Q:¥>more 0<test_s.txt:hidden.txt
指定されたパスが見つかりません。

Q:¥>more 0<test_h.txt:hidden.txt
指定されたパスが見つかりません。

Q:¥>more 0<testdir:hidden.txt
指定されたパスが見つかりません。

Q:¥>more 0<testdir_r:hidden.txt
指定されたパスが見つかりません。

Q:¥>more 0<testdir_a:hidden.txt
指定されたパスが見つかりません。

Q:¥>more 0<testdir_s:hidden.txt
指定されたパスが見つかりません。

Q:¥>more 0<testdir_h:hidden.txt
指定されたパスが見つかりません。
```

Note: The Japanese phrase “指定されたパスが見つかりません。” means that “Specified path is not found.”.

Case 4:

The result of running the validation batch program (validation2.sh) on the “airgap” machine is shown below. In this case, all files hidden in the files and directories are not retrieved.

```
[root@airgap root]# ./validation.sh
./validation.sh: line 2: /mnt/forensic/forensic/test.txt:hidden.txt: そのような
ファイルやディレクトリはありません
./validation.sh: line 3: /mnt/forensic/forensic/test_r.txt:hidden.txt: そのような
ファイルやディレクトリはありません
./validation.sh: line 4: /mnt/forensic/forensic/test_a.txt:hidden.txt: そのような
```

```
ファイルやディレクトリはありません
./validation.sh: line 5: /mnt/forensic/forensic/test_s.txt:hidden.txt: そのような
ファイルやディレクトリはありません
./validation.sh: line 6: /mnt/forensic/forensic/test_h.txt:hidden.txt: そのような
ファイルやディレクトリはありません
./validation.sh: line 7: /mnt/forensic/forensic/testdir:hidden.txt: そのようなファ
イルやディレクトリはありません
./validation.sh: line 8: /mnt/forensic/forensic/testdir_r:hidden.txt: そのような
ファイルやディレクトリはありません
./validation.sh: line 9: /mnt/forensic/forensic/testdir_a:hidden.txt: そのような
ファイルやディレクトリはありません
./validation.sh: line 10: /mnt/forensic/forensic/testdir_s:hidden.txt: そのような
ファイルやディレクトリはありません
./validation.sh: line 11: /mnt/forensic/forensic/testdir_h:hidden.txt: そのような
ファイルやディレクトリはありません
```

Note: The Japanese phrase “そのようなファイルやディレクトリはありません” means that “Specified files or directories are not exist.”.

The result of running the validation batch program (validation.bat) on the “forensic” machine is shown below. In this case, all files hidden in the files and directories are not found.

```
c:\%temp>q:
Q:\>validation.bat
Q:\%forensic>validation
Q:\%forensic>Sfind
Searching...
Finished
Q:\%forensic>Sfind /ns
Searching...
Finished
Q:\%forensic>Sfind testdir
Searching...
Finished
Q:\%forensic>Sfind testdir_r
Searching...
Finished
Q:\%forensic>Sfind testdir_a
Searching...
Finished
Q:\%forensic>Sfind testdir_s
```

```
Searching...
Finished
Q:¥forensic>Sfind testdir_h
Searching...
Finished
Q:¥forensic>more 0<test.txt:hidden.txt
指定されたパスが見つかりません。

Q:¥forensic>more 0<test_r.txt:hidden.txt
指定されたパスが見つかりません。

Q:¥forensic>more 0<test_a.txt:hidden.txt
指定されたパスが見つかりません。

Q:¥forensic>more 0<test_s.txt:hidden.txt
指定されたパスが見つかりません。

Q:¥forensic>more 0<test_h.txt:hidden.txt
指定されたパスが見つかりません。

Q:¥forensic>more 0<testdir:hidden.txt
指定されたパスが見つかりません。

Q:¥forensic>more 0<testdir_r:hidden.txt
指定されたパスが見つかりません。

Q:¥forensic>more 0<testdir_a:hidden.txt
指定されたパスが見つかりません。

Q:¥forensic>more 0<testdir_s:hidden.txt
指定されたパスが見つかりません。

Q:¥forensic>more 0<testdir_h:hidden.txt
指定されたパスが見つかりません。
```

Note: The Japanese phrase “指定されたパスが見つかりません。” means that “Specified path is not found.”.

To determine whether or not dd command have limitation that cannot make complete disk image with alternative data stream, string command is run on the disk image file.

```
[root@airgap root]# strings forensic.dd

(Middle part of the output is omitted.)
```

```

FILE*
forensic tool validation
hidden text
FILE*
forensic tool validation
hidden text
FILE*
forensic tool validation
hidden text
FILE*
forensic tool validation
hidden text
FILE*
forensic tool validation
hidden text
FILE*
hidden text
FILE*
hidden text
FILE*
hidden text
FILE*
hidden text
FILE*
hidden text
FILE*
hidden text

```

(Rest of the output is omitted.)

The result of running strings command shows that the disk image file contains hidden messages. It's supposed that dd command makes complete disk image but NTFS support of Linux kernel cannot mount it completely.

Mac time verification:

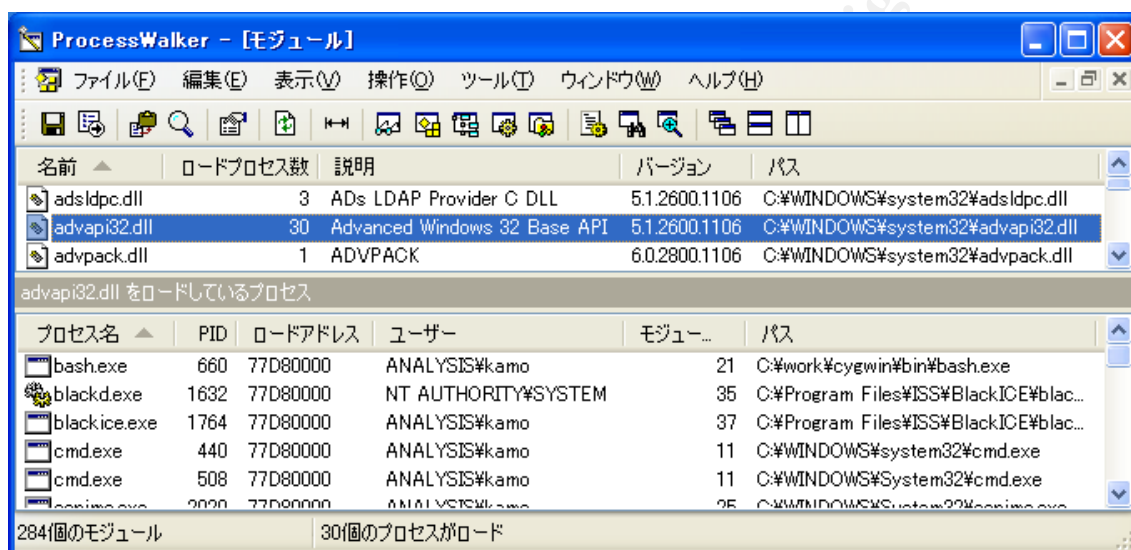
The result of running objdump command on the binary file "Sfind.exe" on the "analysis" machine is shown below. The result indicates that Sfind needs 2 dynamic linked libraries ("kernel32.dll" and "advapi32.dll").

```

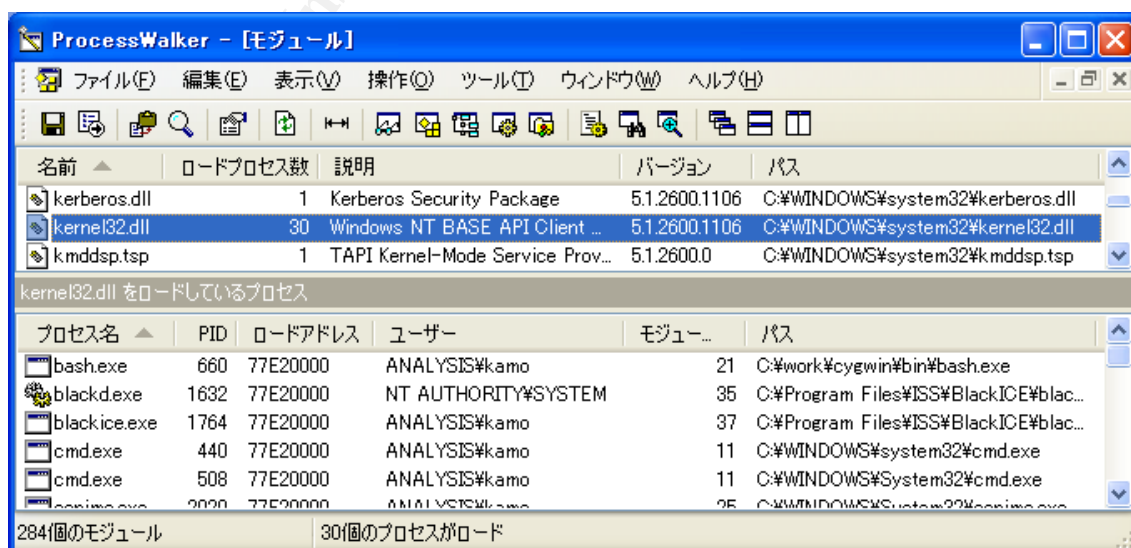
$ objdump -x /cygdrive/c/temp/Sfind.exe | grep -i dll
DllCharacteristics      00000000
vma:                    Hint    Time      Forward DLL        First
      DLL Name: KERNEL32.dll
      DLL Name: ADVAPI32.dll

```

But these dlls are used by many win32 applications. To determine what applications use these dlls, ProcessWalker is run on the analysis machine. ProcessWalker is distributed on [8], and shows processes and dlls running on the machine. It's a very useful tool, but unfortunately it doesn't have English version. On the module screen, processes that use specified dll are listed as shown below.



The screenshot above shows that “advapi32.dll” is loaded by 30 processes such as bash.exe, cmd.exe, etc.



The screenshot above shows that “kernel32.dll” is loaded by 30 processes such as bash.exe, cmd.exe, etc.

The result of running ProcessWalker indicates that these libraries are used by many applications such as cmd.exe, bash.exe (Cygwin), explorer.exe, services.exe, svchost.exe, etc. It indicates that these libraries are already on memory. When Sfind is executed, these dlls files are not accessed because these are already loaded by commonly used other applications (such as Command Prompt, Windows Explorer, etc.). So Sfind doesn't change mac time of any dlls.

The result of running validation batch program (macvalidate.bat) on the analysis machine is shown below. Mac time of the target file is same before and after execution of Sfind. But mac time of the target directory is different before and after execution of Sfind. It means that Sfind breaks evidence left on the target file system.

```
C:\¥temp>macvalidate

C:\¥temp>echo forensic tool validation 1>test.txt

C:\¥temp>echo hidden text 1>test.txt:hidden.txt

C:\¥temp>mkdir testdir

C:\¥temp>echo hidden text 1>testdir:hidden.txt

C:\¥temp>stat test.txt
  File: "test.txt"
  Size: 28          Blocks: 1          IO Block: 1024   Regular File
Device: 40af2294h/1085219476d  Inode: 39581      Links: 1
Access: (0700/-rwx-----)  Uid: ( 1005/kamo)  Gid: ( 513/   なし)
Access: Sun Aug 01 23:32:55 2004
Modify: Sun Aug 01 23:32:55 2004
Change: Sun Aug 01 23:32:54 2004

C:\¥temp>stat testdir
  File: "testdir"
  Size: 0          Blocks: 0          IO Block: 1024   Directory
```

```

Device: 40af2294h/1085219476d Inode: 39582 Links: 2
Access: (0700/drwx-----) Uid: ( 1005/kamo) Gid: ( 513/ なし)
Access: Sun Aug 01 23:32:55 2004
Modify: Sun Aug 01 23:32:55 2004
Change: Sun Aug 01 23:32:55 2004

C:\¥temp>sleep 60

C:\¥temp>Sfind
Searching...
C:\¥temp
test.txt:hidden.txt Size: 14
Finished
C:\¥temp>stat test.txt
File: "test.txt"
Size: 28 Blocks: 1 IO Block: 1024 Regular File
Device: 40af2294h/1085219476d Inode: 39581 Links: 1
Access: (0700/-rwx-----) Uid: ( 1005/kamo) Gid: ( 513/ なし)
Access: Sun Aug 01 23:32:55 2004
Modify: Sun Aug 01 23:32:55 2004
Change: Sun Aug 01 23:32:54 2004

C:\¥temp>stat testdir
File: "testdir"
Size: 0 Blocks: 0 IO Block: 1024 Directory
Device: 40af2294h/1085219476d Inode: 39582 Links: 2
Access: (0700/drwx-----) Uid: ( 1005/kamo) Gid: ( 513/ なし)
Access: Sun Aug 01 23:33:58 2004
Modify: Sun Aug 01 23:32:55 2004
Change: Sun Aug 01 23:32:55 2004

```

2.8. Analysis

In case 1 and 2, all files hidden in the files are found successfully regardless their attributes. But all files hidden in the directories are not found. This means that Sfind has ability to find files hidden in files regardless their attribute. But Sfind hasn't ability to find files hidden in directories. If a malicious attacker hides files in directories, investigator cannot find them using Sfind.

In case 3, Sfind cannot find any files hidden in the files and directories. This

isn't Sfind's limitation because more command attached to Windows also cannot retrieve the hidden files. So, this is supposed to be limitation of SAMBA program on the Linux machine.

In case 4, Sfind cannot find any files hidden in files and directories. This isn't limitation of Sfind because more command attached to Linux also cannot retrieve the hidden files. So, this is supposed to be limitation of dd command or NTFS support of Linux kernel.

In mac time verification, Sfind doesn't change access time of the target file but it changes access time of the target directory. It indicates that Sfind breaks evidence left on the target file system.

From a forensic point of view, Sfind must be able to find all data hidden in alternative data stream and to not break evidence left on the target file system. The first requirement isn't satisfied because it cannot find data hidden in directory. The second requirement isn't satisfied because it changes access time of the target directory.

The conclusion is that using Sfind in forensic situations isn't recommended because investigator may not notice files hidden in directories and evidence is broken.

In addition, SAMBA version 2.2.7a cannot provide file share with alternative data stream. In the forensic situation using Linux air-gap environment such as Case 3, investigator must consider that hidden files in NTFS alternative data stream cannot be found.

And also, NTFS support of Linux kernel 2.4.20-8 cannot mount NTFS file image with alternative data stream. In the forensic situation using NTFS file image such as Case 4, investigator must consider that hidden files in NTFS alternative data stream cannot be found.

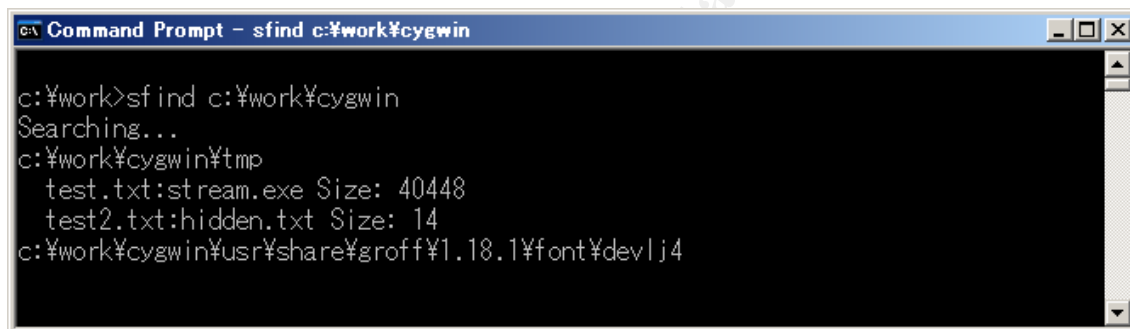
In the forensic situation with possibility that important data is hidden in NTFS alternative data stream, investigation must done stand alone or through windows file share.

2.9. Presentation

Using Sfind, output is shown on command prompt screen as shown below. The output includes file name path and file size of hidden files in the format shown below. There are 2 spaces at the top of the line.

```
<name of host file>:<name of hidden file> Size: <file size of hidden file>
```

And current searching path is shown on the screen, but it is overwritten when current searching path is changed.



```
Command Prompt - sfind c:\work\cygwin
c:\work>sfind c:\work\cygwin
Searching...
c:\work\cygwin\tmp
  test.txt:stream.exe Size: 40448
  test2.txt:hidden.txt Size: 14
c:\work\cygwin\usr\share\groff\1.18.1\font\devlj4
```

If output of Sfind is redirected to a file, all searching path is recorded as shown below. Just under searching path, names of host/hidden files found in NTFS alternative data stream are shown.

```
Searching...

c:\work\cygwin
c:\work\cygwin\bin
c:\work\cygwin
c:\work\cygwin\download
c:\work\cygwin\download\ftp%3a%2f%2fftp.u-aizu.ac.jp%2fpub%2fgnu%2fgnu-win32

(Middle part of the output is omitted.)

c:\work\cygwin
c:\work\cygwin\tmp
  test.txt:stream.exe Size: 40448
  test2.txt:hidden.txt Size: 14
```

```
c:\work\cygwin\usr  
c:\work\cygwin\usr\bin
```

(Rest of the output is omitted.)

So redirection of output to file is preferable as evidence than screenshot because it provides list of all directories scanned by Sfind.

There is no other option to change output format that can be interpreted by other programs.

2.10. Conclusion

The summary of the evaluation result is shown below.

1. Sfind can find files hidden in files, but cannot find files hidden in directories.
2. Sfind can find files hidden in files regardless their attributes.
3. Sfind can find files hidden in files under case 1 and 2. But tool ability cannot be evaluated under case 3 and 4 because of other programs' limitations (SAMBA or Linux kernel).
4. Sfind changes access time of target directories.

From a forensic point of view, Sfind must be able to find all data hidden in alternative data stream and to not break evidence left on the target file system. The first requirement isn't satisfied because it cannot find data hidden in directory. The second requirement isn't satisfied because it changes access time of the target directory.

The conclusion is that using Sfind in forensic situations isn't recommended because investigator may not notice files hidden in directories and evidence is broken.

Using SAMBA 2.2.7a or NTFS support of Linux kernel 8.4.20-8, NTFS file system cannot be mounted completely with alternative data stream. In the

forensic situation with possibility that important data is hidden in NTFS alternative datastream, investigation must done stand alone or through windows file share.

2.11. Additional Information

- [5] Microsoft Corporation, "HOWTO: Use NTFS Alternate Data Streams"
<http://support.microsoft.com/default.aspx?scid=KB;en-us;q105763>
- [6] Foundstone, Inc, "The Forensic Toolkit TM v 2.0"
<http://www.foundstone.com/resources/proddesc/forensic-toolkit.htm>
- [7] George M. Garner Jr., "Forensic Acquisition Utilities"
<http://users.erols.com/gmgarner/forensics/>
- [8] Katsuhiko Yamashita, "ProcessWalker"
<http://www001.upp.so-net.ne.jp/yamashita/product/pw41.htm>

© SANS Institute 2004, Author retains full rights.

3. Legal Issues of Incident Handling

The purpose of this chapter is to answer the questions related to incident handling. The answers are made based on laws in Japan.

3.1. Question A

According to the findings from Part 1 and the assumption in Part 3, John Price was distributing ripped MP3 files on publicly available systems without permission of author and record company. According to the copyright law[9], he has been violating author's right of public transmission (Article 23). He shall be punishable by imprisonment for a term not exceeding one year or a fine not exceeding one million Yen (Article 121bis).

Record companies' rights are also defined as rights of producers of phonograms in the copyright law. So, he also has been violating record company's right of transfer of ownership (Article 97bis).

Case examples in courts are not found, but some people who distributed ripped MP3 files were arrested[10][11].

3.2. Question B

The civil law (called "Minpo" in Japanese)[12] is a main law related to problems on daily life such as dealings, the property, accidents, and families, etc. According to the civil law, when employee commits illegal act with regard to the business, not only the employee but also the employer shall be responsible for compensation for damage (Article 715). The same situation obtains in the case of illegal acts on computer networks[13].

If the employer knows that the employee commits illegal act and leaves it, the employee shall be responsible for compensation for damage because the illegal act is interpreted as with regard to the business of the employer.

So in the case that illegal act such as distribution of copyrighted material is found, employer should order him to stop doing the illegal act. If he doesn't follow it, employer should punish him by dismissal or something.

3.3. Question C

The common opinion of lawyers is that network monitoring is legal under situation shown below[14].

1. The organization has rules about network use and monitoring. Or the organization has general acceptable use policy or secrecy agreement.
2. The organization warns and announces adequately about network monitoring.
3. A person targeted of monitoring has reasonable doubt of violating these rules or policies.

Note: All items (1 to 3) shown above have to be implemented.

There some case examples in courts that monitoring was judged to be legal[15].

In this case, network monitoring should be done to collect further evidence if the company has rules about network monitoring or general acceptable use policy or secrecy agreement. And it's very important that warning and announcement have to be done before beginning monitoring.

3.4. Question D

According to "Law for Punishing Acts Related to Child Prostitution and Child Pornography, and for Protecting Children"[16], a person who distributes child pornography shall be punished (Article 7). He shall be punished with imprisonment for not more than three years or a fine not exceeding three million yen (Article 7). It's three times as severe as distribution of copyrighted material.

And company should also be punished if employee distributes child pornography with regard to the business of the company (Article 11).

There are some case examples that a person who distributed child pornography was punished[17]. But, cases that company was punished owing to employee's distribution child pornography are not found.

In the case that John Price distributed child pornography, company should order him to stop distributing it as in the case that he distributed copyrighted material. But this case is more severe than the case of distribution of copyrighted material. If he doesn't follow it, employer should punish him by dismissal or something.

3.5. Additional Information

- [9] Copyright Research and Information Center, "Copyright Law of Japan"
http://www.cric.or.jp/cric_e/clj/clj.html
- [10] Aichi Police, "Police blotter of hi-tech crime"
<http://www.pref.aichi.jp/police/taisaku/high-tech/jikenbo1-20.html>
- [11] Association of Copyright for Computer Software, "Copyright violation incidents of 1999"
https://www.accsjp.or.jp/news/news_of_1999.html
- [12] houko.com, "Civil Law"
<http://www.houko.com/00/01/M29/089B.HTM>
- [13] Sahoko Kondo, Koji Nagumo, Taketoshi Mishima, "Employer's Responsibility for Employee's Illegal Acts on Network"
<http://www.kisc.meiji.ac.jp/~skondo/ethics/genko000920hp.pdf>
- [14] The Japan Institute for Labour Policy and Training, "Monitoring E-mail"
<http://www.jil.go.jp/jil/kikaku-ga/jirei/19-Q02B2.htm>
- [15] Hisamichi Okamura, "Trend of precedents related to information network in Japan"
<http://www.law.co.jp/cases/netcase.htm>
- [16] Ministry of Justice, "Law for Punishing Acts Related to Child Prostitution and Child Pornography, and for Protecting Children"
<http://www.moj.go.jp/ENGLISH/CRAB/law01.html>

[17] Courts in Japan, “Case H15.10.23 Tokyo local court 4805 violation of law for punishing acts related to child prostitution and child pornography, and for protecting children.”

<http://courtdomino2.courts.go.jp/kshanrei.nsf/webview/0BB9B2415B9B2FC149256E14000B9EE4>

Note: Additional information shown above contains some pages written in Japanese. Machine translation into English is available at URL below.

http://www.excite.co.jp/world/english/web/body/?wb_lp=JAEN&wb_dis=3&wb_co=excitejapan&wb_url=<JAPANESE PAGE URL>

© SANS Institute 2004, Author retains full rights.