Interested in learning
more about security?

SANS Institute
InfoSec Reading Room

This paper is from the SANS Institute Reading Room site. Reposting is not permitted without express written permission.

The Coroners Toolkit - In depth

In this paper I will describe evidence gathering on a Unix system using 'The Coroners Toolkit' version 1.09 (TCT). TCT is freeware. The two types of evidence I will focus on are ephemeral and static evidence. Ephemeral evidence refers to evidence, which generally doesn't last a long time. They are the process and network states of your system. Static evidence refers to all other data that resides on the system in a more or less long-term state. The README provided with TCT describes the package as "...a collection of t...

Copyright SANS Institute
Author Retains Full Rights

AD

Incident Handling & Forensics – GSEC v1.3
The Coroners Toolkit – In depth
By Clarke L. Jeffris

Intro:

First came the reconnaissance, quietly tapping on your defenses looking for a weak spot. Then all at once a barrage of attacks came. Your IDS responded perfectly, the remote loggers caught all the action, and the firewall kept you safe once again. Or did it?
It was during that quiet time between the reconnaissance and the attack where your attacker was able to compromise your security and acquire copious amounts of proprietary company data. The large-scale attack happened to be a cover to provide an easier exit for the attacker. During the compromise the attacker installed several Trojan system files and re-wrote history by modifying your logs to erase his or her tracks. It isn't until after the attacks are over when you begin to suspect something is very wrong. Now you have two choices; do nothing, or investigate. If you chose to do nothing, you could bet that if anyone found out, you will be looking for a new job. However, if you investigate you must be careful to follow all commonly accepted procedures for evidence gathering.

Abstract:

In this paper I will describe evidence gathering on a Unix system using 'The Coroners Toolkit' version 1.09 hereafter referred to as TCT. TCT can be downloaded freely from http://www.porcupine.org/forensics/tct.html. The two types of evidence I will focus on are ephemeral and static evidence. Ephemeral evidence refers to evidence, which generally doesn't last a long time. They are the process and network states of your system. Static evidence refers to all other data that resides on the system in a more or less long-term state.
The README provided with TCT describes the package as "…a collection of tools – some large, some small, some in perl, and some in C – That are all either oriented towards gathering or analyzing forensic data on a Unix system." (TCT README) For the purposes of this paper I will not attempt to provide any training on the analyses of data. Instead I will be focusing on the collection capabilities of TCT and only pointing out and describing the tools for data analyses where appropriate. The GSEC paper "The Coroners Toolkit: A Handy Suite of Utilities " by Mike Wagner provides a general usage of TCT. Although this paper covers what is probably the most common usage of the suite. I will attempt to go further and describe some of the extra functionality that Dan Farmer and Weitze Venema, the authors, provided the user in this robust suite of forensic utilities. As with any software package there are pros and cons associated with it, which I will attempt to address some of these as well.

The Coroners Toolkit - Overview

What is forensic computing? It is the "Process of identifying, preserving, analyzing and presenting digital evidence in a manner that is legally acceptable in any legal proceedings (i.e. a court of law)." (Robert McKennish cited by Steve Romig - Forensic Computing slide 5) When gathering evidence it is imperative to do so in a manner such that the evidence results in the least amount of change and when change occurs you must be able to account for the changes. (Forensic Computing, slide 6) TCT takes this into account when gathering its data. TCT is composed of four different primary tools. Grave-robber, unrm and lazarus, and mactime. There are also a small collection of tools written in C and perl, however I will not be covering them

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

since most of the data collection is performed by the four tools mentioned. In Mike Wagner's GSEC paper he appropriately lists and describes each of these extra utilities. Grave-robber uses a variety of perl modules and according to the README document for it,

> *"It was explicitly designed to never send commands directly to the Unix command shell for parsing (to avoid nasty side effects of meta characters, among other things) and it also logs all shell commands and what time they were run at." (Grave-robber README)*

This design allows for the minimal change and maximum accountability for what caused the change. Grave-robber structures its evidence collection based on an Order of Volatility. By looking at higher volatility items first, such as memory and active processes, grave-robber will have a better chance of collecting it before either the memory is over-written or the process dies. TCT, through grave-robber, thus has the ability to gather a large quantity of valuable ephemeral evidence.

Mactime is a tool used for time stamp evidence gathering. MAC stands for **M**time, **A**time, and **C**time. Mtime is the last time the file was modified, atime is the last time the file was accessed, and ctime was the time the file status was changed; meaning permissions or owner. These are the three time stamps given to a file and are considered ephemeral in their nature since "…almost any operation (reading, writing, etc.) other than the stat() system call … will change them…"(MACTIME README). Grave-robber can be passed the '-m' flag which will cause it to collect these time stamps from all files it examines. Mactime can also be run alone, but it is generally used by grave-robber first. After grave-robber is run however, mactime can be run alone to search for a particular date in its most general use. What occurs is all files falling under a particular date which had their MAC times changed will be dumped. It should be noted that any decent cracker will modify these times to suit their purpose thus rendering this tool useless. It is, in any sense, a good tool to use.

The third and fourth tool in TCT is unrm and lazarus. A mandatory and standard operating procedure of forensics is to create a bit-by-bit copy of a hard drive. Once this is done dumpster diving can commence. This is where unrm is used. It gathers all data from the unused portion of the hard drive. Which is to say that "…if you have a 10 GB disk and you're currently using 2, unrm would generate 8 gigabytes of data." (README in /docs). It then places the data into a file for later analysis. Lazarus takes the data collected by unrm and attempts to organize it in a human readable fashion. Passing lazarus the –h switch causes it to create an HTML clickable mapping of the data with a legend of the various symbols used in the map.

Application Usage grave-robber:

Grave-robber, as described by Mike Wagner;

> *"…is a file walking tool that runs various perl modules, in the lib directory, collecting the basic system information and saving various files necessary to interpret that data." (Wagner)*

Issuing the following command performs the execution of grave-robber in its most general form:

```
# ./grave-robber
```

When run as root in this default mode, grave-robber will run over the entire system and attempt to capture files and process information that are not available to normal users. (grave-robber README) The first thing that grave-robber does is to examine tools which you may need to use while grave-robber is walking the system. While this is occurring you will see output similar to the following:

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

```
Starting preprocessing paths and filenames on hotspot.aoml.noaa.gov...
      Processing $PATH elements...
      /usr/local/sbin
      /usr/sbin
      /sbin
      /bin
      /sbin
      /usr/bin
      /usr/sbin
      /usr/local/bin
      /usr/local/sbin
      /usr/bin/X11
      /usr/X11R6/bin
      /usr/local/mpi/bin
      /root/bin
              Processing dir /home/jeffris/tct-1.09/bin
              Processing dir /etc
              Processing dir /bin
              Processing dir /sbin
              Processing dir /dev
```

Finished preprocessing your toolkit... you may now use programs or examine files in the above directories

At this point depending on the size of the drive being examined, you could be in for a long wait. Grave-robber will continue walking the file system and gathering data for analysis.

There are, of course, many options that one could pass to grave-robber to cause it to perform other actions in addition to or instead of the default file system walk. The "General" (grave-robber source) options available to grave-robber include:

-b body	'This creates a body file for grave-robber. The body file becomes the MACTime database'
-c corpse_dir	'When the system is a "corpse" and the drive is mounted in a directory, this directory gets prepended to everything for example: -c /foo. This would cause grave-robber to look at /foo/home/someone/revenge.txt rather than /home/someone/revenge.txt as it would on a live system.'
-d dir	'This allows the user to override the default data storage directory which is normally located in $DATA/hostname with in the TCT working directory. Handy if you are moving the data to a secure location.'
-e file	'If the user wanted to redirect any errors sent to stderr to a file this is where they would do it. Since it is necessary to account for any changes to the evidence or problems encountered, this flag documents any errors grave-robber may encounter.'
-o OS_type	'If the user is using the –c option then this flag is mandatory. It specifies what kind of corpse grave-robber will be dealing with.'
-v verbose	'grave-robber will send everything it sees to stdout. Useful if you want to see what grave-robber is doing at any given time.'
-D debug	'The authors highly recommend against using this flag.'

(grave-robber source code)

Some of the following "Micro Data Collection" (grave-robber source) options require a live system in order for them to be used.

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

-F ‘This flag enables grave-robber to copy certain files it comes across as it walks the file system. It uses pattern matching to perform this action. These patterns are located in the $conf_pattern variable which is set in coroner.cf. The patterns mostly consist of *.cf, *.conf, etc. The flag also implies the –m flag.’

-i ‘This flag collects all dead inode data’

-I ‘This flag requires a live system. It attempts to capture running processes based on their inodes/proc information.’

-l (sic) ‘This flag too requires a live system. It is referred to as the “look@first” flag. After processing the path it tells tells grave-robber to look at a particular directory first.’

-m ‘Capture MACtime data and run lstat()’

-M ‘This flag does md5’s on all files. It also implies the –m flag.’

-O ‘This flag requires a live system. It saves files which are open but have been deleted from the disk i.e. they are in memory. This is an excellent command if there is an unknown process running that you want to examine.’

-p ‘Using pcat grave-robber copies the process memory to file. The authors make a note in the source code to warn that some systems have trouble with this command. It requires a live system.’

-P ‘This command flies in the face of general evidence gathering procedures and runs the process commands on the live system. These are commands such as ps, lsof, df, etc.’

-s ‘Like the –P option this to does not do well for gathering evidence without risking change. However, both are needed at times. This option runs the general shell commands on the host. Including network and host information gathering, such as netstat, df, chkconfig, etc.’

-S ‘This option saves files listed in the file “save_these_files” in the conf directory’

-t ‘This flag gathers trust information. This could be useful for locating potential security breach points or other potentially compromised computers.’

-V ‘The authors note this flag as “do some mucking around in /dev…”’

(grave-robber source)

The options provided under the micro data collection will provide a wealth of data but it will take along time for the collection to proceed. There are three other options that are available. They are listed under the “Macro Data Collection” (grave-robber source).

-f ‘Fast/quick capture. This flag avoids the file system walk, and it doesn’t run any MD5’s. It does however imply the –P, -s, and –O flag.’

-n ‘This is equivalent to running simply ./grave-robber. It is the default option. This sets the following flags:

-I,-I,-m,-M,-P,-s,-t,-l,-O,-F,-S,-V’

-E ‘This flag is equivalent to running a vacuum cleaner on your system. It will collect everything it can including dangerous stuff. This is to say that it runs the default options with the additional –p flag.’

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

These are all the options available to grave-robber. Many options can be changed simply by modifying the "coronor.cf" config file.(grave-robber source). When run, grave-robber creates for itself two logs. The error and command log which remain in the TCT main directory. It also creates a data directory. In that directory there will be a symbolic link to a hostname-subdirectory containing all the results of the grave-robber run. Within this subdirectory will be located several files and directories which will be of utmost importance to the investigator. There will be a command_out subdirectory, which contains the output from most of the commands, which grave-robber runs (grave-robber README). From a run on my system some of the resulting output is as follows:

```
From the ps command output file:
24382 ?        00:00:00 gnome-terminal
24385 ?        00:00:00 gnome-pty-helpe
24386 pts/0    00:00:00 bash
24414 pts/0    00:00:00 su
24418 pts/0    00:00:00 bash
24464 pts/0    00:00:00 grave-robber
```

Another directory, conf_vault, is an archive of "…all the files that the grave-robber found of interest."(grave-robber README) These include config files, critical files, directories, etc. What gets stored here can be controlled by the files "save_these_files", "coronor.cf", and "grave-robber.cf". There is also an HTML file that indexes all files in that folder. Figure 1. is a screen shot of the index file listing some of the files found on a Linux server. As you can see it is listing many of the logs, which can be found on a given server.

The directory structure of conf_vault keeps the files grave-robber finds in the same structure it found them. This is to say, if grave-robber found /sbin/initlog on the server, then it would be stored as $TCT/data/conf_vault/*sbin/initlog***.** Also in this directory are three directories, which will contain files of processes currently running on the machine. They are icat/, removed_but_running/, and proc/. icat/, contains files of the processes, which were running but still located on the disk. Since they were still on disk grave-robber runs md5's on them and stores them in the same directory. As an example of what the directory would look like, here are a few of the files and their associated md5 files which were listed in my directory:

```
1.out_2002_01_16_09:42:45_-0500
5.out_2002_01_16_09:42:45_-0500
1.out_2002_01_16_09:42:45_-0500.md5
5.out_2002_01_16_09:42:45_-0500.md5
```

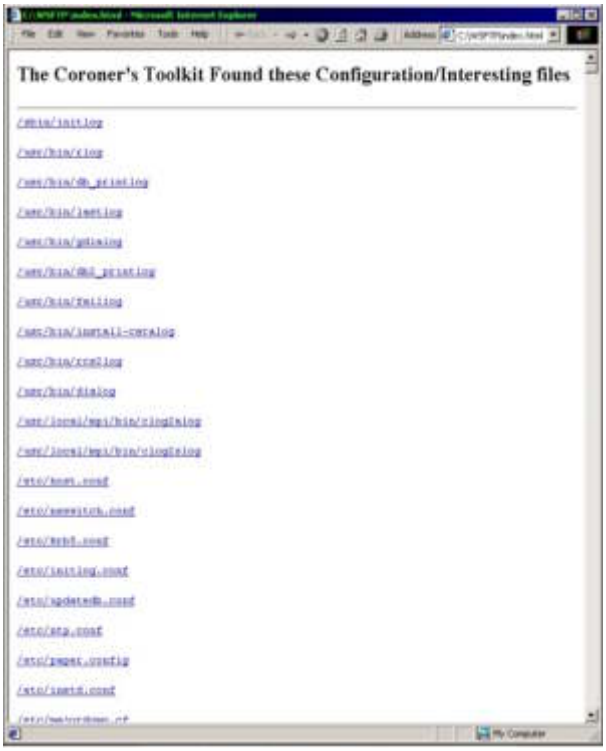

Fig. 1

While the main file contains the data, the md5 file contains the cryptographic signature, directory, and file name of the file for which this key belongs.

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

It looks like this:
495067789c278849c047f338c87a4db0/home/jeffris/tct-1.09/data//myputer.mydomain.somedomain/icat/1.out_2002_01_16_09:42:45_-0500

The proc/ folder looks exactly like the icat folder since the processes are also resident on the disk. However, the removed-but-running/ directory does not contain md5's because the processes, while running, are no longer present on the disk. What remains is simply the process. This directory listing is very similar to the two above with the exception that it does not contain any md5 files.
In the hostname directory there is a directory called user_vault/. This directory contains files pertaining to trust relationships between computers, command history of users, and cryptographic keys as setup by a user. Grave-robber searches the following files for this information:
.forward, rhosts, history, .pgp/, .ssh/, and .ssh/*

Typically these will contain information, which could lead the investigator to other computers, which may have been compromised.
For each run of grave-robber it creates two files. 'Coroner.log' and 'error.log'. Both of these files will be useful for tracking exactly what grave-robber did on a particular run. An example of my coroner.log file follows:

```
2002/01/09 11:33:50 -0500 PIPEFROM_CMD /bin/hostname
2002/01/09 11:33:50 -0500 PIPEFROM_CMD /bin/df /proc
2002/01/09 11:33:50 -0500 PIPEFROM_CMD /bin/uname -s
2002/01/09 11:33:50 -0500 PIPEFROM_CMD /bin/uname -r
2002/01/09 11:33:50 -0500 PIPEFROM_CMD /home/jeffris/tct-1.09/bin/md5 /sbin/arp
```

Each line shows the date, time, and command that grave-robber ran. The error log consists of the names of files, which grave-robber could not find or the error grave-robber received when a command either timed out or didn't work properly. Here's an example:

```
Timeout: aborting command `/home/jeffris/tct-1.09/bin/md5'' with signal 9
Can't open /dev/fd/7 via opendir (in process_dev_dir())
/bin/cp: cannot stat `/proc/1128/exe': No such file or directory
/bin/cp: cannot stat `/proc/2/exe': No such file or directory
```

It's easy to see that grave-robber can be a very powerful tool in the hands of an experienced professional. Standard methodology, completeness, and careful procedure is the key to evidence gathering. Jim McMillian stated in his GSEC paper:

> *"Following a standard methodology is crucial to successful and effective computer forensics. Just as professional programmers use a through programming methodology, computer forensic professionals should use a through investigative methodology."(Importance of a standard Methodology in computer forensics)*

Grave-robber makes every attempt to follow a standard methodology in its capture of evidence. As documented, grave-robber attempts to run the fewest commands in the shell on a live system, to avoid evidence disturbance. It documents everything it does and cryptographically signs all files it dealt with. For initial evidence gathering, grave-robber will do the job fine.

Application Usage MACTime:

MACTime is a program used to collect data on file creation times, modification times, and meta-data times. Its primary role is in support of grave-robber. Grave-robber uses MACTime at run-

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

time and stores the data collected in a database. The database file is located in TCT root and is named 'body'. A second file located in the root named 'body.s' contains the attributes of all SUID files. (grave-robber README)
After the database is created and grave-robber is finished with its file system walk, MACTime can be run in a stand-alone manner to extract information from the database. To do so the user types:

```
# ./mactime 12/28/2001
```

"It will dump to stdout all the files that had their MAC times changed since then…." (MACTime README.) An example of what is returned is as follows:

```
[…]
Dec 31 01 23:58:00  1119477 m.c -rw-rw-rw- jeffris  aoml    /web/daps_hist/311201.daily
Jan 01 02 00:01:04    12230 m.c -rw-rw-r-- root     root    /var/lib/tripwire/report/myfile.twr
Jan 01 02 04:01:03    12230 m.c -rw-rw-r-- root     root    /var/lib/tripwire/report/myfile.twr
Jan 01 02 04:02:00        0 ..c -rw-r--r-- root     root    /var/log/netconf.log.2
[…]
```

If the goal was to simply extract MAC times from files then grave-robber could be passed the –m flag as described earlier. This would cause grave-robber to walk the file system collecting MAC times or a specific file system could be passed for collection.
As with grave-robber, mactime can be passed several flags which I will describe.

-b [file]	This flag is used if grave-robber was told to create a different body file other than the default 'body' file.
-B [file]	By default the output of mactime will go to stdout. This flag tells mactime to redirect the output to a particular file.
-d [directory]	When running mactime alone a directory can be passed to it. This instructs mactime to walk this directory and collect data. It does not however, create a standard database.
-D	This is the debug flag. Like grave-robber, the authors advise against the use of this flag. It produces massive amounts of output.
-f [filename]	When this flag is combined with the HTML option, mactime will flag all files listed in 'filename' for listing in a different color.
-g	The standard group file on a Unix system is /etc/group. This flag allows mactime to look in another file for group information.
-h	This flag creates HTML output. At the moment however this is only in the experimental stages of development. As noted by the authors in the source code.
-l	A mildly functioning takeoff from last. This flag looks at the user log in time and sorts off of that. It formats the list in a manner similar to last.
-n	This flag takes a normal data output, sucj as one provided by the date function, rather than the short version. This version looks like: "Wed Jan 24 8:52:23 EDT 2002"
-p	In Unix, the usernames and passwords are sometimes stored in /etc/passwd. This file, similar to –g, allows for an alternate passwd file to be looked at.

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

-R	This flag causes the file system walk to move recursively through the directories.
-s	When used together with the HTML flag, this flag marks all SUID and SGID files in a different color.
-t	This flag, according to the authors, uses time machine format. This format sets the time stamp output to long form.
-u [user]	This flag when used with the HTML flag sets all files belonging to 'user' in a different color.
-v	Outputs verbose content.
-y	This flag prints the year first. This is so that it will "avoid euro/US ambiguity". The normal output is MM/DD/YYYY, this output is YYYY/MM/DD.

(mactime source code)

If caught in enough time, the time information collected by mactime could be a great asset to the evidence collector. However, it should be noted that these times are extremely ephemeral. Typing in a single command will change a time somewhere. Extreme caution when collecting evidence cannot be stressed enough. It is the single most important factor and will determine whether the evidence can be used in court or not.

Application Usage unrm and Lazarus:

Unrm:

Unrm is a utility program used to recover unallocated blocks on a raw Unix file system. The program is generally used together with Lazarus as input. To run unrm there are a few things that must be known first. If you wanted to collect the data from a particular file system, say /home/someone, you need to know how large it is and how much unallocated space there is on that file system. To do this, the recommended manner is:

```
# df /home/someone
```

The result will look something like:

```
File system          1k-blocks      Used      Available Use% Mounted on
        /dev/hdb1              5036284    585276   4195176    13%     /home
```

At this point unrm is run to collect the unallocated blocks of data on that file system. It is imperative that the data you collect is not stored on the same file system. Doing so will almost certainly overwrite the data you are trying to collect. Instead, use a different file system and use unrm in the following manner:

```
# ./unrm /dev/hdb1 > /some/other/file system
```

At the moment unrm is limited to two different file systems. Ext2 on Linux and ufs on Unix or BSD. Unrm has several flags which can assist with the data collection, they are:

-b	This flag works with "*file systems that consist of fragments, don't insert null-byte padding to preserve logical block alignment in the output. This option is a no-op with the Linux ext2fs file system, where logical blocks and fragments have the same size.*"

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

-e	This flag works in the same manner as dd. It will copy every block.
-f fstype	This flag is used to specify the fstype. As previously stated, unrm only handles two fstypes at the moment.
-v	This turns on verbose mode and sends it to stderr.
-V	This turns on verbose mode and sends it to stdout.

(mactime source)

The device, which the program is to be run on, must be provided after these flags. There is also a start-stop option that can be sent which specifies a starting block and a stopping block. By default, unrm will run on the entire file system.

Lazarus:

"Lazarus – tries to revive things that have died and gone into the binary spirit world… deleted files, data in memory, swap, etc." (Lazarus Source code) Lazarus examines the data collected by unrm and if the data is readable and recognizable, then it is displayed. When used with the '-h' flag, Lazarus formats the data so that it can be read using an Internet browser. It should also be noted that running Lazarus can take a long time since it has to read over what was collected by unrm. Depending on the amount of data collected by unrm one can reasonably expect to wait twice as long for lazarus to complete than they waited for unrm to complete. To run lazarus and have the output go to HTML format enter the following:

```
# ./lazarus -h /some/otherfilesystem/the_file_created_by_unrm
```

As with the other utilities in TCT, lazarus too as flags which it can be passed to achieve certain ends.

-l	This flag processes the data in byte style. This means that the data gets processed one byte at a time rather than block by block.
-b	This instructs lazarus not to write unrecognizable binary blocks. It does write them by default.
-B	This doesn't allow any binary blocks to be written. They are by default.
-d	Debug. As before, the authors don't recommend this.
-h	As I explained before, this flag writes data to HTML. Lazarus also writes three files which are necessary to view the data. They are filename.HTML, filename.menu.HTML, and filename.frame.HTML. Where filename is the name of the unrm file passed to lazarus. The file that should be passed to the browser is the filename.frame.HTML file. These will be located in the same directory as the 'filename' is located unless the –H flag is passed. In which case, the user will specify the directory to write the HTML files to.
-H directory	Write the HTML files to 'directory'.
-D directory	This flag instructs lazarus to write the block files to a specific directory.
-t	Similar to –b, by default, lazarus will write unrecognizable text blocks. This instructs it not to.
-T	Similar to –B, this flag instructs lazarus not to write any text blocks.
-w directory	This instructs lazarus to write all HTML code to 'directory'. This flag must be used with –h.

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

Figure 2 is a screen shot of what lazarus presents in HTML.

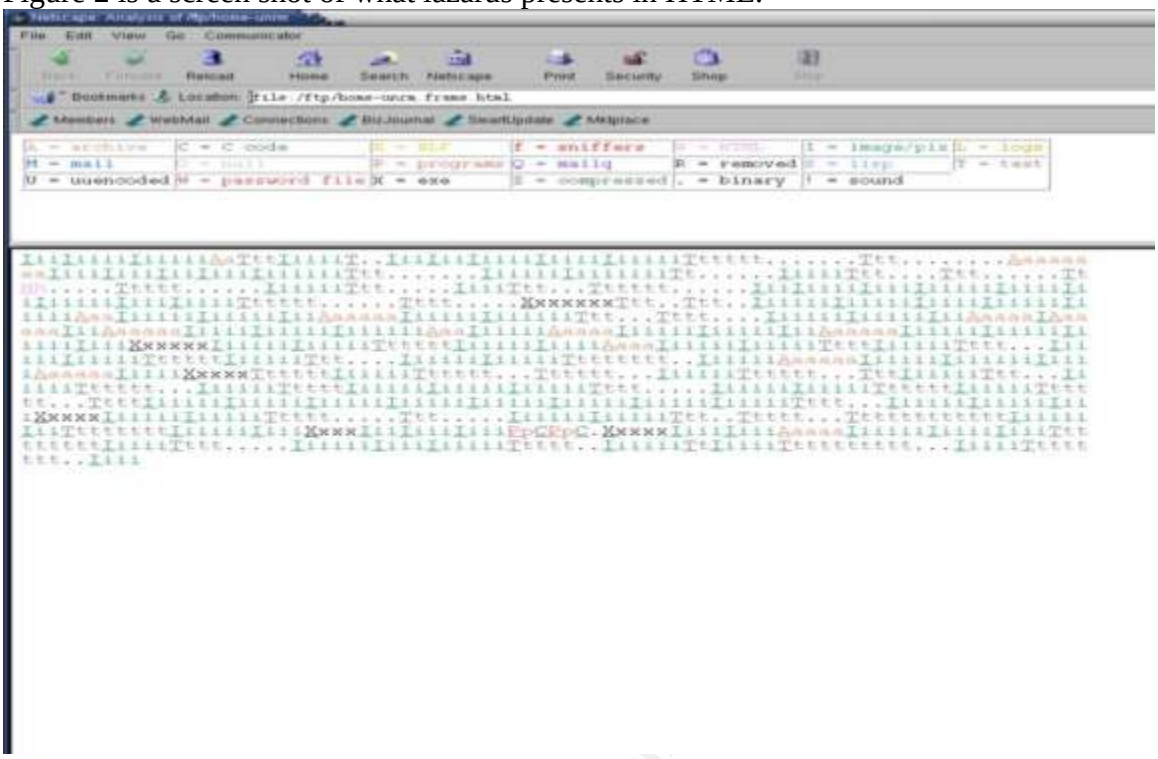

Fig. 2

As a supplement to the data map, lazarus provides a legend, located at the top of the figure, to assist with the data analysis. Each individual character in the map represents a block of data. The block size was determined when unrm was run. The mapping above represents blocks of 1K in size. If the data read in from unrm matches these types of data, it will be displayed in the mapping.

The alternative to passing the '-h' flag is to allow lazarus to run without creating any additional html files. In this manner lazarus output the data in two forms. The first is a block form where it places each block of information into files and names them according to block location. The other form is a map file which contains a list of characters that represents the contents of the block file.(lazarus README) All block file names follow a convention of 'blocknumber.type.txt' or 'blocknumber.type.(jpg,gif,png,etc)' where type could be any character listed in the mapping. The mapping is the same as the legend in Figure 2 above. Since it is expected that these blocks will be read at some later date, in a text reader, they are given the '.txt' extension. By doing this the files won't be interpreted as potentially harmful code by a virus scanner.(same) There is one advantage to using this method of data analysis; it being the case where the data being sought is known. For instance, if you were looking for a C code file and knew something about that file. Then the most appropriate course of action would be to search the data blocks with some keywords. For example:

```
# egrep -l 'keyword1|keyword2|…' blocks/*.txt > filelist
```

A short note on the preservation of evidence would be to say that when lazarus recovers data blocks from the drive, it is in no way modifying the original data. It is instead copying it and breaking the copy into blocks for reading.(lazarus README)

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

The Pros and Cons of TCT

At this time I'd like to cover the advantages and disadvantages of TCT and its use. I will cover the tools in the same manner that I covered their use.

Grave-robber

Pros:

- The greatest advantage for grave-robber is that it attempts to automate the forensic evidence gathering process as much as possible. Instead of fumbling around the operating system trying to remember what you did last and potentially disturbing evidence, the use of grave-robber eliminates most of the worry.
- Grave-robber, by default collects Mactime. This is good because it minimizes the amount of wait time for collection. Since time stamps are very ephemeral they are liable to change before the completion of grave-robber.
- Grave-robber also attempts to collect as much ephemeral data as possible. If there happens to be an unauthorized process running, it will be collected for later examination. This type of collection also provides the investigator a snap shot of the system state, which can be helpful in analysis.

Cons:

- The catch-22 to this tool, and any other like it, is that it needs to be run on a live system in order to collect some of the evidence. This opens the investigator up to the potential of disturbing evidence.
- While grave-robber attempts to avoid running commands on the shell, it still has to write to the files system and run some commands.

Mactime

Pros:

- By virtue of its purpose, it provides the investigator with very useful information regarding the files it's looking at.
- Mactime is usually run by grave-robber. When run under grave-robber, it creates a searchable database file.

Cons:

- The ease in which mactime can be changed for a file, almost makes this utility useless.
- The slightest modification to the file system has the potential to change a mactime on a file located several directories deep, rendering this utility useless.

Unrm

Pros:

- Unrm copies the ever-important un-used blocks of data located on a hard disk.
- Unrm is fairly straightforward to use.

Cons:

- If you aren't interested in analyzing the free blocks only, then unrm isn't for you. Use dd.

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

- Unrm is not run as part of grave-robber and will have to be run separately.
- There is the potential to create an extremely large image file. An example would be; given a "10 GB disk and you're currently using 2, unrm would generate 8 gigabytes of data."(TCT README)

Lazarus

Pros:

- The html flag becomes extremely useful for visualizing the data after it has been dumped.
- Lazarus uses the image created by unrm to create a data mapping for analysis.
- Lazarus can also ream images from RAM or swap.

Cons:

- Given the various flags provided by lazarus I couldn't find a con to comment on. However, I'm sure there are others who can find many.

Conclusion:

In most court cases it is more difficult to set a precedent than it is to use tried and tested tools. TCT is a tried and test tool, which, even though it is old by computer standards, it is most likely being employed in the ongoing investigation of the September 11th terrorist, attacks. Evidence to this fact is given in a CNN interview with @stake's managing security architect Phil Huggins:

> "*One tool that Huggins believes may be used to track digital data is The Coroners Toolkit (TCT), a suite of freeware tools, parts of which @stake distributes. TCT was originally written by Dan Farmer, a researcher for Earthlink Networks, and Wietse Venema a researcher at IBM Corp.'s T.J. Watson Research Center "TCT is a standard tool, or rather a collection of tools that are designed to assist in a forensic examination of a computer. It's designed for Unix systems, but it can also get some data collection and analysis from non-Unix disks and media," Huggins said."* (CNN Online article)

When combined with good evidence gathering technique and methodology, TCT will be a powerful asset in the library of any forensic examiner.

Other Tools that may be used with TCT:

Autopsy Forensics Browser: This software package was designed to act as a graphical front-end to TCT. "It allows drive images to be analyzed at a file, block , and inode level. It also allows easy searches for strings in images." (Autopsy Forensics Browser README) This package can be downloaded at http://www.cerias.purdue.edu/homes/carrier/forensics

TCTUtils: This package is also located at the purdue website listed above. It adds extra functionality to TCT. It provides more inode information, mactime information on deleted files, it also provides the contents of a given block in various forms, and performs calculations to give the original block numbers from a block in an unrm image. (same)

dd: dd is a linux utility, which is used to copy files or entire file systems. It is very useful when combined with unrm. It is provided by default with Linux or it can be downloaded from http://www.redhat.com

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

PGP and variants: In the TCT README file the authors recommend that once the data collection is done, cryptographically sign all files collected. PGP can provide this functionality. There are numerous sites where PGP can be downloaded. These include http://www.gnupg.org/ (for Unix and Linux), http://www.pgp.com/ (now owned by Network associates, for Windows), and http://www.download.com/ (perform a search for PGP).

Bibliography:

Wagner, Mike. "The Coroners Toolkit: A Handy Suite of Utilities".
2000 http://rr.sans.org/threats/coroners_toolkit.php

Romig, Steve. "Forensic Computing".
1999 http://www.net.ohio-state.edu/security/talks/1999-10_forensics/forensics.pdf

Carrier, Brian. "Autopsy Forensics Browser – README".
2001 http://www.cerias.purdue.edu/homes/carrier/forensics/

McMillan, Jim. "Importance of a standard Methodology in Computer Forensics".
2000 http://rr.sans.org/incident/methodology.php

Rohde, Laura. "Forensic tool may play a role in investigation".
CNN.com/sci-tech http://www.cnn.com/2001/tech/industry/09/12/tech.forums.idg
12 September 2001

Farmer, Dan and Venema, Weitze. "The Coroners Toolkit – README"
/$TCT_HOME/docs/README

Farmer, Dan and Venema, Weitze. "Grave-robber source code"
/$TCT_HOME/bin/grave-robber

Farmer, Dan and Venema, Weitze. "Grave-robber README"
/$TCT_HOME/docs/grave-robber.README

Farmer, Dan and Venema, Weitze. "Lazarus README"
/$TCT_HOME/docs/lazarus.README

Farmer, Dan and Venema, Weitze. "Lazarus source code"
/$TCT_HOME/lazarus/lazarus

Farmer, Dan and Venema, Weitze. "mactime README"
/$TCT_HOME/docs/mac.README

Farmer, Dan and Venema, Weitze. "mactime source"
/$TCT_HOME/bin/mactime

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

Last Updated: September 1st, 2015

Upcoming SANS Training

Click Here for a full list of all Upcoming SANS Events by Location

SANS Crystal City 2015	Crystal City, VAUS	Sep 08, 2015 - Sep 13, 2015	Live Event
SANS Network Security 2015	Las Vegas, NVUS	Sep 12, 2015 - Sep 21, 2015	Live Event
SANS Seoul 2015	Seoul, KR	Sep 14, 2015 - Sep 19, 2015	Live Event
Data Breach Investigation Summit & Training	Dallas, TXUS	Sep 21, 2015 - Sep 26, 2015	Live Event
SANS Baltimore 2015	Baltimore, MDUS	Sep 21, 2015 - Sep 26, 2015	Live Event
SANS Perth 2015	Perth, AU	Sep 21, 2015 - Sep 26, 2015	Live Event
SANS Tallinn 2015	Tallinn, EE	Sep 21, 2015 - Sep 26, 2015	Live Event
SANS ICS Amsterdam 2015	Amsterdam, NL	Sep 22, 2015 - Sep 28, 2015	Live Event
SANS Bangalore 2015	Bangalore, IN	Sep 28, 2015 - Oct 17, 2015	Live Event
SANS DFIR Prague 2015	Prague, CZ	Oct 05, 2015 - Oct 17, 2015	Live Event
SANS Seattle 2015	Seattle, WAUS	Oct 05, 2015 - Oct 10, 2015	Live Event
SANS Tysons Corner 2015	Tysons Corner, VAUS	Oct 12, 2015 - Oct 17, 2015	Live Event
SOS: SANS October Singapore 2015	Singapore, SG	Oct 12, 2015 - Oct 24, 2015	Live Event
SANS Gulf Region 2015	Dubai, AE	Oct 17, 2015 - Oct 29, 2015	Live Event
SANS Cyber Defense San Diego 2015	San Diego, CAUS	Oct 19, 2015 - Oct 24, 2015	Live Event
SANS Tokyo Autumn 2015	Tokyo, JP	Oct 19, 2015 - Oct 31, 2015	Live Event
SANS South Florida 2015	Fort Lauderdale, FLUS	Nov 09, 2015 - Nov 14, 2015	Live Event
SANS Sydney 2015	Sydney, AU	Nov 09, 2015 - Nov 21, 2015	Live Event
SANS London 2015	London, GB	Nov 14, 2015 - Nov 23, 2015	Live Event
Pen Test Hackfest Summit & Training	Alexandria, VAUS	Nov 16, 2015 - Nov 23, 2015	Live Event
SANS Hyderabad 2015	Hyderabad, IN	Nov 24, 2015 - Dec 04, 2015	Live Event
SANS Cape Town 2015	Cape Town, ZA	Nov 30, 2015 - Dec 05, 2015	Live Event
SANS San Francisco 2015	San Francisco, CAUS	Nov 30, 2015 - Dec 05, 2015	Live Event
SANS Milan 2015	OnlineIT	Sep 07, 2015 - Sep 12, 2015	Live Event
SANS OnDemand	Books & MP3s OnlyUS	Anytime	Self Paced