



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>

GIAC Certified Forensic Analyst Practical Assignment

Version 1.0 (April 3, 2002)

SANS 2002 Conference

**David L. Wagner
September 18, 2002**

Table of Contents

Introduction	3
Part 1 – Forensic Analysis of a System	4
Synopsis of Case Facts	4
System Description	4
Hardware	5
Media Imaging	5
Media Analysis Preparation	8
Media and MAC Time Analysis	11
Deleted File Recovery	20
String Search	22
Conclusions	24
Part 2 – Analysis of an Unknown Binary	27
Binary Details	27
Program Description and Forensic Details	30
Program Identification	35
Legal Implications	38
Interview Questions	39
Additional Information	39
Part 3 – Legal Issues of Incident Handling (Wiretap Statute)	40
Appendices	42
Appendix A – Win2k/XP incident response script	42
Appendix B – Supporting media imaging files	45
Appendix C – sn.dat apptrace log	55
References	57

Introduction

The following paper is a practical assignment as required for GIAC Forensic Analyst Certification. Per the assignment instructions this paper has been divided into three distinct areas. These areas include forensic analysis of a system in an unknown state, analysis of an unknown binary application downloaded from the GIAC web site, and research into the Wiretap Statute as it relates to me as a system administrator.

It should be noted that creative license has been taken when related to the case described in Part 1 of this paper: Forensic Analysis of a System. Specific details have been modified as necessary to “protect the innocent” and to hide information that could potentially assist in identifying the system being investigated and the organization involved. Signatures such as usernames, dates and times, IP addresses and DNS names have been sanitized to maintain anonymity.

While many of the case facts have been altered, the system analyzed in this document was an actual machine on my organization’s network. Absolutely no modifications were made to the system prior to the investigation with the deliberate intent of aiding in the forensic investigation.

Part 1 – Forensic Analysis of a System

This section demonstrates the use of various media imaging and forensic analysis tools during the investigation of a suspect computer system.

Synopsis of Case Facts

On August 26th, 2002, my unit was contacted to perform an investigation related to the attempted access of an internal web server by an unauthorized user. The web site residing on this server has been organized into both public and restricted areas depending upon content sensitivity. Restricted areas of the web site require unique user authentication prior to allowing browser access while public areas accept anonymous connections.

During a routine check of the web server logs, one of the server's administrators noticed that over the previous weekend a particular machine on our network had repeatedly attempted to access restricted areas of the server. While information contained in the logs showed the attempts as being unsuccessful, management was concerned that someone may have attempted to breach the system's security. Unfortunately, the inclusion of sensitive information prevented the server administrators from turning the logs over at this time for further analysis by my section.

While I was unable to investigate the log files directly, the web server administrators were willing to provide the IP address of the machine that was listed as the source of the attempts. Using this information I was able to quickly locate the suspect machine on our internal network.

The machine in question is a standard PC used by network administration personnel to perform routine tasks. Work logs show that the owners of the suspect machine were not scheduled to work during the timeframe that the access attempts occurred. In addition, building access logs do not show the owners of the suspect system as being on site. All employees known to use the system deny making the unauthorized access attempts.

Management would like someone from my unit to identify if evidence exists on the suspect machine that could help to identify if the system was indeed used to make the access attempts and, if so, who may have been operating the machine at the time of the incident.

System Description

By interviewing the owners of the suspect system I was able to obtain the following information:

The system being analyzed is used to run various network management applications and is located within an office area inhabited by several of my organization's network management personnel. Personnel other than network administration frequently access the area. While a lock exists on the door of the office area where the machine resides, most often the door is left unlocked, even after hours.

The suspect system consists of a PC loaded with a standard install of Microsoft Windows XP Professional. Administrators of the system state that various network management applications have been loaded on the system. The system has also been used in the testing of potential management applications. Most unused applications have been removed after they have been evaluated. Some applications may have been loaded on the system for testing and have since been forgotten.

The suspect system is connected directly to a switch that provides network access for several floors of the building where the system resides. No special VLAN configuration has been made on ports of the switch to isolate the system from regular users. The system is reported to be left running at all times but logged out when not in use.

The victim web server exists at a different location on a dedicated subnet behind a firewall. The firewall is configured to allow access by the organization's internal IP addresses but to block all services except TCP ports 80 and 443. The two locations are connected via an internal network maintained by my organization.

Hardware

Below is a detailed list of the hardware that was seized for the investigation. A tag has been affixed to the confiscated equipment bearing identifiers matching the tag number listed below.

Tag # 3601 – Dell OptiPlex GX1 Tower Computer System (Serial # xxxxx)

- 350MHz Pentium II Processor w/ 256MB RAM
- (1) Internal Fujitsu 6GB IDE hard drive
- (1) Internal 3.5" floppy drive
- (1) Internal CDROM drive
- Integrated video, sound, and 10/100 Ethernet adapters

Media Imaging

Confiscation of the suspect machine was performed without prior notification. When I arrived on site the system was running but logged out. I immediately connected an analysis machine to the network to enable the transferring of images. The analysis machine consists of an Intel PC with a large hard drive running RedHat Linux 7.2 and is described in more detail in the next section.

NetCat v1.10 from @stake (<http://www.atstake.com/research/tools/nc110.tgz>) was loaded on the machine to assist in copying drive images and other evidence across the network.

After the analysis machine was up and running and configured with a usable IP address, I created and then changed into a directory that was to be used for evidence collection. I began the imaging process with the intent of preserving as much evidence as possible. I did this by logging into the suspect machine as Administrator and then inserting a response toolkit CD I've put together to assist in handling Windows 2000/XP incidents.

My response toolkit CD includes many of the tools provided on the course CD that was distributed during the SANS 2002 System Forensics, Investigation, and Response class. In addition, I've included updates to several of the tools, added several new tools, and created a script to assist in the capturing of pertinent data from a live Windows 2000/XP machine. The script is included in Appendix A of this paper.

Once the toolkit had been loaded in the suspect machine's CDROM drive, I executed the following command from the Start | Run menu to open a known, safe command prompt from the response toolkit:

```
e:\win2k_xp\cmd.exe
```

I then proceeded to dump the entire contents of the suspect machine (MCON)'s RAM to a file on the analysis machine (FA1). This was accomplished by executing the following commands on the designated boxes:

```
(FA1 ) nc -l -p 5000 > ram.dd
(MCON) .\dd if=\\.\PhysicalMemory conv=noerror | .\nc fa1 5000
```

Once the process had completed, I executed my response script using the commands below to assist in the collection of pertinent information on the live system and store the output on the analysis machine:

```
(FA1 ) nc -l -p 5000 > live.txt
(MCON) \ir-win2k_xp.cmd | .\nc fa1 5000
```

After the script completed execution, I removed the response toolkit CD and pulled the system power cable to prevent any additional changes from being made to MCON. I disconnected the analysis machine from the live network and connected it to a small, standalone hub. I then disconnected the suspect system from the live network and connected it to the hub. Finally, I reconnected MCON's power cable, inserted a copy of the @stake Pocket Security Toolkit CD I picked up at SANS 2002, and booted off of the CD.

The @stake Pocket Security Toolkit CD is a stripped down, bootable Linux distribution that includes many useful tools for performing forensic analysis. An ISO image of the CD can be downloaded from the @stake web site at the following location:

http://www.atstake.com/research/tools/pst/pocket_security_toolkit_3.iso

Once MCON had booted and was running on the CD image, I immediately generated MD5 hashes of the suspect system's disk partitions. This was done to provide documentation that evidence was properly preserved throughout the investigation. I created these hashes and stored them on the analysis system by executing the following commands:

```
(FA1 ) nc -l -p 5000 > md5.org
(MCON) md5sum /dev/hda1 /dev/hda2 | nc fa1 5000
```

I then configured MCON's network interface with a usable IP address and proceeded to dump its hard drive partitions to image files on the analysis machine. This was accomplished using the commands shown below:

```
(FA1 ) nc -l -p 5000 > c.dd
(MCON) dd if=/dev/hda1 | nc fa1 5000

(FA1 ) nc -l -p 5000 > d.dd
(MCON) dd if=/dev/hda2 | nc fa1 5000
```

After the above process had completed, I removed the @stake CD and pulled the power to the system. I then generated MD5 hashes for all of the files I had stored on the analysis system using the command below:

```
(FA1 ) md5sum * > md5.txt
```

Finally, as MCON was not critical to normal business operations and its functionality could easily be reproduced, I confiscated the system and transported it back to my office where a case identification tag was attached. I then copied the *md5.org*, *live.txt*, and *ram.dd* files I had created above to a CD-R, labeled the disk with the case identifier, and stored the disk in a secured area along with the confiscated computer system.

The MD5 hash files and the output from the incident response script have been included at the end of this document in Appendix B. I have also included the MD5 hash results below for comparison purposes:

Original –

```
145e0b8df7ce54c3ef0a82e724059b4f /dev/hda1
165387cf9c056761d977eea268b91eeb /dev/hda2
```


Image –

```
145e0b8df7ce54c3ef0a82e724059b4f  c.dd  
165387cf9c056761d977eea268b91eeb  d.dd
```

Note that both sets of hashes are identical which proves that the imaging process created an exact duplicate.

Media Analysis Preparation

Prior to performing any of the steps outlined above I had created an analysis machine equipped with the necessary hardware and software that would provide me the ability to perform common tasks related to media imaging and analysis. The forensic system consisted of a Dell OptiPlex GX150 with a 1GHz Pentium III processor and 256MB of RAM. The system's original 19GB hard drive was replaced with a 60GB Ultra ATA/100 hard drive from Western Digital.

Before initiating the operating system install process, the machine's hard drive was wiped using the *dd* command as shown below to remove any residual information:

```
(FA1 ) dd bs=1000k if=/dev/zero of=/dev/hda
```

Once the process above had completed, the system was connected to a dedicated analysis network and RedHat Linux 7.2 was loaded. The system was then updated with the latest patches.

To enable the copying of disk images and other data over the network, I loaded NetCat v1.10. I also installed The @stake Sleuth Kit v1.50 (<http://www.atstake.com/research/tools/task>) and the Autopsy Forensic Browser v1.60 (<http://www.atstake.com/research/tools/autopsy>) for media analysis.

The @stake Sleuth Kit, also known as TASK, is a collection of tools that can be used for forensic investigations involving both UNIX and Windows file systems. Support for Microsoft's NTFS was recently added in version 1.50. The tools included in TASK can be executed from the command line but are much easier to use via @stake's graphical interface known as the Autopsy Forensic Browser.

@stake's Autopsy Forensic Browser consists of a set of Perl scripts that provide an HTML interface to the TASK toolkit. The web interface is easy to use and provides access to the most common tools that are used during a forensic investigation. The system allows investigators to connect remotely to the forensic analysis system using a web browser.

It should be noted that prior to installing the above applications, I was required to rebuild Perl with large file support as per Autopsy's installation instructions. This

step was necessary to provide analysis support for image files greater than 2GB in size.

Now that I had all of my prep work done, I decided to begin the forensic investigation by translating the memory dump I had made at the start of the imaging process into a more accessible format. The file *ram.dd* had been created using the UNIX *dd* tool. I used the UNIX *strings* command to locate string information in the dump file and store it into a file called *ram.str* as shown below:

```
(FA1 ) strings ram.dd > ram.str
```

The resulting file was over 70MB in size which made it impractical to step through manually. Even though I didn't intend to make use of the file right at this moment, it could prove useful should I need to scan it for specific information later in the investigation.

I then examined the output of the *ir-win2k_xp.cmd* script I had executed on the live system. The script is listed in its entirety in Appendix A of this paper. Summarized output from running this script on MCON is included in Appendix B and is called *live.txt*. I was unable to provide the entire output due to the size of the resulting file.

The *ir-win2k_xp.cmd* script provided a snapshot of the suspect system before it had been powered off. Information that had been preserved by the script included: environment variables, active processes, network configuration, and system logs. I analyzed this data looking for suspicious items such as the following:

- 1) Unexplained environment variables
- 2) Unidentified processes and services
- 3) Mysterious ARP entries, network connections, and network shares
- 4) Unauthorized software installations
- 5) Log file abnormalities

After spending several hours sifting through the output of the *ir-win2k_xp.cmd* script I had been unable to find anything that seemed out of place. While the volume of information was daunting, nothing stood out as overly suspicious. Even the system logs around the time of the event seemed legitimate and appeared to be intact.

I decided it was time to start analyzing the drive images I had obtained. If I found anything abnormal while digging through their contents I could always go back and reexamine both the *live.txt* and the *ram.str* files. This information could prove valuable if I were to find anything later in the investigation.

The Autopsy Forensic Browser requires that images related to an investigation be placed in a “morgue” directory for analysis. In addition to the image files, the directory must contain a file called *fsmorgue* that acts as a table of contents for the investigation. To prevent modification to my original image files I made copies of the files and placed them in the morgue directory. This is demonstrated below:

```
(FA1 ) cp /opt/gcfa/c.dd /opt/gcfa/d.dd /opt/morgue
```

So that I could verify later in the investigation that the analysis process hadn’t modified evidence, I changed to the */opt/morgue* directory and executed the following command which generated MD5 hashes for each file:

```
(FA1 ) md5sum c.dd d.dd > md5.txt
```

I then added the following entries to the *fsmorgue* file to reference my image files:

c.dd	ntfs	C:	EST5EDT
d.dd	ntfs	D:	EST5EDT

Now that I had everything configured as necessary I started Autopsy using the following command where <InvestigatorIP> is the IP address of the investigator that will be connecting to the Autopsy server to perform analysis:

```
(FA1 ) autopsy 8888 <InvestigatorIP>
```

Executing this command on FA1 started the Autopsy server which in turn displayed the following instructions on how to connect using the investigative machine:

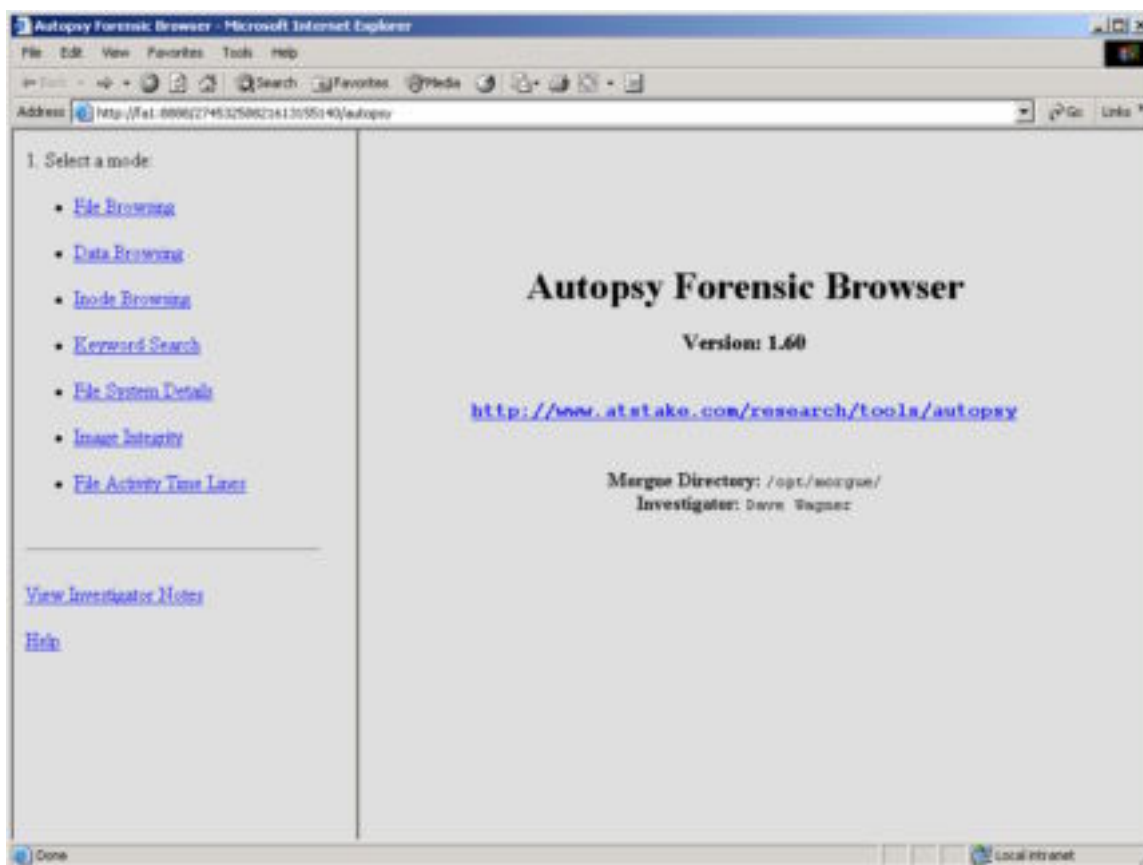
```
=====
Autopsy Forensic Browser
ver 1.60
=====

Morgue: /opt/morgue
Start Time: Wed Sep  4 19:28:42 2002
Investigator: Dave Wagner

Paste this as your browser URL on <InvestigatorIP>:
    fal:8888/27453258821613155140/autopsy

Keep this process running and use <ctrl-c> to exit
```

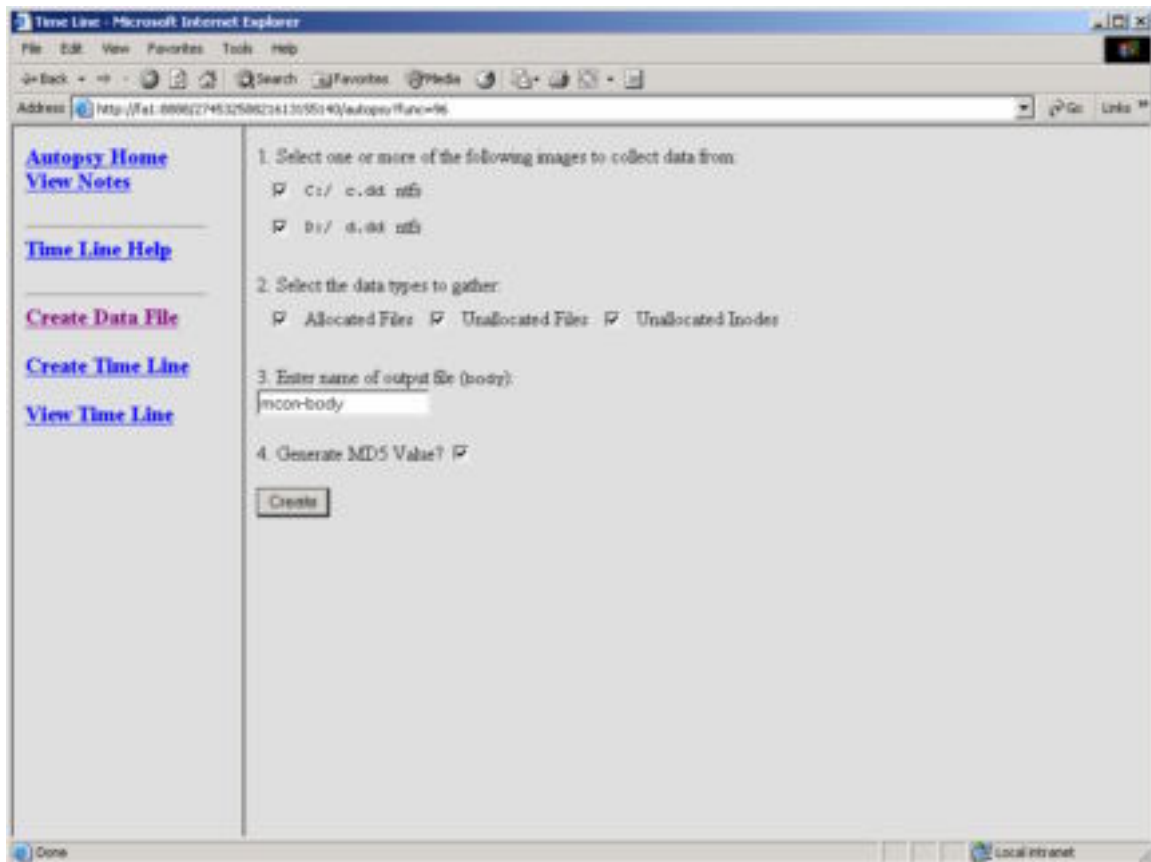
Per the instructions above I pasted the URL generated by the Autopsy server into a web browser on the investigator’s machine. This displayed the following web page:



Media and MAC Time Analysis

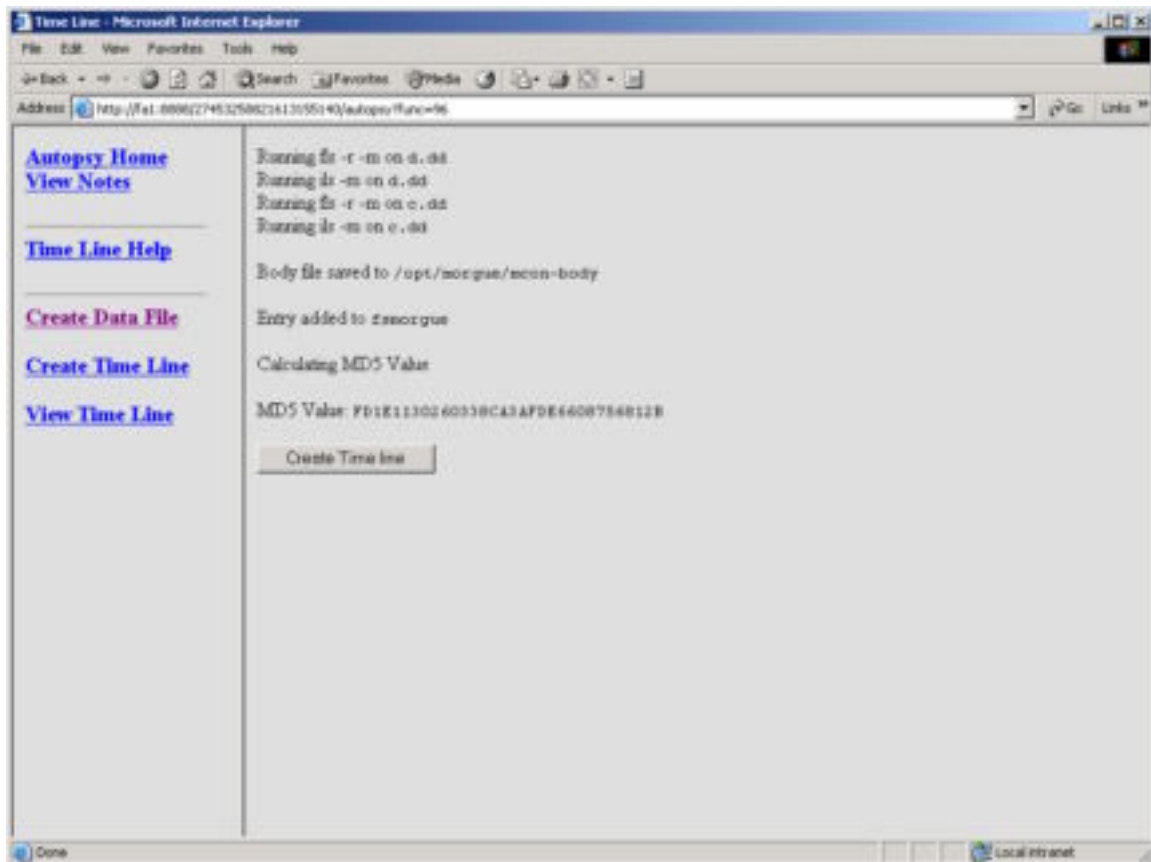
Now that I was connected to the analysis system from my investigative machine I was ready to begin analyzing the image files that had been created. I started by generating a time line of events for the hard drives.

Autopsy makes this job simple. The first step is to create a body file. The body file contains MAC times and other pertinent information for all files on the file systems being analyzed. To create this file I selected "File Activity Time Lines" from the Autopsy main menu and then "Create Data File" from the resulting web page. The "Create Data File" link opened a form where I could enter specifics about the body file I wanted to generate. The completed form is shown below:



I then pressed the “Create” button which generated the body file for my project. The creation process took less than a minute and generated the following output:

© SANS Institute 2000 - 2002



I then selected “Create Time Line” from the page above which in turn displayed a form where the investigator could specify what information the timeline should contain. The completed form is displayed below:

© SANS Institute

Time Line - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Search Favorites Media Print

Address http://fa1.8886/27453258621413195140/autopsy/func=96 Go Links

[Autopsy Home](#)
[View Notes](#)

[Time Line Help](#)

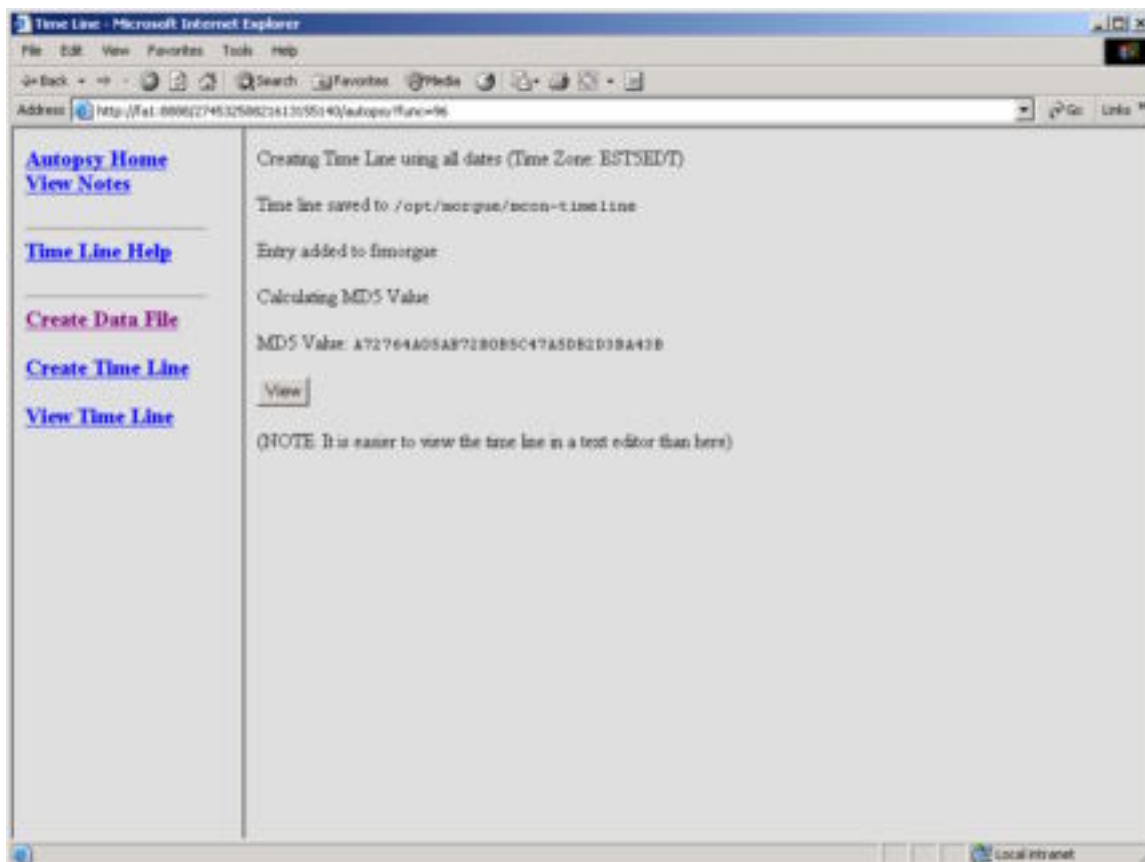
[Create Data File](#)
[Create Time Line](#)
[View Time Line](#)

1. Select the data input file (body):
☒ moon-body
2. Enter the starting date:
None ☒
Specify:
3. Enter the ending date:
None ☒
Specify:
4. Select the time zone:
5. Enter the file name to save as:
6. Enter the location of the /etc/passwd file:
 Inode:
7. Enter the location of the /etc/group file:
 Inode:
8. Generate MD5 Value? ☒

Done Local intranet

I completed creation of the timeline by pressing the “Create” button on the form shown above. This process took less than a minute to complete and resulted in the following web page being displayed:

© SANS Institute

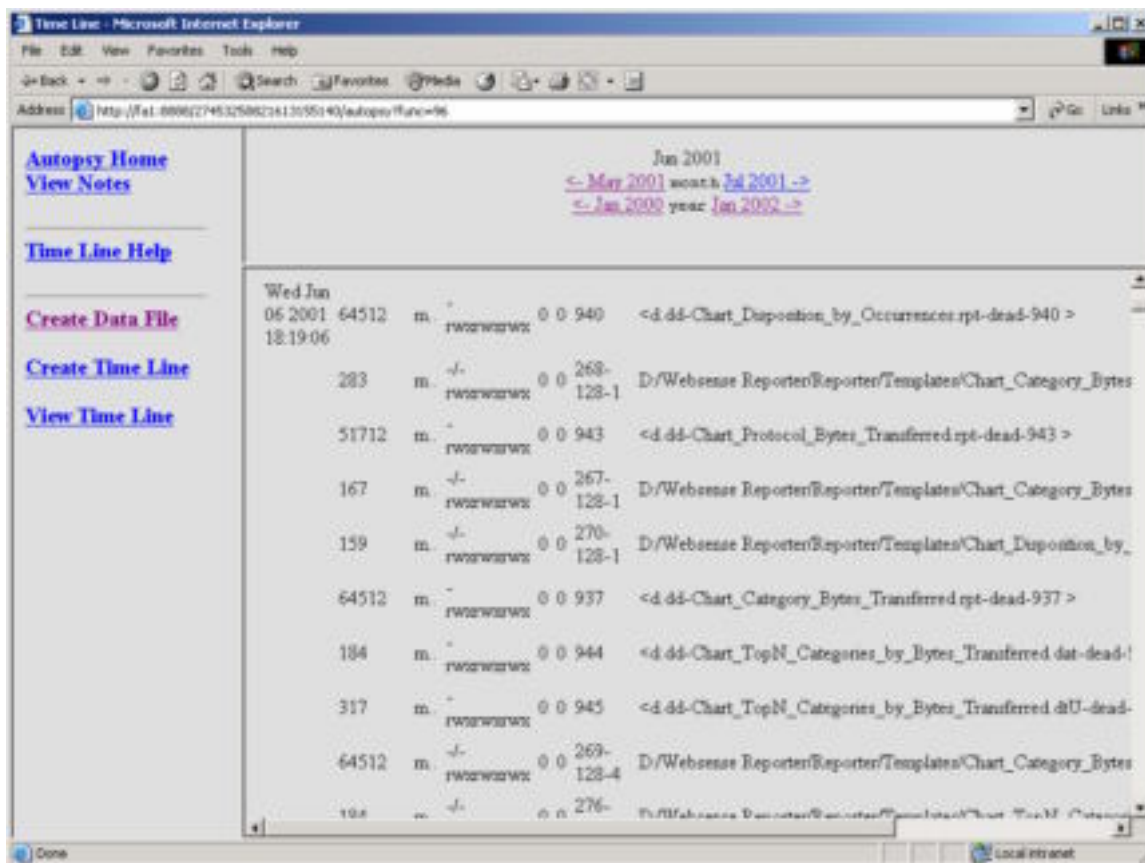


By pressing the “View” button I was able to see the timeline using the web browser. As stated in the screen above it is much easier to get a complete picture of the timeline using a text editor than using the web browser. The web browser feature is nice for walking through the timeline step-by-step.

Using the timeline I was able to display the following web page which shows when the operating system was installed on the machine (February 19, 2002):

Date	Time	Size	File Name	Path
Tue Feb 19 2002	04:34:07	4096	mac -/t-rr-rr-r	C:\MFTMarr
		131072	mac -/t-rr-rr-r	C:\UpCase
		127768	mac -/t-rr-rr-r	C:\Bdmmap
		8192	mac -/t-rr-rr-r	C:\B-not
		0	mac -/t-rr-rr-r	C:\33133
		310280	mac -/t-rr-rr-r	C:\Secure\SDS
		224	mac -/t-rr-rr-r	C:\Secure\SDH
		344	mac -/t-rr-rr-r	C:\Extend
		34471936	mac -/t-rr-rr-r	C:\LogFile
		200	mac -/t-rr-rr-r	C:\Secure\SD
		0	mac -/t-rr-rr-r	C:\3-128

I was also able to determine that data on the D: drive pre-existed the OS installation on the C: drive. As I searched through the timeline I found where files had been created, modified, and deleted on the D: drive as early as June of 2001. Below is a screen dump showing this:



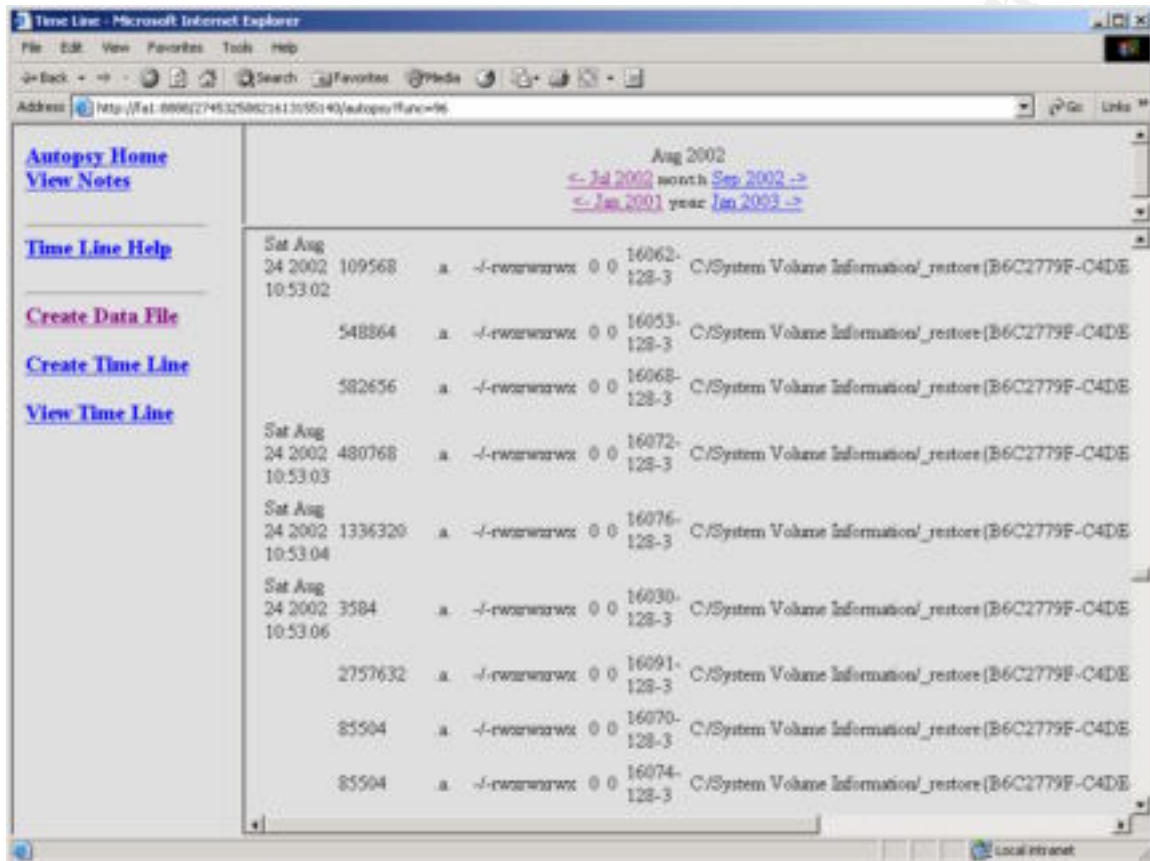
The text version of the timeline was useful as well in tracking when the system was loaded and when patches and applications were installed. As I manually paged through the data I realized the system had been through many modifications in its short lifetime.

While scanning through the timeline was very interesting, I decided I needed to concentrate my analysis on the dates that the incident took place, August 24th – 25th, 2002. Due to the bulk of information it was very easy to become both sidetracked and overwhelmed.

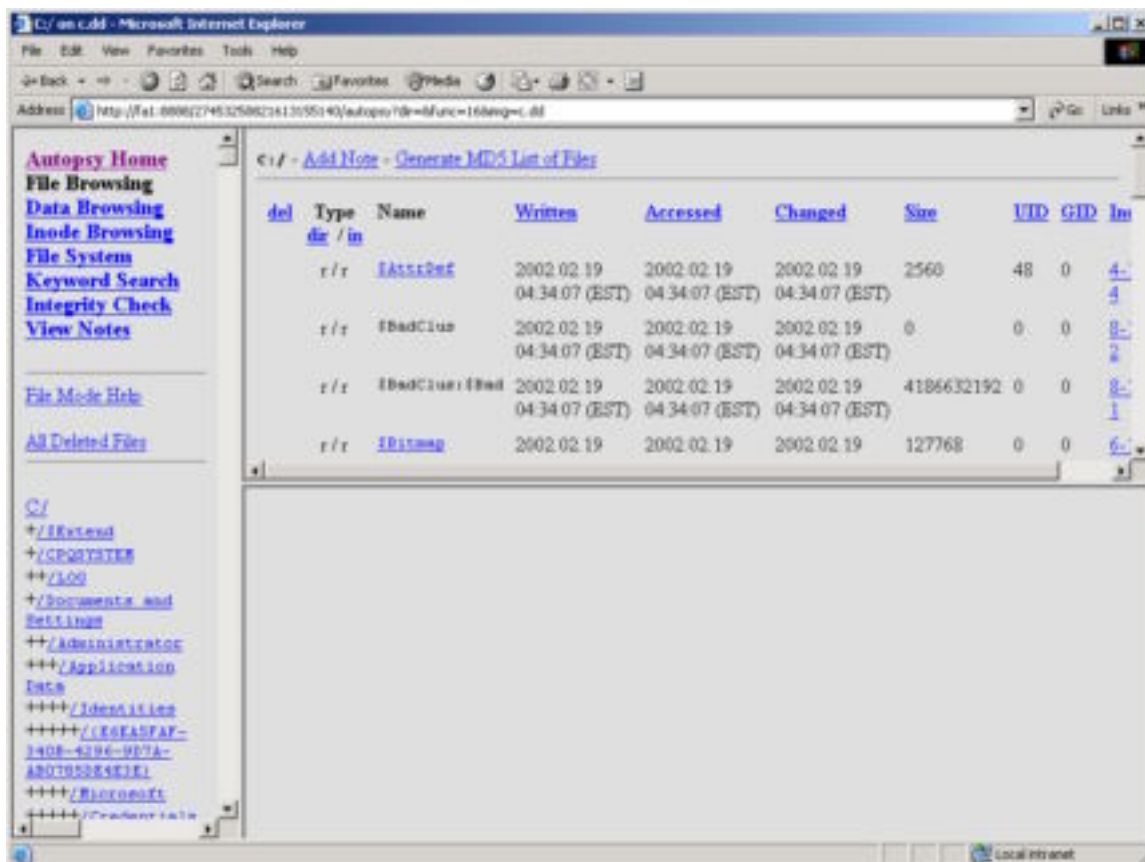
Using the *mcon-timeline* file that had been generated above, I was able to put dates to important events that took place in the system's lifetime. The list below shows examples of some of these events:

- 1) Operating system loaded: Tuesday, February 19, 2002
- 2) Patches and applications loaded: February 19 – 22, 2002
- 3) Additional applications loaded: Wednesday, March 27, 2002
- 4) Last user login: Friday, August 23, 2002
- 5) Start of the forensic investigation: Tuesday, August 27, 2002

Using Autopsy's web interface I looked for any suspicious activity that took place during the timeframe of the incident. Using the timeline I was able to see that files had been accessed and modified on the dates in question but everything appeared legitimate. All of the entries seemed to be caused by normal system processes similar to what is shown in the screen dump below:



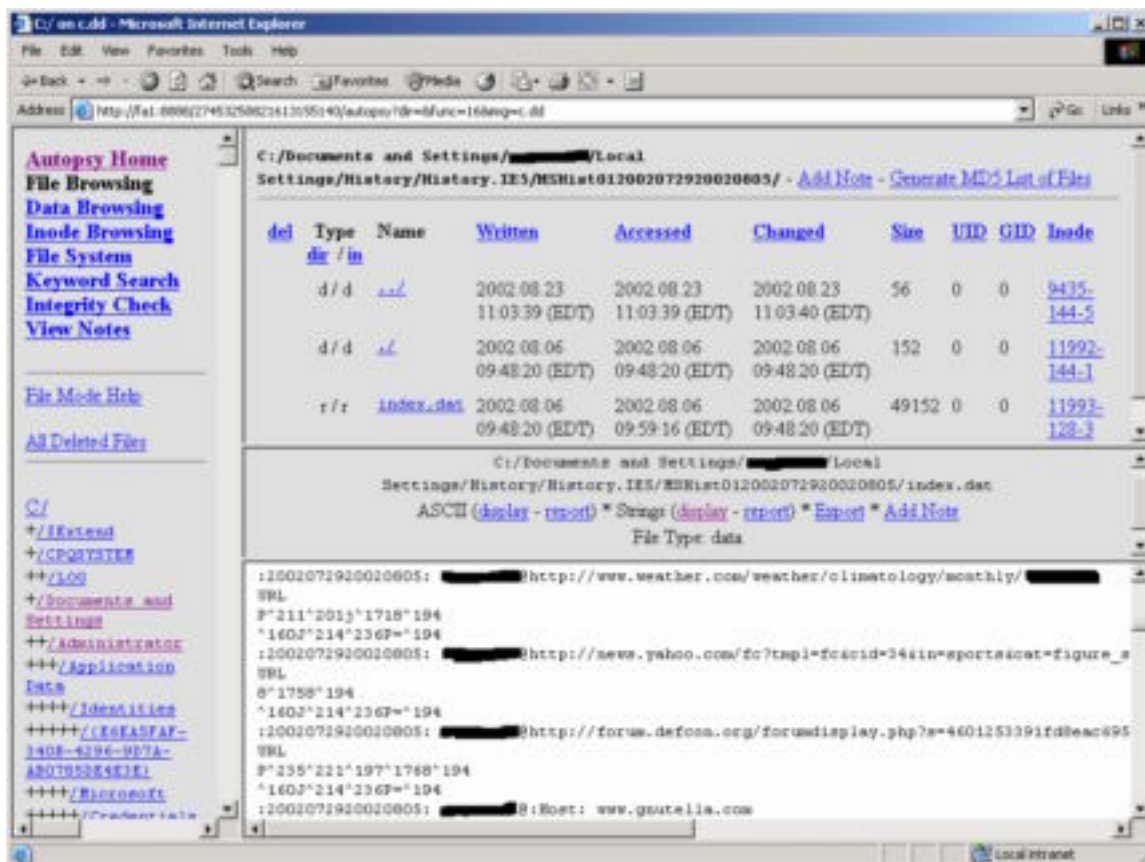
Even though nothing irregular had shown up in MCON's timeline I decided to analyze the system's internet history use information to see if there was anything there that showed the machine had ever been used to access the restricted site. I did this by selecting "Autopsy Home" on the current web page and then "File Browsing" from the main menu. These actions displayed a page where I could select an image to browse. I selected the C:/ image and the following page was displayed:



From this web page I was able to easily browse the file system of the selected drive. Starting with the page displayed above, I checked to see if the user “Administrator” had any references to the victim web server in the following folders:

C:/Documents and Settings/Administrator/Local Settings/History
C:/Documents and Settings/Administrator/Local Settings/Temporary
Internet Files

Scanning through the directories for the Administrator user account revealed nothing of significance so I decided to check the same internet history directories for the rest of the system’s users. While I didn’t come across anything directly related to the investigation, the screenshot below shows how easily Autopsy allows for the viewing of file content contained within an imaged file system. This can be accomplished either in raw format or by parsing for string information as was done in the following example by selecting a file and then pressing “display”:



Had I found anything of significance I could have easily generated a report showing detailed file information by selecting the “report” link. Autopsy also allows the investigator to add descriptive notes containing links to pertinent data to the forensic investigation. Once a note has been created the investigator can later review it and all other case notes by selecting “View Notes”.

I performed the same process as above for the Temporary Internet Files directory for each user and came up empty handed. So far I had found no evidence that the suspect machine had ever accessed the victim web server.

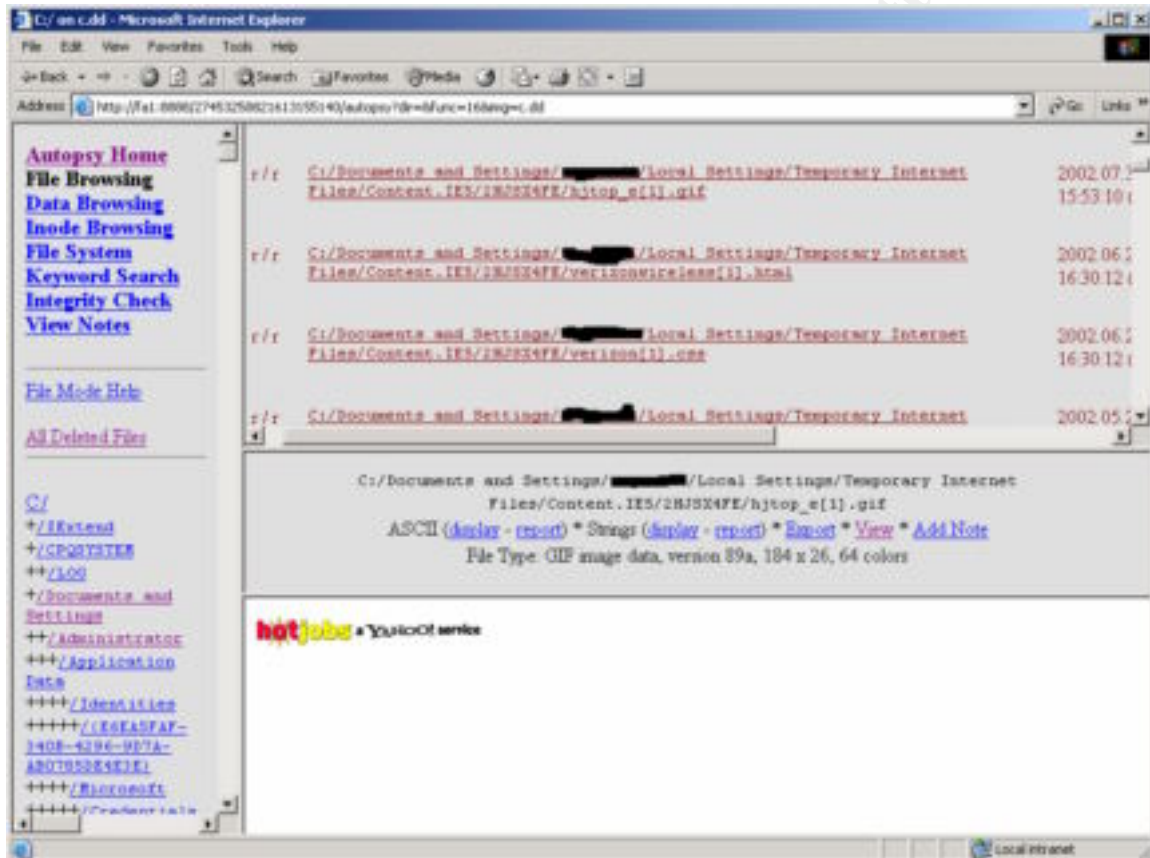
Before I moved on in the investigation I decided to at least export the internet history files to my machine so that I could archive them for later review. Once again, Autopsy made this process extremely easy to perform. All I had to do was select the file I wanted to export using the web browser and then click the web link called “Export”. This opened a window where I could select the destination to use for saving the file to my investigative machine. Once I did this, the raw file was downloaded and saved on my hard drive.

Deleted File Recovery

While I hadn’t found anything suspicious using Autopsy’s file browsing functionality, I decided to examine the system for deleted files that might relate to

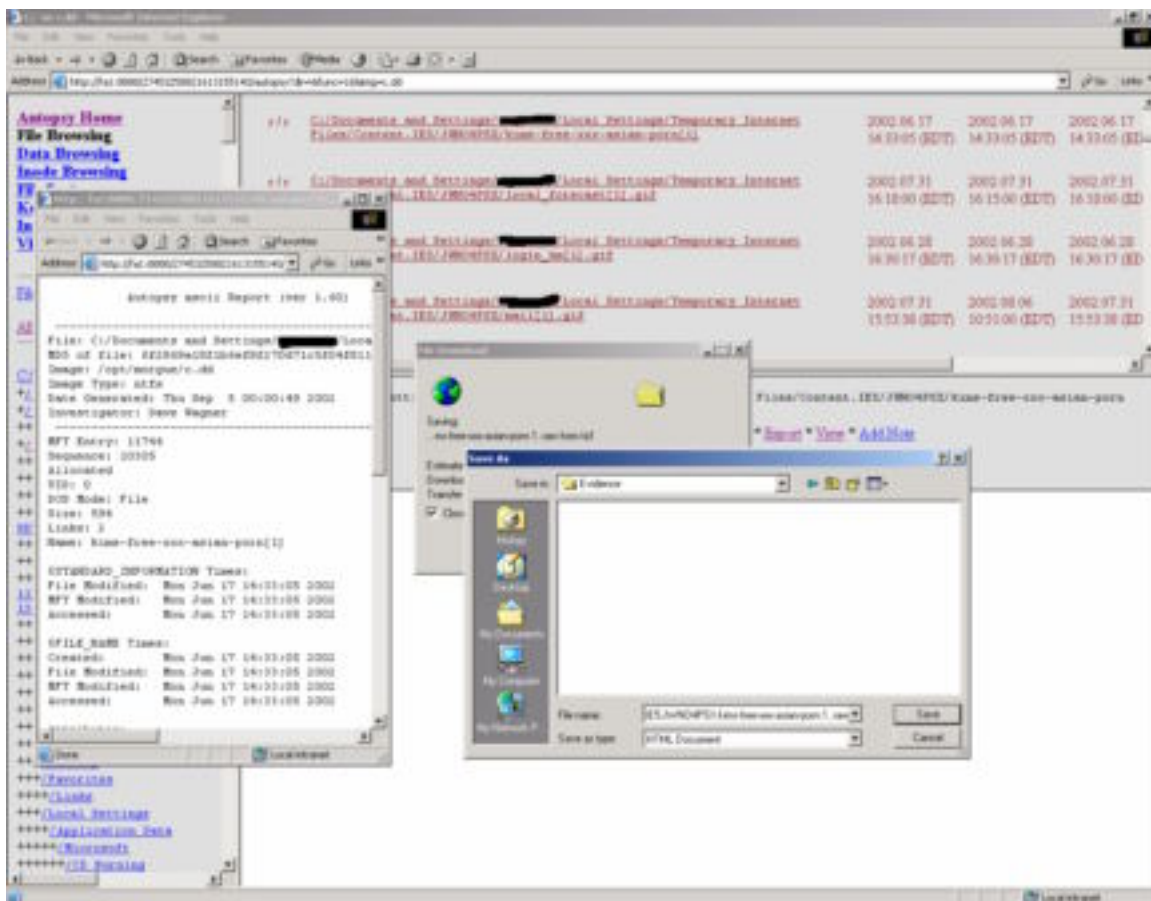
the incident. Autopsy makes this easy to accomplish. All the investigator needs to do is select the “All Deleted Files” link while under “File Browsing” and Autopsy will generate a list of files that have been deleted from the system.

Using a web browser the investigator can scan through a list containing all of the deleted files on a file system. If an interesting file is found, the investigator can view the contents of the file by selecting it. If the file contains an image, the investigator can click the “View” link to display it. This functionality is shown below:



The deleted files report did not show anything of significance that related to the case. While I was unable to find evidence that MCON had been used to access the victim web server, I did notice that one of the system’s users had been using the system to browse sites that appeared to be pornographic in nature. Although this information was unrelated to the case at hand it did violate my organization’s acceptable use policy.

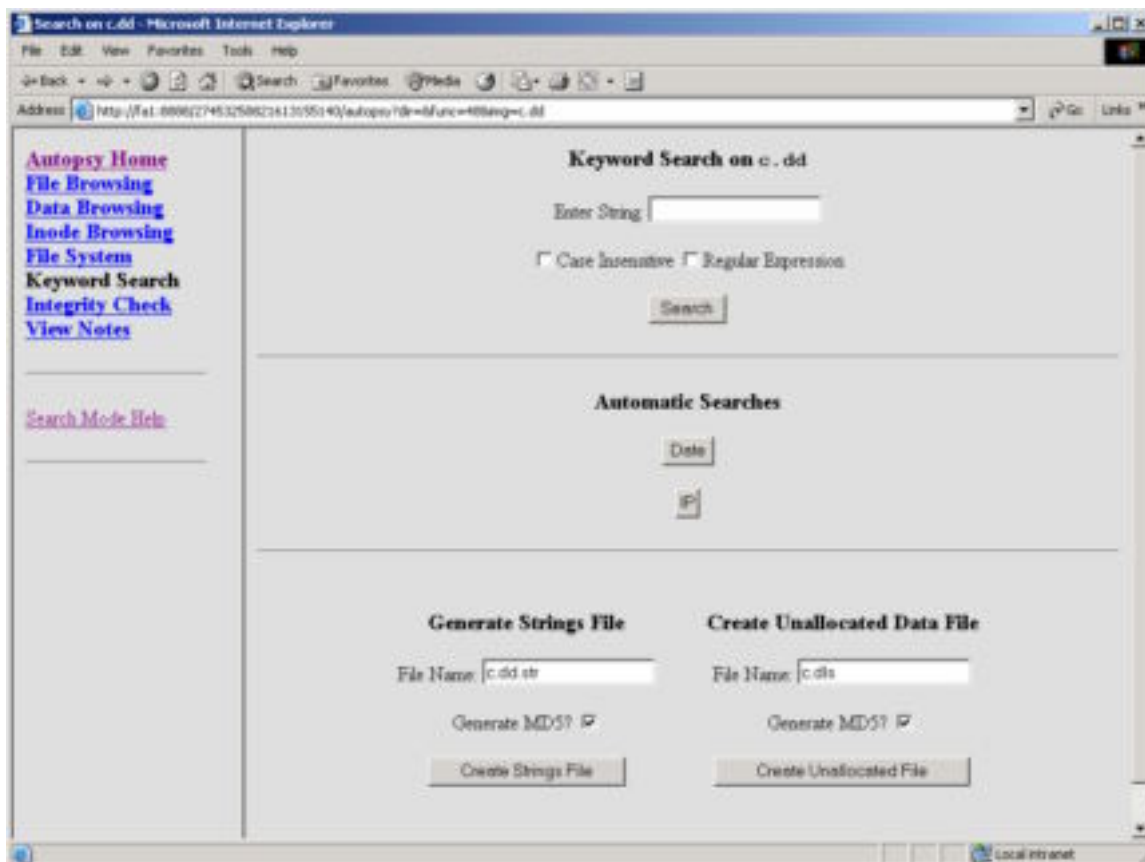
I decided to preserve the data by using Autopsy to generate detailed reports and then downloading the files to the investigative machine. This was easily accomplished by selecting each explicit file and then clicking the “Report” and “Export” links. Use of these functions is demonstrated in the screen dump below:



Using the Autopsy undelete feature I was able to recover the explicit web files so that they could be turned over to management for further review.

String Search

I decided to make one last attempt at locating additional evidence that could be used to link MCON to the incident. The final tool I put to use was the Autopsy keyword search functionality. This feature can be accessed from the Autopsy main menu by clicking “Keyword Search” and then selecting the drive image to search. Selecting these links presented the following form:



Before attempting a search I used the “Create Strings File” and “Create Unallocated Data File” buttons at the bottom of the page to collect string information from the selected image file.

Creating string information for unallocated data using the above method worked. Unfortunately, the “Create Strings File” button failed with an error message about the *c.dd* file being too large. I realized that this was an issue with large file support so I scanned the internet for a quick fix. Customizing a solution I found at the following location I was able to create a workaround to the *strings* problem:

<http://lists.jammed.com/forensics/2002/04/0033.html>

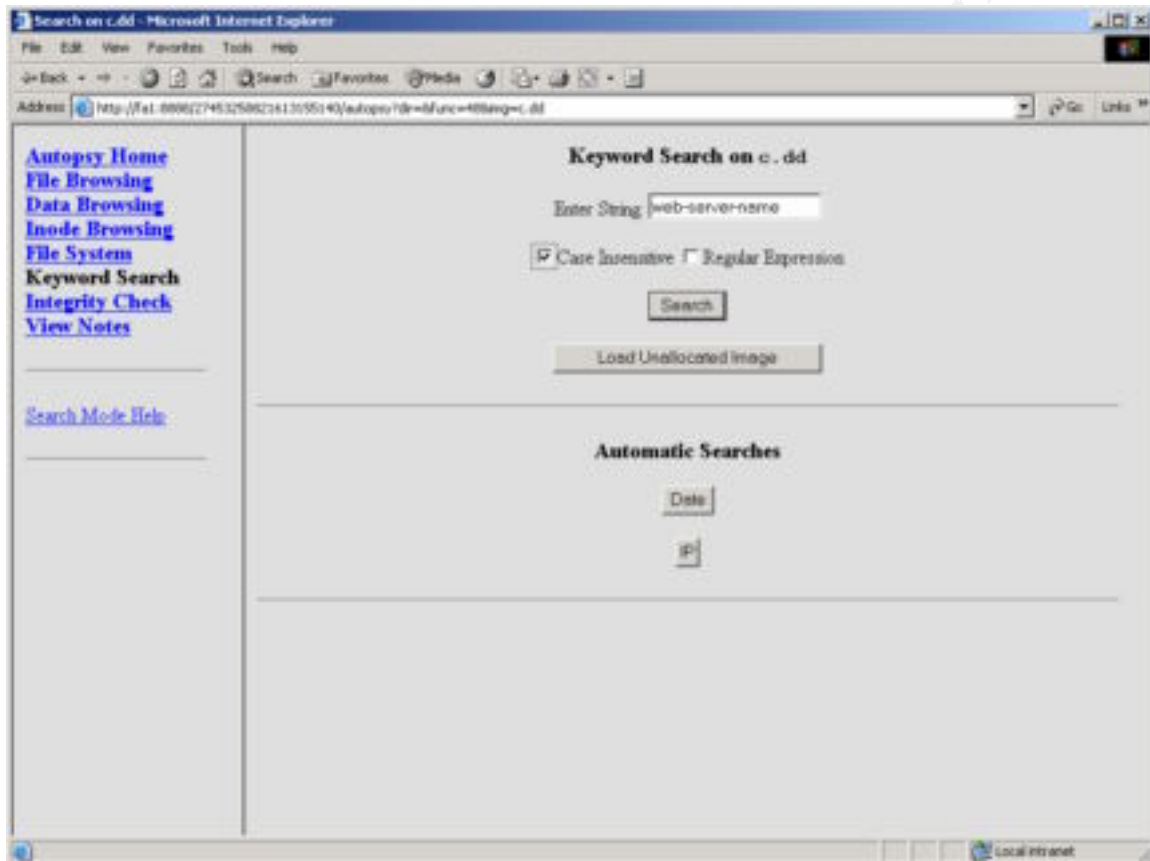
By manually redirecting the *c.dd* file into the *strings* command I was able to generate the information I required. Below is the command I issued on the forensic analysis machine with the parameters required for use by Autopsy:

```
(FA1 ) strings -t d < c.dd > c.str
```

This created the *c.str* file which I needed to perform the string search. I then used the *md5sum* command to generate a hash which I in turn concatenated onto the end of the *md5.txt* file. Finally, I manually edited the *fsmorgue* file by adding the line shown below so that Autopsy would see my string data:


```
c.str strings c.dd
```

The above modifications allowed me to search the *c.dd* drive image for string information. By entering the web server's name into the search box I was able to quickly scan the allocated portion of the file system for any references. The image below shows this functionality:

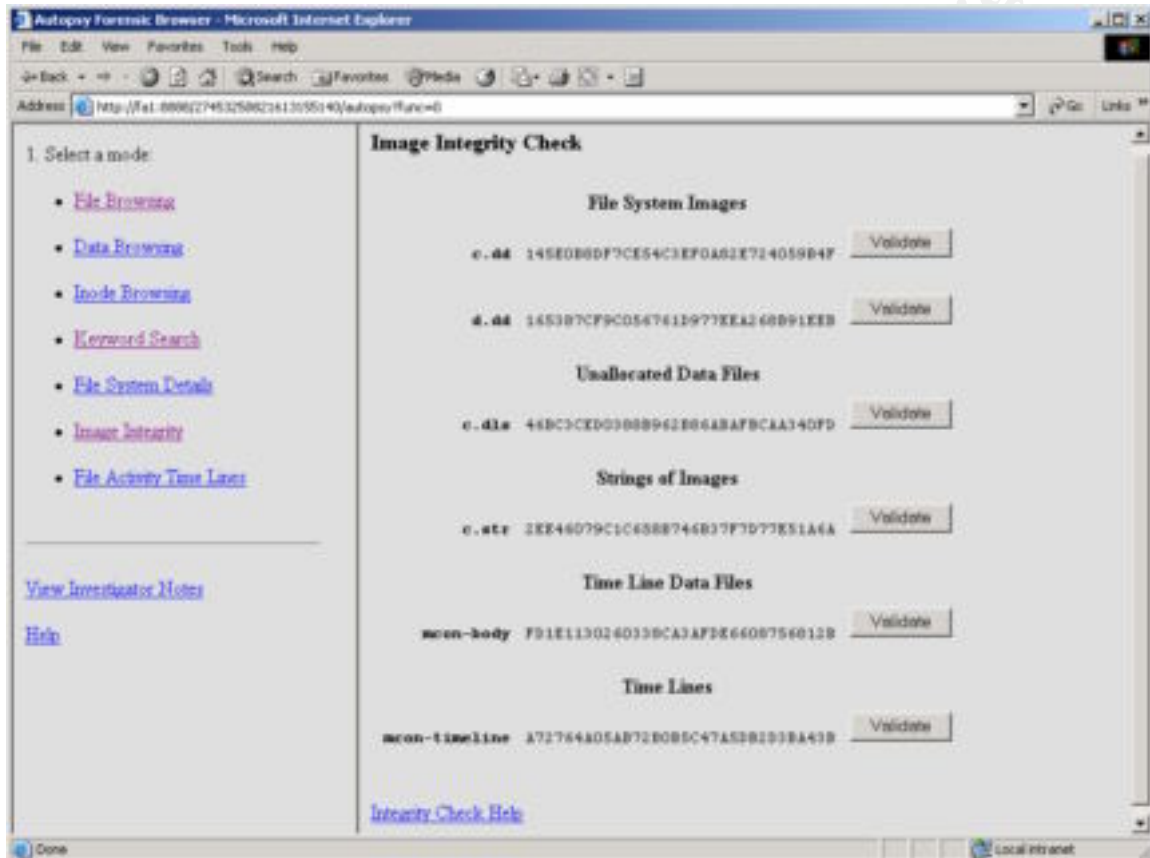


After the search completed and I analyzed its results, I selected the “Load Unallocated Image” button shown on the form above so that I could look for unallocated blocks containing references to the web server. I then repeated the search and analysis process for the unallocated portion of the file system.

Using the string search capability of Autopsy I was finally able to uncover references to the victim web server. Although, after further investigation, I discovered that while the references I found did point to the server in question, they did not point to the restricted folders that this case was concerned with. In other words, the case had not been solved.

Conclusions

To finalize my initial investigation of the suspect machine, I used Autopsy's MD5 integrity check capability to prove that the analysis process hadn't modified the MCON image files. I accomplished this by selecting "Image Integrity" from Autopsy's main menu. This displayed a screen where I could validate my image files merely by pressing a "Validate" button. I performed this action for each of the images contained in my project. The validation screen is shown below:



Once I had successfully validated each image using Autopsy, I performed an additional check by manually comparing the hashes that Autopsy displayed against the MD5 hashes I had generated during the imaging process. Analyzing this information showed that the image files had remained intact throughout the entire investigation.

The Autopsy tools had proven useful during the investigation in providing a timeline of events for the suspect machine. Using the tools I was able to see everything from when the machine had been loaded up until it had been taken off line for imaging.

As I was unable to discover any evidence directly linking MCON to the incident, my next step would be to broaden the scope of the investigation and pull in more resources. This was necessary as the possibility existed that the IP address information recorded in the victim web server logs could have been generated by

a machine using a spoofed address or by someone tampering with the logs. In addition, a more detailed analysis would need to be performed on the MCON system to prove that someone hadn't used the machine and then covered their tracks to avoid being caught.

© SANS Institute 2000 - 2002, Author retains full rights.

Part 2 – Analysis of an Unknown Binary

This section demonstrates the use of various, commonly available tools to analyze an unknown file that was discovered on a compromised system. The file analyzed in this section of the paper was retrieved from the GIAC website at the following location: <http://www.giac.org/gcfa/sn.zip>. Once downloaded, the ZIP file was copied to a RedHat Linux 7.3 virtual machine running on a PC with VMWare 3.1 (<http://www.vmware.com>) installed.

The Linux install on the analysis system was performed just prior to attempting this assignment to help ensure that the machine was as clean as possible. The following tools were also installed to assist in the investigation:

TCPDump v3.7.1 (<http://www.tcpdump.org/release/tcpdump-3.7.1.tar.gz>)

LIBPCAP v7.1 (<http://www.tcpdump.org/release/libpcap-0.7.1.tar.gz>)

APPTRACE v0.1 (<ftp://ftp.stearns.org/pub/apprace>)

LSOF v4.64 (<ftp://ftp.cerias.purdue.edu/pub/tools/unix/sysutils/lsof/lsof.tar.gz>)

Binary Details

The screen dump below shows the initial steps that were taken to help identify the unknown binary:

© SANS Institute 2000 - 2002

```
root@wagnerdl:opt2/gcfa
File Edit Settings Help

[root@wagnerdl gcfa]# ls
sn.zip
[root@wagnerdl gcfa]# unzip sn.zip
Archive: sn.zip
  inflating: sn.dat
  extracting: sn.md5
[root@wagnerdl gcfa]# ls -li sn.dat
32770 sn.dat
[root@wagnerdl gcfa]# debugfs -R "stat <32770>" /dev/sdb1
debugfs 1.27 (8-Mar-2002)
Inode: 32770   Type: regular   Mode: 0666   Flags: 0x0   Generation: 141049
User:    0   Group:    0   Size: 399124
File ACL: 0   Directory ACL: 0
Links: 1   Blockcount: 792
Fragment: Address: 0   Number: 0   Size: 0
ctime: 0x3d751edf -- Tue Sep  3 16:43:11 2002
atime: 0x3cb58fd6 -- Thu Apr 11 09:29:58 2002
mtime: 0x3cb58fd6 -- Thu Apr 11 09:29:58 2002
BLOCKS:
(0-11):66053-66064, (IND):66065, (12-97):66066-66151
TOTAL: 99

[root@wagnerdl gcfa]# md5sum sn.dat
0e954f43fd73f56e812a7285f32e41d3  sn.dat
[root@wagnerdl gcfa]# file sn.dat
sn.dat: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), statically linked, stripped
[root@wagnerdl gcfa]# strings -a sn.dat > sn.str
[root@wagnerdl gcfa]#
```

To begin the investigation I unzipped the *sn.zip* archive which output two files, *sn.dat* and *sn.md5*. The *sn.dat* file appeared to be the binary I was supposed to analyze and the *sn.md5* file contained an MD5 hash of the *sn.dat* file. My primary objective at this point was to obtain MAC (Modify/Access/Create) time information from the *sn.dat* file before it could be modified.

I first used the command *ls -li sn.dat* to obtain the inode number for the unknown binary. As shown above the inode for the *sn.dat* file was 32770. I then used the command *debugfs -R "stat <32770>" /dev/sdb1* to obtain the file's MAC information. Note the use of the inode within the *debugfs* command. The device name of the partition where the file resides was also required, */dev/sdb1*.

Using the *debugfs* command I was able to collect the file's MAC information before it could inadvertently be changed by my investigation. The MAC times that were output are listed below:

```
Create time:  Tue Sep  3 16:43:11 2002
Access time:  Thu Apr 11 09:29:58 2002
Modify time:  Thu Apr 11 09:29:58 2002
```

In addition to the MAC information, the *debugfs* command also provided *sn.dat*'s file size and owner information. This is shown below:

```
Size: 399124
User: 0
Group: 0
```

Next I used the *md5sum* command to generate a MD5 hash for the *sn.dat* file to compare it against the hash in the *sn.md5* file and to assist in verifying the preservation of evidence throughout the investigation. Below is the output of the *md5sum* command:

```
0e954f43fd73f56e812a7285f32e41d3 sn.dat
```

I then used the *file* command to identify what type of file I was dealing with. The output of this command is shown below:

```
sn.dat: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
statically linked, stripped
```

The *file* command was able to identify the *sn.dat* file as a statically linked binary stripped of symbol information. This was very useful as it proved that the file was an application of some sort.

Finally, I issued the command *strings -a sn.dat > sn.str* which searched the *sn.dat* file for string values and redirected the results to the *sn.str* file for further investigation.

I manually parsed the *sn.str* file for information that could help to identify what the unknown file was or might be used for. I quickly observed the following interesting string values that would need to be researched in further detail:

```
ADMSniff %s <device> [HEADERSIZE] [DEBUG]
ex : admsniff le0
..ooOO The ADM Crew OOoo..
cant open pcap device :<
init_pcap : Unknown device type!
ADMSniff %s in libpcap we trust !
credits: ADM, mel , ^pretty^ for the mail she sent me
The_logz
@(#) $Header: pcap-linux.c,v 1.15 97/10/02 22:39:37 leres Exp $ (LBL)
@(#) $Header: pcap.c,v 1.29 98/07/12 13:15:39 leres Exp $ (LBL)
@(#) $Header: savefile.c,v 1.37 97/10/15 21:58:58 leres Exp $ (LBL)
@(#) $Header: bpf_filter.c,v 1.33 97/04/26 13:37:18 leres Exp $ (LBL)
1997-12-20
+45 3325-6543
+45 3122-6543
keld@dkuug.dk
Keld Simonsen
ISO/IEC 14652 i18n FDCC-set
C/o Keld Simonsen, Skt. Jorgens Alle 8, DK-1615 Kobenhavn V
ISO/IEC JTC1/SC22/WG20 - internationalization
GCC: (GNU) 2.96 20000731 (Red Hat Linux 7.1 2.96-97)
```

Program Description and Forensic Details

In the previous section, I was able to discover many important clues as to the identity of the *sn.dat* file. The *file* command was useful in identifying the file as a Linux binary application. The *debugfs* command was able to provide MAC time information that showed the file had most likely not been executed as the modify and access times (which were preserved when the file was zipped) were identical. The string output provided me with additional information as to what the purpose of the file might be. Of special interest were the following lines:

```
ex    : admsniff le0
ADMsni ff %s  in libpcap we trust !
```

The first line shown above appears to be an example of how the command is to be used. Based on this information and information from the second line, the string output leads me to believe that the program is called ADMsniff. The example also shows that ADMsniff takes a network device as its input, i.e. le0. Below are two additional lines that I believe help to identify the purpose of the application:

```
@(#) $Header: pcap-linux.c,v 1.15 97/10/02 22:39:37 leres Exp $ (LBL)
@(#) $Header: pcap.c,v 1.29 98/07/12 13:15:39 leres Exp $ (LBL)
```

The above lines show that the program makes use of the PCAP (Packet Capture) Library. All of this information can be used to infer that the application is intended to be used as some type of network sniffer.

My next course of action was to execute the program to see what it did. This needed to be done in a controlled manner so that I could monitor what system resources the application attempted to use. To assist in the monitoring process I made use of *apptrace*. The screen dump below shows me setting the *sn.dat* file as executable and then configuring *apptrace* to monitor the application's resource usage:

```
root@wagnerdl:/opt2/gcfa
File Edit Settings Help
[root@wagnerdl gcfa]# ls
sn.dat sn.md5 sn.str sn.zip
[root@wagnerdl gcfa]# chmod u+x sn.dat
[root@wagnerdl gcfa]# appttrace sn.dat
[root@wagnerdl gcfa]# ll
total 604
lrwxrwxrwx 1 root root 23 Sep 3 17:07 sn.dat -> /usr/local/bin/appttrace
-rwxrwxrwx 1 root root 399124 Apr 11 09:29 sn.dat.orig
-rwxrwxrwx 1 root root 37 Apr 11 09:29 sn.md5
-rw-r--r-- 1 root root 28211 Sep 3 17:06 sn.str
-r-xr-xr-x 1 root root 175185 Aug 30 09:27 sn.zip
[root@wagnerdl gcfa]#
```

To monitor the machine's network interface, I ran *tcpdump* and redirected its output to a log file using the following command:

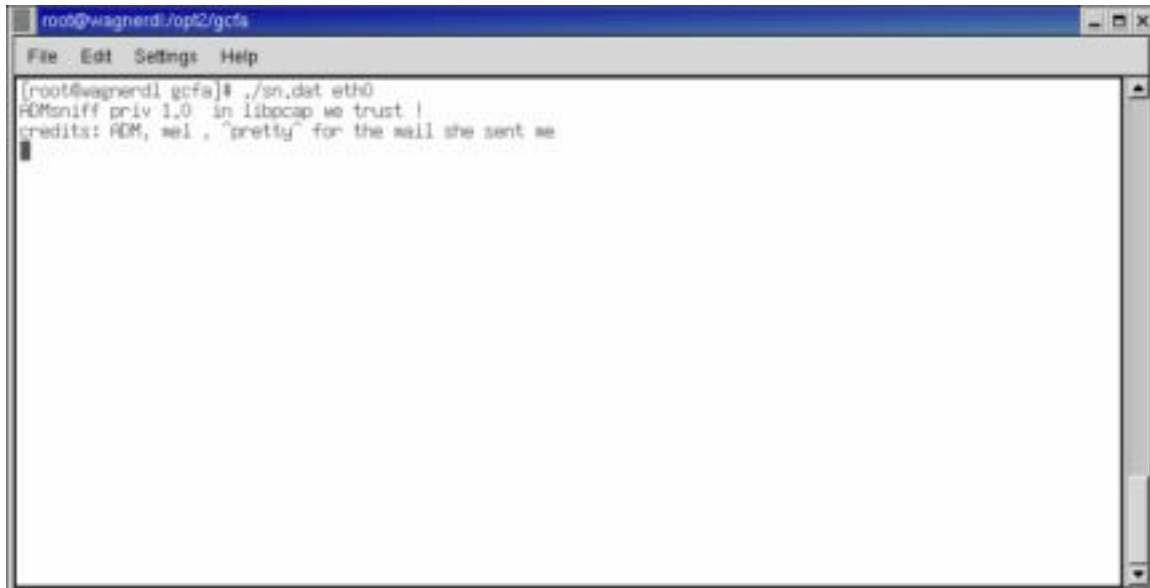
```
tcpdump -i eth0 > tcpdump.log
```

Using *tcpdump* I would be able to verify what, if any, network communications the *sn.dat* application attempted to make. Now that I had *tcpdump* and *appttrace* configured and ready to go, I executed the application see what it did. This is shown in the screen capture below:

```
root@wagnerdl:/opt2/gcfa
File Edit Settings Help
[root@wagnerdl gcfa]# ./sn.dat
ADMSniff priv 1.0 <device> [HEADERSIZE] [DEBUG]
ex : admsniff [a0]
..0000 The ADM Crew 0000..
[root@wagnerdl gcfa]#
```

Executing *sn.dat* without any parameters caused the program to display help information. Before doing anything else, I checked the *appttrace* and *tcpdump* logs to see if the application attempted anything besides printing help

information, which, according to the logs, it didn't. Next I ran *sn.dat* again but this time with *eth0* for the device parameter. This resulted in the following output:



```
root@wagnerd:/opt2/gcfa
File Edit Settings Help
[root@wagnerd1 gcfa]# ./sn.dat eth0
ADMoniff priv 1.0 in libcap we trust |
credits: ADM, sei, 'pretty' for the mail she sent me
```

Immediately after executing the *sn.dat* file with *eth0* for its input parameter I checked both the *tcpdump* and *appttrace* logs to see what the application might be doing. While *tcpdump* showed that the application had not attempted to send anything via the network, the *appttrace* log showed that it had created a file called "The_I0gz" in the current directory. Appendix C shows a listing from the *appttrace* log after letting the application sit idle for several minutes.

I checked the contents of the file *The_I0gz* that *sn.dat* had created and found that it was currently empty. I decided to test my theory that the program was acting as a network sniffer by pinging another box on the analysis network. While *tcpdump* showed the ICMP packets, nothing showed up in the *The_I0gz* file.

I then tried to FTP to another machine on the analysis network and enter my username and password. Once again the traffic was detected by *tcpdump* but this time it also appeared in the *The_I0gz* file as shown below:

```
--=[ 192.168.200.1:21 --> 192.168.200.128:32772 ]--
.....[.....220 Microsoft FTP
Service.....[.....331 Password required for <user>.....\.....230
User <user> logged

in.....\.....215 Windows_NT.....\.....F227 Entering Passive Mode
(192,168,200,1,5,230).....\.....G125 Data connection already open;
Transfer

starting.....\.....K226 Transfer complete.....\)...=221
.....\)...=.

--=[ 192.168.200.128:32772 --> 192.168.200.1:21 ]--
```

```

.....[.....[.USER
<user>.....[.....[.PASS

<password>.....\.....\..SYST.....\.....F..\..PASV.
.....F..\.....G..\..LIST.....K..\.....K..\.....=...\..QUIT
.....

=..\)\.....=...\)\.

```

The above information appears to be a decoded trace of my FTP session. The trace shows both sides of the conversation and includes the username and password I tested with. This information could prove very dangerous if it fell into the wrong hands.

I then tried using both telnet and SSH to access another box on my analysis network to see if *sn.dat* would log data from those applications as well. While the remote box did not have telnet running, I wanted to see if something would still show up in the *sn.dat* log file. Below are the results taken from the *The_I0gz* file. Note, SSH, like ICMP, did not appear to have been captured.

```

--=[ 192.168.200.1:23 --> 192.168.200.128:32782 ]==
.....

```

Next I tried connecting between two different machines on the same hub as the machine running the *sn.dat* application using FTP and telnet. The intent was to see if *sn.dat* was able to promiscuously capture traffic on the network like it did when traffic was initiated from the same box as the application. After performing the tests I checked the *The_I0gz* file and found that the test sessions had been captured.

This final test showed that the application might have configured the network interface to be in promiscuous mode and was monitoring all network traffic. To verify my hypothesis I decided to reboot my machine as the use of *tcpdump* would also have switched the interface into promiscuous mode. I needed to prove that executing *sn.dat* would do this as well.

I rebooted the machine, immediately ran *ifconfig*, and verified that interface eth0 was not in promiscuous mode. I then executed *sn.dat* and ran *ifconfig* again. The interface had now switched to promiscuous mode proving that the *sn.dat* file was indeed acting as a sniffer.

The image below shows the output of executing *ifconfig*. Note the keyword PROMISC for interface eth0. This means the interface is configured in promiscuous mode and has the ability to capture all traffic that it can physically see on the network.

```
root@wagnerdl:~  
File Edit Settings Help  
[root@wagnerdl root]# ifconfig eth0  
eth0      Link encap:Ethernet  HWaddr 00:50:56:F1:88:1B  
          inet addr:192.168.200.128  Bcast:192.168.200.255  Mask:255.255.255.0  
          UP BROADCAST NOTRAILERS RUNNING PROMISC MULTICAST  MTU:1500  Metric:1  
          RX packets:9 errors:0 dropped:0 overruns:0 frame:0  
          TX packets:2 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:100  
          RX bytes:1395 (1.3 Kb)  TX bytes:650 (650.0 b)  
          Interrupt:10 Base address:0x10a0  
[root@wagnerdl root]#
```

Next, I restarted the *sn.dat* binary and used the *ps* command to identify *sn.dat*'s process ID. Using the results of the *ps* command I executed */sof* to verify what resources *sn.dat* was using. The output of the */sof* command is listed below:

```
COMMAND      PID USER   FD      TYPE DEVICE  SIZE  NODE NAME  
sn.dat.or 3249 root   cwd      DIR    8,17   4096 32769 /opt2/gcfa  
sn.dat.or 3249 root   rtd      DIR     8,2   4096    2 /  
sn.dat.or 3249 root   txt      REG    8,17 399124 32770  
/opt2/gcfa/sn.dat.orig  
sn.dat.or 3249 root    0u      CHR   136,0    2 /dev/pts/0  
sn.dat.or 3249 root    1u      CHR   136,0    2 /dev/pts/0  
sn.dat.or 3249 root    2u      CHR   136,0    2 /dev/pts/0  
sn.dat.or 3249 root    3u     sock     0,0 18412 can't identify  
protocol  
sn.dat.or 3249 root    4w     REG    8,17    0 32775 /opt2/gcfa/The_10gz
```

Finally, I went step-by-step through the *appttrace* and *tcpdump* log files that I had obtained throughout the testing process. From the log files I was able to observe that when *sn.dat* was executed using a network device as its parameter it consistently performed the following actions:

- 1) Opened the network interface specified on the command line

```
socket(PF_INET, SOCK_PACKET, 0x300 /* IPPROTO_??? */) = 3
```

- 2) Wrote to the display

```
write(1, "ADMsniff priv 1.0 in libpcap we"..., 41) = 41
```

```
write(1, "credits: ADM, mel , ^pretty^ for"... , 54) = 54
```

3) Opened a log file

```
open("The_l0gz", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 4
```

4) Captured data from the network

```
recvfrom(3,  
"\0PV\300\0\1\0PV\300\30\276\10\6\0\1\10\0\6\4\0\2\0PV\300"... ,  
1564, 0, {sin_family=AF_UNIX, path="eth0"}, [18]) = 60
```

5) Wrote data to The_l0gz

```
write(4, "\n--[ 192.168.200.128:32772 --> "... , 322) = 322
```

6) Executed until terminated

```
--- SIGINT (Interrupt) ---  
+++ killed by SIGINT +++
```

From the information collected so far I was able to determine that the *sn.dat* file required a network device name as an input parameter and acted as a network sniffer when executed. It appeared to be selective as to what it captured as ICMP and SSH traffic were ignored. It decoded and saved what it captured from the network to a file called *The_l0gz* in the directory where it was executed. And, while testing, it did not attempt to make outbound connections from the machine it was executed on.

Program Identification

I was able to find what appeared to be the source code for the *sn.dat* file on the internet by performing a Google search (<http://www.google.com>) with the keyword ADMsniff. The first result took me to a web page on SecurityFocus Online (<http://online.securityfocus.com/tools/215/scoreit>) that identified the tool as a sniffer. On that page was a link for <http://adm.freelsd.net> which provided access to "The ADM CreW official ftp site" (<http://adm.freelsd.net/ADM>). This site allowed me to download the source code for the ADMsniff application (<http://adm.freelsd.net/ADM/ADMsniff.tar.gz>).

I downloaded the source code and compiled it on my RedHat 7.3 system by issuing the *make* command within the source subdirectory. The resulting file was much smaller than *sn.dat* file as it was only 29511 bytes long where the original was 399124 bytes in length.

I looked back at my notes taken earlier in the investigation and noticed that when using the *file* command the *sn.dat* file had been identified as a statically linked binary with symbol information stripped as shown below:

```
sn.dat: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
statically linked, stripped
```

I ran the *file* command against the binary I had just created to compare the results and verified my suspicions. The newly created file was dynamically linked and not stripped of symbol information. Both static and stripped are link time options that would create a file of different size than the *sn.dat* file if not set.

I modified the project's Makefile by adding *-static -s* to the CFLAGS line. I then rebuilt the project using the *make* command and reexamined the resulting object file. The new file was still smaller than the original as it was only 389104 bytes in length.

I made one last attempt at creating a duplicate of the original file by enabling log compression by making the changes described at the beginning of the project's Makefile. I doubted this would make any difference as the applications log files did not appear to be compressed but it was worth a try.

This change resulted in a file that was larger than the original. The new file was 431796 bytes in length so I disabled the compression setting within the Makefile and rebuilt the application with only the *-static* and *-s* parameters enabled.

My conclusion at this point was that the original file was of different size due to it being built on a different system with different libraries. I was compiling the application on RedHat Linux 7.3 but when I had executed the *strings* command it had shown that *sn.dat* had been created on a machine running RedHat Linux 7.1 as shown below:

```
GCC: (GNU) 2.96 20000731 (Red Hat Linux 7.1 2.96-97)
```

Even though the new and old files would obviously not match up due to differences in content I ran *md5sum* to create a hash of the new file. This hash and the original contained in the *sn.md5* file are shown below for comparison purposes:

```
0e954f43fd73f56e812a7285f32e41d3  sn
c41f249806ea719b97b09f0db7878dd4  ADMsniff-1
```

The only way I could verify my conclusion without attempting to recreate a machine I had no knowledge about other than its kernel version was to execute the newly created binary and compare its actions against the original. I configured *appttrace* to monitor the new binary as shown below:

```

root@wagnerdl:opt2/ADMsniff
File Edit Settings Help
[root@wagnerdl ADMsniff]# ls
1 bpf.h libpcap-0.4 libpcap.a pcap.h tcp.h
ADMsniff-1 ip.h libpcap-0.4.tar Makefile README thesniff.c
[root@wagnerdl ADMsniff]# appttrace ADMsniff-1
[root@wagnerdl ADMsniff]# ll
total 1012
-rw-r--r-- 1 root root 738 Sep 3 18:29 1
lrwxrwxrwx 1 root root 23 Sep 3 18:47 ADMsniff-1 -> /usr/local/bin/appttrace
-rwxr-xr-x 1 root root 389104 Sep 3 18:42 ADMsniff-1.orig
-r--r--r-- 1 root root 8447 Jan 19 1999 bpf.h
-rw-r--r-- 1 root root 486 May 7 1999 ip.h
drwxr-xr-x 6 root root 4096 Sep 3 18:42 libpcap-0.4
-rw-r--r-- 1 root root 487424 May 7 1999 libpcap-0.4.tar
-rw-r--r-- 1 root root 86930 Sep 3 18:42 libpcap.a
-rw-r--r-- 1 root root 740 Sep 3 18:42 Makefile
-rw-r--r-- 1 root root 4908 Jan 19 1999 pcap.h
-rw-r--r-- 1 root root 1072 May 30 1999 README
-rw-r--r-- 1 root root 1491 Jan 19 1999 tcp.h
-rw-r--r-- 1 root root 8432 May 11 1999 thesniff.c
[root@wagnerdl ADMsniff]#

```

Using *appttrace* and *tcpdump*, I first executed the new binary with no parameters and then again using eth0 as the device name. These commands and their output are shown below:

```

root@wagnerdl:opt2/ADMsniff
File Edit Settings Help
[root@wagnerdl ADMsniff]# ./ADMsniff-1
ADMsniff priv 1.0 <device> [HEADERSIZE] [DEBUG]
ex : admniff le0
..oo00 The ADM Crew 00oo..
[root@wagnerdl ADMsniff]# ./ADMsniff-1 eth0
ADMsniff priv 1.0 in libpcap we trust !
credits: ADM, mel , ^pretty^ for the mail she sent me

```

I then performed the same ICMP, FTP, telnet, and SSH tests as I did while evaluating the *sn.dat* file. Analyzing the *appttrace* and *tcpdump* logs showed that both programs acted identically. Below I've included the output of running *lsOf* on the new application which shows it using the same system resources as *sn.dat*:

COMMAND	PID	USER	FD	TYPE	DEVICE	SIZE	NODE	NAME
ADMsniff-	4645	root	cwd	DIR	8,17	4096	65537	/opt2/ADMsniff
ADMsniff-	4645	root	rtd	DIR	8,2	4096	2	/

```

ADMsniiff- 4645 root  txt      REG    8,17 389104 65542
/opt2/ADMsniiff/ADMsniiff-1.orig
ADMsniiff- 4645 root    0u    CHR   136,0          2 /dev/pts/0
ADMsniiff- 4645 root    1u    CHR   136,0          2 /dev/pts/0
ADMsniiff- 4645 root    2u    CHR   136,0          2 /dev/pts/0
ADMsniiff- 4645 root    3u  sock    0,0        23539 can't identify
protocol
ADMsniiff- 4645 root    4w    REG    8,17          0 65551
/opt2/ADMsniiff/The_10gz

```

My belief is that the *sn.dat* application is the same as ADMsniiff. I believe the application files differ in size and MD5 hashes because they were compiled on machines with a different OS and libraries installed. Even with these inconsistencies, the applications appear to function in the same manner and I believe are the same application.

Finally, I do not believe that the *sn.dat* file was executed on the compromised system. The fact that the MAC access time of the original file matches the modify time shows that the file was not copied or executed since it was created. Unless the binary was copied to the system more than once and a duplicate executed or the MAC times were adjusted, the application appears not to have been run on the compromised machine.

Legal Implications

From the analysis above I was not able to provide conclusive evidence as to whether or not the *sn.dat* file was executed on the compromised machine. As discussed in the previous section, the file's MAC access time showed that it had not been accessed since it was created. With the evidence at hand there is no way to prove that the application was or was not executed on the machine. Even though the previous analysis shows that the analyzed file had not been executed, it does not rule out the possibility that another copy had been installed and executed or that the MAC times had been adjusted.

Legal implications related to the installation of the *sn.dat* file are dependent on many variables including the intent of the person who installed the application (i.e. fraud, extortion,...) and whether the compromised system is defined as a "protected computer" under the Computer Fraud and Abuse Act (18 U.S.C. Section 1030(e)(2)). Documents pertaining to this and other federal laws may be found at the web page of the Office of the Law Revision Counsel on the U.S. House of Representatives web site (<http://uscode.house.gov>).

The individual that installed the sniffer could be prosecuted under the Computer Fraud and Abuse Act if they are in violation of any of the items outlined in 18 U.S.C. Section 1030(a). In addition, if the application was executed, the trespasser may be in violation of the Federal Wiretap Statute (18 U.S.C. Section 2511).

Even if the individual that installed the application on the compromised system is not in violation of the laws listed above, they may be punishable under an organization's acceptable use policy. Such a policy may apply if the compromised system is owned by the intruder's employer and, depending on the policy, may result in termination of the employee.

Interview Questions

The following are questions that could be useful in interrogating a person suspected of installing the *sn.dat* application on the compromised system:

- 1) Are you familiar with the Linux operating system?
- 2) Have you ever compiled an application on a RedHat Linux 7.1 system?
- 3) Did you copy a file to the compromised system on April 11th, 2002?
- 4) Why did you copy the file to the system?
- 5) What was the name of the file?
- 6) Did you build the file?
- 7) What version of compiler did you use?
- 8) What was the purpose of the file?
- 9) Did you execute the file? Why or why not?

Additional Information

The following are links that proved useful in performing my investigation:

- 1) Google – <http://www.google.com>
- 2) SecurityFocus web site – <http://online.securityfocus.com/tools/215/scoreit>
- 3) The ADM web site – <http://adm.freelsd.net/ADM>
- 4) U.S. Code – <http://uscode.house.gov>
- 5) Apptrace – <http://www.stearns.org/apprtrace>
- 6) TCPDump – <http://www.tcpdump.org>
- 7) LSOF – <ftp://vic.cc.purdue.edu/pub/tools/unix/lsof>

Part 3 – Legal Issues of Incident Handling (Wiretap Statute)

U.S. Code Section 2511 of the Electronic Communications Privacy Act of 1986 is commonly known as the Federal Wiretap Statute (18 U.S.C. Section 2511). Documents pertaining to this and other federal laws may be found at the web page of the Office of the Law Revision Counsel on the U.S. House of Representatives web site (<http://uscode.house.gov>).

The Wiretap Statute is primarily concerned with the unauthorized interception of various communications. Violations can result in serious repercussions which are outlined in subsections 4 and 5 of the statute. While the statute is very stringent, it does outline several exceptions where monitoring is permissible.

Section (2)(a)(i) of the Wiretap Statute grants a system administrator/owner the right to monitor his systems with the intent to protect said systems from harm (18 U.S.C. Section 2511(2)(a)(i)). It also protects an administrator from prosecution due to the incidental interception of communications while performing standard duties directly related to his position.

While the Wiretap Statute provides system administrators significant authority to protect their systems, it does not grant a system administrator unlimited rights. Unnecessarily targeting individuals would not be considered permissible (18 U.S.C. Section 2511(2)(a)(i)). All monitoring should be done in a controlled manner so that the collected information directly relates to the protection of an organization's property.

In addition, the Wiretap Statute grants a system administrator/owner additional rights in the monitoring of users of a system that have consented to such monitoring (18 U.S.C. Section 2511(2)(d)). To make this exception affective, an organization must provide notification that such monitoring may be in affect. This can be accomplished via the use of login banners on protected systems and making authorized users sign acceptable use policies.

Unfortunately, login banners are not entirely affective as there are only a limited number of ports on a computer system that will allow for their display. In addition, acceptable use policies will most likely not provide protection from the misuse of a system by an external entity that was not required to sign or adhere to the policy.

It should be noted that section 217 of the USA Patriot Act of 2001 amended the Wiretap Statute to grant law enforcement additional authority while assisting the owner of a system in cyber crime investigations (18 U.S.C. Section 2511(2)(i)). Section 217 also helps to identify someone gaining unauthorized access to a system as a "computer trespasser" (18 U.S.C. Section 2510(21)(A)). The USA Patriot Act can be downloaded from the Library of Congress web site at

<http://thomas.loc.gov> under "Public Laws by Law Number". The Patriot Act is Public Law Number 107-56.

© SANS Institute 2000 - 2002, Author retains full rights.

Appendices

Appendix A – Win2k/XP incident response script

ir-win2k xp.cmd

```
@echo off

echo -----
echo date -- Current date and time
echo -----
echo.
\win2k_xp\date

echo.
echo.
echo.
echo -----
echo hostname -- System name
echo -----
echo.
\win2k_xp\hostname

echo.
echo.
echo.
echo -----
echo id -- Current user info
echo -----
echo.
\win2k_xp\id

echo.
echo.
echo.
echo -----
echo env -- Environment info
echo -----
echo.
\win2k_xp\env

echo.
echo.
echo.
echo -----
echo ps -ealW -- Process info
echo -----
echo.
\win2k_xp\ps -ealW

echo.
echo.
echo.
echo -----
echo netstat -arn -- Routing and connection info
echo -----
echo.
\win2k_xp\netstat -arn

echo.
echo.
echo.
echo -----
echo arp -a -- ARP table info
echo -----
echo.
```

```

\win2k_xp\arp -a

echo.
echo.
echo.
echo -----
echo net file -- Open shared files
echo -----
echo.
\win2k_xp\net file

echo.
echo.
echo.
echo -----
echo net session -- Open sessions
echo -----
echo.
\win2k_xp\net session

echo.
echo.
echo.
echo -----
echo net share -- Open shares
echo -----
echo.
\win2k_xp\net share

echo.
echo.
echo.
echo -----
echo net start -- Running services
echo -----
echo.
\win2k_xp\net start

echo.
echo.
echo.
echo -----
echo net use -- Open connections
echo -----
echo.
\win2k_xp\net use

echo.
echo.
echo.
echo -----
echo psinfo -h -s -d -- System info
echo -----
echo.
\win2k_xp-misc\psinfo -h -s -d

echo.
echo.
echo.
echo -----
echo psloggedon -- Logged on users
echo -----
echo.
\win2k_xp-misc\psloggedon

echo.
echo.
echo.
echo -----
echo pslist -- Process info
echo -----

```

```

echo.
\win2k_xp-misc\pslist

echo.
echo.
echo.
echo -----
echo pservice -- Service info
echo -----
echo.
\win2k_xp-misc\pservice

echo.
echo.
echo.
echo -----
echo psfile -- Open shared file info
echo -----
echo.
\win2k_xp-misc\psfile

echo.
echo.
echo.
echo -----
echo handle -- File usage info
echo -----
echo.
\win2k_xp-misc\handle

echo.
echo.
echo.
echo -----
echo fport -- Connection info
echo -----
echo.
\win2k_xp-misc\fport

echo.
echo.
echo.
echo -----
echo psloglist system -- System log dump
echo -----
echo.
\win2k_xp-misc\psloglist system

echo.
echo.
echo.
echo -----
echo psloglist application -- Application log dump
echo -----
echo.
\win2k_xp-misc\psloglist application

echo.
echo.
echo.
echo -----
echo psloglist security -- Security log dump
echo -----
echo.
\win2k_xp-misc\psloglist security

```

Appendix B – Supporting media imaging files

md5.org

```
145e0b8df7ce54c3ef0a82e724059b4f /dev/hda1
165387cf9c056761d977eea268b91eeb /dev/hda2
```

md5.txt

```
145e0b8df7ce54c3ef0a82e724059b4f c.dd
165387cf9c056761d977eea268b91eeb d.dd
0821c57ac67c526717ca7cb60823cab3 live.txt
4862b0a7578a59b0b6f05eb4bbaf3cf1 md5.org
accf9ee507eb0d0ecb36481f5b58f674 ram.dd
```

live.txt

```
-----
date -- Current date and time
-----
```

```
Tue Aug 27 18:37:09 2002
```

```
-----
hostname -- System name
-----
```

```
MCON
```

```
-----
id -- Current user info
-----
```

```
uid=500 (Administrator) gid=544 (Administrators) groups=544 (Administrators)
```

```
-----
env -- Environment info
-----
```

```
!::=::\
!E:=E:\win2k_xp
!EXITCODE=00000000
ALLUSERSPROFILE=C:\Documents and Settings\All Users
APPDATA=C:\Documents and Settings\Administrator\Application Data
COMMONPROGRAMFILES=C:\Program Files\Common Files
COMPUTERNAME=MCON
COMSPEC=C:\WINDOWS\system32\cmd.exe
HOMEDRIVE=C:
HOMEPATH=\
LOGONSERVER=\\MCON
NUMBER_OF_PROCESSORS=1
OS=Windows_NT
PATH=/cygdrive/c/WINDOWS/system32:/cygdrive/c/WINDOWS:/cygdrive/c/WINDOWS/System32/Wbem:/
cygdrive/c/Program Files/MSDESQL7/BINN
PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH
PROCESSOR_ARCHITECTURE=x86
PROCESSOR_IDENTIFIER=x86 Family 6 Model 5 Stepping 1, GenuineIntel
PROCESSOR_LEVEL=6
PROCESSOR_REVISION=0501
PROGRAMFILES=C:\Program Files
PROMPT=$P$G
SESSIONNAME=Console
SYSTEMDRIVE=C:
```

```

SYSTEMROOT=C:\WINDOWS
TEMP=/cygdrive/c/DOCUME~1/ADMINI~1/LOCALS~1/Temp
TMP=/cygdrive/c/DOCUME~1/ADMINI~1/LOCALS~1/Temp
USERDOMAIN=MCON
USERNAME=Administrator
USERPROFILE=C:\Documents and Settings\Administrator
WINDIR=C:\WINDOWS
TERM=cygwin

```

```

-----
ps -ealW -- Process info
-----

```

PID	PPID	PGID	WINPID	TTY	UID	STIME	COMMAND
4	0	0	4	?	0	15:24:48	*** unknown ***
444	0	0	444	?	0	Aug 23	\SystemRoot\System32\smss.exe
524	0	0	524	?	0	Aug 23	
\\??C:\WINDOWS\system32\winlogon.exe							
568	0	0	568	?	0	Aug 23	C:\WINDOWS\system32\services.exe
580	0	0	580	?	0	Aug 23	C:\WINDOWS\system32\lsass.exe
744	0	0	744	?	0	Aug 23	C:\WINDOWS\system32\svchost.exe
792	0	0	792	?	0	Aug 23	C:\WINDOWS\System32\svchost.exe
1040	0	0	1040	?	0	Aug 23	C:\WINDOWS\system32\spoolsv.exe
1148	0	0	1148	?	0	Aug 23	C:\Program Files\Network
Associates\VirusScan\Avsynmgr.exe							
1332	0	0	1332	?	0	Aug 23	C:\Program Files\Network
Associates\VirusScan\VsStat.exe							
1348	0	0	1348	?	0	Aug 23	C:\Program Files\Network
Associates\VirusScan\Vshwin32.exe							
1364	0	0	1364	?	0	Aug 23	C:\Program Files\Common
Files\Network Associates\McShield\Mcshield.exe							
1480	0	0	1480	?	0	Aug 23	C:\Program Files\Network
Associates\VirusScan\Avconsol.exe							
1308	0	0	1308	?	0	11:17:22	C:\WINDOWS\Explorer.EXE
1740	0	0	1740	?	0	11:17:25	C:\Program
Files\Messenger\msmsgs.exe							
1908	0	0	1908	?	0	11:17:26	C:\Program
Files\MSDESQL7\Binn\sqlmangr.exe							
1184	0	0	1184	?	0	11:17:44	c:\program files\snmpc network
manager\snmpc32.exe							
1736	0	0	1736	?	0	11:17:44	c:\program files\snmpc network
manager\discagt.exe							
484	0	0	484	?	0	11:17:45	c:\program files\snmpc network
manager\hist32.exe							
424	0	0	424	?	0	18:25:35	e:\win2k_xp\CMD.EXE
1236	0	0	1236	?	0	18:37:02	C:\WINDOWS\system32\cmd.exe
632	0	0	632	?	0	18:37:02	E:\win2k_xp\nc.exe
1344	1	1344	1344	con	500	18:37:10	/cygdrive/e/win2k_xp/ps

```

-----
netstat -arn -- Routing and connection info
-----

```

Interface List

```

0x1 ..... MS TCP Loopback interface
0x2 ...00 c0 4f 6f 68 ec ..... 3Com 3C918 Integrated Fast Ethernet Controller (3C905B-TX
Compatible) - Packet Scheduler Miniport

```

Active Routes:

Network	Destination	Netmask	Gateway	Interface	Metric
0.0.0.0	0.0.0.0		<<GATEWAY>>	<<MCON>>	1
X.X.X.0	255.255.255.0		<<MCON>>	<<MCON>>	20
<<MCON>>	255.255.255.255		127.0.0.1	127.0.0.1	20
X.255.255.255	255.255.255.255		<<MCON>>	<<MCON>>	20
127.0.0.0	255.0.0.0		127.0.0.1	127.0.0.1	1
224.0.0.0	240.0.0.0		<<MCON>>	<<MCON>>	20
255.255.255.255	255.255.255.255		<<MCON>>	<<MCON>>	1

Default Gateway: <<GATEWAY>>
=====

Persistent Routes:
None

Route Table

Active Connections

Proto	Local Address	Foreign Address	State
TCP	0.0.0.0:135	0.0.0.0:0	LISTENING
TCP	0.0.0.0:165	0.0.0.0:0	LISTENING
TCP	0.0.0.0:166	0.0.0.0:0	LISTENING
TCP	0.0.0.0:167	0.0.0.0:0	LISTENING
TCP	0.0.0.0:168	0.0.0.0:0	LISTENING
TCP	0.0.0.0:445	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1025	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1179	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1181	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1220	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1221	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1223	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1229	0.0.0.0:0	LISTENING
TCP	0.0.0.0:3389	0.0.0.0:0	LISTENING
TCP	0.0.0.0:5000	0.0.0.0:0	LISTENING
TCP	<<MCON>>:139	0.0.0.0:0	LISTENING
TCP	<<MCON>>:165	<<MCON>>:1179	ESTABLISHED
TCP	<<MCON>>:168	<<MCON>>:1181	ESTABLISHED
TCP	<<MCON>>:1179	<<MCON>>:165	ESTABLISHED
TCP	<<MCON>>:1181	<<MCON>>:168	ESTABLISHED
TCP	<<MCON>>:1229	<<FA1>>:5000	ESTABLISHED
UDP	0.0.0.0:135	*:*	
UDP	0.0.0.0:162	*:*	
UDP	0.0.0.0:164	*:*	
UDP	0.0.0.0:445	*:*	
UDP	0.0.0.0:500	*:*	
UDP	0.0.0.0:1026	*:*	
UDP	0.0.0.0:1027	*:*	
UDP	0.0.0.0:1047	*:*	
UDP	0.0.0.0:1176	*:*	
UDP	0.0.0.0:1177	*:*	
UDP	0.0.0.0:1178	*:*	
UDP	0.0.0.0:1180	*:*	
UDP	<<MCON>>:123	*:*	
UDP	<<MCON>>:137	*:*	
UDP	<<MCON>>:138	*:*	
UDP	<<MCON>>:1900	*:*	
UDP	127.0.0.1:123	*:*	
UDP	127.0.0.1:1900	*:*	

arp -a -- ARP table info

Interface: <<MCON>> on Interface 2		
Internet Address	Physical Address	Type
<<GATEWAY>>	00-00-0c-37-95-9a	dynamic
<<FA1>>	00-06-5b-22-f2-66	dynamic

net file -- Open shared files

There are no entries in the list.

net session -- Open sessions

There are no entries in the list.

```
-----  
net share -- Open shares  
-----
```

Share name	Resource	Remark
D\$	D:\	Default share
ADMIN\$	C:\WINDOWS	Remote Admin
C\$	C:\	Default share
IPC\$		Remote IPC

The command completed successfully.

```
-----  
net start -- Running services  
-----
```

These Windows services are started:

Automatic Updates
AVSync Manager
Background Intelligent Transfer Service
COM+ Event System
Computer Browser
Cryptographic Services
DHCP Client
Distributed Link Tracking Client
DNS Client
Error Reporting Service
Event Log
Help and Support
IPSEC Services
Logical Disk Manager
McShield
Messenger
Network Connections
Network Location Awareness (NLA)
Plug and Play
Portable Media Serial Number
Print Spooler
Protected Storage
Remote Procedure Call (RPC)
Remote Registry
Secondary Logon
Security Accounts Manager
Server
Shell Hardware Detection
SSDP Discovery Service
System Event Notification
System Restore Service
Task Scheduler
TCP/IP NetBIOS Helper
Terminal Services
Themes
Upload Manager
WebClient
Windows Audio
Windows Management Instrumentation
Windows Time
Wireless Zero Configuration
Workstation

The command completed successfully.

```
net use -- Open connections
```

```
-----  
New connections will be remembered.  
There are no entries in the list.  
  
-----
```

```
psinfo -h -s -d -- System info  
-----
```

```
PsInfo 1.34 - local and remote system information viewer  
Copyright (C) 2001-2002 Mark Russinovich  
Sysinternals - www.sysinternals.com
```

```
Querying information for MCON...
```

```
System information for \\MCON:
```

```
Uptime:                4 days, 7 hours, 41 minutes, 32 seconds  
Kernel version:        Microsoft Windows XP, Uniprocessor Free  
Product type:          Professional  
Product version:       5.1  
Service pack:          0  
Kernel build number:   2600  
Registered organization: <<ORG>>  
Registered owner:      <<OWNER>>  
Install date:          2/19/2002, 11:23:35 AM  
Activation status:     Activated  
IE version:            6.0000  
System root:           C:\WINDOWS  
Processors:            1  
Processor speed:       350 MHz  
Processor type:        Intel Pentium II or Celeron  
Physical memory:       256 MB
```

Volume	Type	Format	Label	Size	Free	Free
A:	Removable					0%
C:	Fixed	NTFS		3.9 GB	851.4 MB	21%
D:	Fixed	NTFS		2.1 GB	1.1 GB	49%
E:	CD-ROM	CDFS	020827_0900	KB		0%

```
OS Hot Fix      Installed  
Q147222         2/19/2002  
Q309521         2/19/2002  
Q311889         2/19/2002  
Q311967         4/23/2002  
Q313450         8/20/2002  
Q313484         2/19/2002  
Q314147         2/19/2002  
Q314862         3/11/2002  
Q315000         2/19/2002  
Q315403         3/11/2002  
Q317277         3/25/2002  
Q318138         6/13/2002  
Q319580         4/23/2002  
Q326830         8/23/2002
```

```
Applications:
```

```
Adobe Acrobat 5.0 5.0  
Check Point Management Clients 4.1  
Check Point Management Clients 4.1 SP5  
Check Point Management Clients 4.1 SP6  
Java 2 Runtime Environment Standard Edition v1.3.1_02  
LiveReg (Symantec Corporation) 2.1.0.1419  
LiveUpdate 1.6 (Symantec Corporation)  
MSDE  
McAfee VirusScan 4.5.1  
SNMPc Network Manager  
UltimateZip 2.6 2.6  
WebFldrs XP 9.50.5318  
Websense Reporter  
Windows XP Application Compatibility Update[Q313484]  
Windows XP Application Compatibility Update[Q319580]  
Windows XP Hotfix (SP1) [See Q309521 for more information]
```

Windows XP Hotfix (SP1) [See Q311889 for more information]
 Windows XP Hotfix (SP1) [See Q311967 for more information]
 Windows XP Hotfix (SP1) [See Q313450 for more information]
 Windows XP Hotfix (SP1) [See Q314147 for more information]
 Windows XP Hotfix (SP1) [See Q314862 for more information]
 Windows XP Hotfix (SP1) [See Q315000 for more information]
 Windows XP Hotfix (SP1) [See Q315403 for more information]
 Windows XP Hotfix (SP1) [See Q317277 for more information]
 Windows XP Hotfix (SP1) [See Q318138 for more information]
 Windows XP Hotfix (SP1) [See Q326830 for more information]

 psloggedon -- Logged on users

PsLoggedOn v1.21 - Logon Session Displayer
 Copyright (C) 1999-2000 Mark Russinovich
 SysInternals - www.sysinternals.com

Users logged on locally:
 <Unknown> NT AUTHORITY\LOCAL SERVICE
 <Unknown> NT AUTHORITY\NETWORK SERVICE
 8/27/2002 11:17:21 AM MCON\Administrator
 <Unknown> NT AUTHORITY\SYSTEM

No one is logged on via resource shares.

 pslist -- Process info

PsList 1.21 - Process Information Lister
 Copyright (C) 1999-2002 Mark Russinovich
 Sysinternals - www.sysinternals.com

Process information for MCON:

Name	Pid	Pri	Thd	Hnd	Mem	User Time	Kernel Time	Elapsed Time
Idle	0	0	1	0	20	0:00:00.000	103:25:13.180	0:00:00.000
System	4	8	46	237	216	0:00:00.000	0:00:21.991	0:00:00.000
smss	444	11	3	21	344	0:00:00.010	0:00:00.270	103:41:06.884
csrss	500	13	12	366	3296	0:00:06.980	0:00:22.382	103:40:58.552
winlogon	524	13	20	488	1896	0:00:04.626	0:00:08.922	103:40:57.921
services	568	9	17	295	3376	0:00:03.084	0:00:09.313	103:40:57.540
lsass	580	9	21	311	1536	0:00:01.351	0:00:02.022	103:40:57.520
svchost	744	8	9	243	3552	0:00:00.510	0:00:00.941	103:40:56.279
svchost	792	8	65	1167	19788	0:04:48.524	0:00:49.631	103:40:56.078
svchost	864	8	5	81	2724	0:00:00.120	0:00:00.200	103:40:54.456
svchost	884	8	15	164	3548	0:00:00.110	0:00:00.120	103:40:54.276
spoolsv	1040	8	14	158	4424	0:00:00.230	0:00:00.230	103:40:53.054
Avsynmgr	1148	8	4	88	3204	0:00:02.012	0:00:00.610	103:40:52.483
VSSStat	1332	8	2	60	3332	0:00:00.410	0:00:00.851	103:40:48.899
vshwin32	1348	8	6	99	6612	0:00:01.832	0:00:00.480	103:40:48.518
McshIELD	1364	13	16	122	5448	0:02:17.167	0:00:26.117	103:40:47.096
Avconsol	1480	8	2	66	3900	0:00:01.171	0:00:01.802	103:40:44.633
explorer	1308	8	10	266	12080	0:00:03.715	0:00:10.114	7:20:05.134
msmsgs	1740	8	2	118	1488	0:00:00.100	0:00:00.230	7:20:01.940
sqlmangr	1908	8	2	50	2512	0:00:00.370	0:00:01.021	7:20:01.629
snmpc32	1184	8	2	75	12980	0:00:03.565	0:00:03.234	7:19:43.353
discagt	1736	8	2	47	4332	0:00:01.852	0:00:01.532	7:19:43.263
hist32	484	8	2	48	4728	0:00:00.180	0:00:00.160	7:19:42.672
CMD	424	8	1	25	1696	0:00:00.030	0:00:00.090	0:11:52.734
cmd	1236	8	1	19	1044	0:00:00.180	0:00:00.600	0:00:25.296
nc	632	8	1	25	1316	0:00:00.020	0:00:00.090	0:00:25.266
wmiprvse	1692	8	11	146	3892	0:00:00.170	0:00:00.260	0:00:09.283
pslist	1280	13	2	68	1336	0:00:00.020	0:00:00.060	0:00:00.160

psservice -- Service info

PsService v1.01 - local and remote services viewer/controller
Copyright (C) 2001 Mark Russinovich
Sysinternals - www.sysinternals.com

SERVICE_NAME: Alerter

DISPLAY_NAME: Alerter

Notifies selected users and computers of administrative alerts. If the service is stopped, programs that use administrative alerts will not

receive them. If this service is disabled, any services that explicitly depend on it will fail to start.

TYPE : 20 WIN32_SHARE_PROCESS
STATE : 1 STOPPED
(NOT_STOPPABLE,NOT_PAUSABLE,IGNORES_SHUTDOWN)
WIN32_EXIT_CODE : 1077 (0x435)
SERVICE_EXIT_CODE : 0 (0x0)
CHECKPOINT : 0x0
WAIT_HINT : 0x0

SERVICE_NAME: ALG

DISPLAY_NAME: Application Layer Gateway Service

Provides support for 3rd party protocol plug-ins for Internet Connection Sharing and the Internet Connection Firewall

TYPE : 10 WIN32_OWN_PROCESS
STATE : 1 STOPPED
(NOT_STOPPABLE,NOT_PAUSABLE,IGNORES_SHUTDOWN)
WIN32_EXIT_CODE : 1077 (0x435)
SERVICE_EXIT_CODE : 0 (0x0)
CHECKPOINT : 0x0
WAIT_HINT : 0x0

SERVICE_NAME: AppMgmt

DISPLAY_NAME: Application Management

Provides software installation services such as Assign, Publish, and Remove.

TYPE : 20 WIN32_SHARE_PROCESS
STATE : 1 STOPPED
(NOT_STOPPABLE,NOT_PAUSABLE,IGNORES_SHUTDOWN)
WIN32_EXIT_CODE : 1077 (0x435)
SERVICE_EXIT_CODE : 0 (0x0)
CHECKPOINT : 0x0
WAIT_HINT : 0x0

.....

psfile -- Open shared file info

PsFile v1.01 - local and remote network file lister
Copyright (C) 2001 Mark Russinovich
Sysinternals - www.sysinternals.com

No files opened remotely on MCON.

handle -- File usage info

Handle v2.01

Copyright (C) 1997-2001 Mark Russinovich

Sysinternals - www.sysinternals.com

System pid: 4 NT AUTHORITY\SYSTEM

378: File C:\pagefile.sys

37c: File C:\WINDOWS\system32\config\software

```

380: File      C:\WINDOWS\system32\config\SECURITY
388: File      C:\WINDOWS\system32\config\SECURITY.LOG
394: File      C:\WINDOWS\system32\config\software.LOG
39c: File      C:\WINDOWS\system32\config\system
3a0: File      C:\WINDOWS\system32\config\system.LOG
3a8: File      C:\WINDOWS\system32\config\default
3ac: File      C:\WINDOWS\system32\config\default.LOG
3b0: File      C:\hiberfil.sys
3b4: File      C:\WINDOWS\system32\config\SAM
3b8: File      C:\WINDOWS\system32\config\SAM.LOG
3c8: File      C:\Documents and Settings\NetworkService\ntuser.dat.LOG
3cc: File      C:\Documents and Settings\NetworkService\Local Settings\Application
Data\Microsoft\Windows\UsrClass.dat
3d4: File      C:\Documents and Settings\NetworkService\NTUSER.DAT
3d8: File      C:\Documents and Settings\NetworkService\Local Settings\Application
Data\Microsoft\Windows\UsrClass.dat.LOG
3dc: File      C:\Documents and Settings\LocalService\NTUSER.DAT
3e0: File      C:\Documents and Settings\LocalService\ntuser.dat.LOG
3e4: File      C:\Documents and Settings\LocalService\Local Settings\Application
Data\Microsoft\Windows\UsrClass.dat
3e8: File      C:\Documents and Settings\LocalService\Local Settings\Application
Data\Microsoft\Windows\UsrClass.dat.LOG
4bc: File      C:\Documents and Settings\Administrator\NTUSER.DAT
518: File      C:\Documents and Settings\Administrator\Local Settings\Application
Data\Microsoft\Windows\UsrClass.dat
5f4: File      C:\WINDOWS\CSC\00000001
644: File      C:\Documents and Settings\Administrator\Local Settings\Application
Data\Microsoft\Windows\UsrClass.dat.LOG
650: File      C:\Documents and Settings\Administrator\NTUSER.DAT.LOG
8bc: File      C:\System Volume Information\_restore{B6C2779F-C4DE-4C6A-A439-
A748775A9FB5}\RP204\change.log

```

```

-----
smss.exe pid: 444 NT AUTHORITY\SYSTEM
8: File      C:\WINDOWS
24: File     C:\WINDOWS\system32
-----

```

```

csrss.exe pid: 500 NT AUTHORITY\SYSTEM
c: File      C:\WINDOWS\system32
40: Section   \NLS\NlsSectionUnicode
44: Section   \NLS\NlsSectionLocale
48: Section   \NLS\NlsSectionCTYPE
4c: Section   \NLS\NlsSectionSortkey
50: Section   \NLS\NlsSectionSortTbls
544: File     C:\WINDOWS\system32\ega.cpi

```

.....

```

-----
fport -- Connection info
-----

```

FPort v2.0 - TCP/IP Process to Port Mapper
 Copyright 2000 by Foundstone, Inc.
<http://www.foundstone.com>

Pid	Process	Port	Proto	Path
744	svchost	-> 135	TCP	C:\WINDOWS\system32\svchost.exe
4	System	-> 139	TCP	
1184	snmpc32	-> 165	TCP	c:\program files\snmpc network manager\snmpc32.exe
1736	discagt	-> 166	TCP	c:\program files\snmpc network manager\discagt.exe
484	hist32	-> 167	TCP	c:\program files\snmpc network manager\hist32.exe
1184	snmpc32	-> 168	TCP	c:\program files\snmpc network manager\snmpc32.exe
4	System	-> 445	TCP	
792	svchost	-> 1025	TCP	C:\WINDOWS\System32\svchost.exe
1736	discagt	-> 1179	TCP	c:\program files\snmpc network manager\discagt.exe
484	hist32	-> 1181	TCP	c:\program files\snmpc network manager\hist32.exe
792	svchost	-> 1220	TCP	C:\WINDOWS\System32\svchost.exe
792	svchost	-> 1221	TCP	C:\WINDOWS\System32\svchost.exe
792	svchost	-> 1223	TCP	C:\WINDOWS\System32\svchost.exe
632	nc	-> 1229	TCP	E:\win2k_xp\nc.exe

```

792  svchost      -> 3389  TCP    C:\WINDOWS\System32\svchost.exe
884                                     -> 5000  TCP

632  nc           -> 123   UDP    E:\win2k_xp\nc.exe
1184 snmpc32       -> 123   UDP    c:\program files\snmpc network manager\snmpc32.exe
744  svchost      -> 135   UDP    C:\WINDOWS\system32\svchost.exe
792  svchost      -> 137   UDP    C:\WINDOWS\System32\svchost.exe
884                                     -> 138   UDP
1184 snmpc32       -> 162   UDP    c:\program files\snmpc network manager\snmpc32.exe
1736 discagt     -> 164   UDP    c:\program files\snmpc network manager\discagt.exe
484  hist32      -> 445   UDP    c:\program files\snmpc network manager\hist32.exe
1184 snmpc32       -> 500   UDP    c:\program files\snmpc network manager\snmpc32.exe
4    System      -> 1026  UDP
792  svchost      -> 1027  UDP    C:\WINDOWS\System32\svchost.exe
1736 discagt     -> 1047  UDP    c:\program files\snmpc network manager\discagt.exe
484  hist32      -> 1176  UDP    c:\program files\snmpc network manager\hist32.exe
792  svchost      -> 1177  UDP    C:\WINDOWS\System32\svchost.exe
792  svchost      -> 1178  UDP    C:\WINDOWS\System32\svchost.exe
792  svchost      -> 1180  UDP    C:\WINDOWS\System32\svchost.exe
4    System      -> 1900  UDP
1184 snmpc32       -> 1900  UDP    c:\program files\snmpc network manager\snmpc32.exe

```

```

-----
psloglist system -- System log dump
-----

```

PsLogList v2.2 - local and remote event log viewer
 Copyright (C) 2000-2001 Mark Russinovich
 Sysinternals - www.sysinternals.com

System log on \\MCON:

[1419] Service Control Manager

Type: INFORMATION

Computer: MCON

Time: 8/27/2002 6:37:26 PM ID: 7036

The WMI Performance Adapter service entered the stopped state.

[1418] Service Control Manager

Type: INFORMATION

Computer: MCON

Time: 8/27/2002 6:37:26 PM ID: 7036

The WMI Performance Adapter service entered the running state.

[1417] Service Control Manager

Type: INFORMATION

Computer: MCON

Time: 8/27/2002 6:37:26 PM ID: 7035

User: Administrator\MCON

The WMI Performance Adapter service was successfully sent a start control.

[1416] W32Time

Type: INFORMATION

Computer: MCON

Time: 8/27/2002 12:45:11 PM ID: 35

The time service is now synchronizing the system time with the time source <<NTP>>
 (ntp.m\0x1|<<MCON>>:123-><<NTP>>:123).

.....

```

-----
psloglist application -- Application log dump
-----

```

PsLogList v2.2 - local and remote event log viewer
 Copyright (C) 2000-2001 Mark Russinovich
 Sysinternals - www.sysinternals.com

Application log on \\MCON:

[1407] McUpdate

Type: INFORMATION

Computer: MCON

Time: 8/27/2002 12:08:09 PM ID: 4550

The .DAT and Engine versions on the system (4219 and 4160) are the newest available.

[1406] McUpdate

Type: INFORMATION

Computer: MCON

Time: 8/27/2002 12:08:06 PM ID: 4570

AutoUpdate Task started.

[1405] MsiInstaller

Type: INFORMATION

Computer: MCON

Time: 8/27/2002 8:51:29 AM ID: 11728

Product: WebFldrs XP -- Configuration completed successfully.

.....

psloglist Security -- Security log dump

PsLogList v2.2 - local and remote event log viewer
Copyright (C) 2000-2001 Mark Russinovich
Sysinternals - www.sysinternals.com

Security log on \\MCON:

No records in Security event log on MCON.

Appendix C – sn.dat apptrace log

```
3054  execve("./sn.dat.orig", ["/sn.dat.orig", "eth0"], [/* 28 vars */]) = 0
3054  fcntl64(0, F_GETFD) = 0
3054  fcntl64(1, F_GETFD) = 0
3054  fcntl64(2, F_GETFD) = 0
3054  uname({sys="Linux", node="wagnerdl", ...}) = 0
3054  geteuid32() = 0
3054  getuid32() = 0
3054  getegid32() = 0
3054  getgid32() = 0
3054  brk(0) = 0x80ab488
3054  brk(0x80ab4a8) = 0x80ab4a8
3054  brk(0x80ac000) = 0x80ac000
3054  socket(PF_INET, SOCK_PACKET, 0x300 /* IPPROTO ??? */) = 3
3054  bind(3, {sin_family=AF_INET, sin_port=htons(25972),
sin_addr=inet_addr("104.48.0.0")}, 16) = 0
3054  ioctl(3, 0x8927, 0xbffff890) = 0
3054  ioctl(3, 0x8921, 0xbffff890) = 0
3054  ioctl(3, 0x8913, 0xbffff890) = 0
3054  ioctl(3, 0x8914, 0xbffff890) = 0
3054  fstat64(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 0), ...}) = 0
3054  old_mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x40000000
3054  write(1, "ADMSniff priv 1.0 in libpcap we"... , 41) = 41
3054  write(1, "credits: ADM, mel , ^pretty^ for"... , 54) = 54
3054  brk(0x80ad000) = 0x80ad000
3054  open("The_logz", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 4
3054  recvfrom(3, "\377\377\377\377\377\377\0PV\300\0\1\10\6\0\1\10\0\6\4"... , 1564, 0,
{sin_family=AF_UNIX, path="eth0"}, [18]) = 60
3054  ioctl(3, 0x8906, 0xbffff880) = 0
3054  recvfrom(3, "\0PV\300\0\1\0PV\300\30\276\10\6\0\1\10\0\6\4\0\2\0PV\300"... , 1564,
0, {sin_family=AF_UNIX, path="eth0"}, [18]) = 60
3054  ioctl(3, 0x8906, 0xbffff880) = 0
3054  recvfrom(3, "\0PV\300\30\276\0PV\300\0\1\10\0E\0\1V\253\260\0\0\200"... , 1564, 0,
{sin_family=AF_UNIX, path="eth0"}, [18]) = 356
3054  ioctl(3, 0x8906, 0xbffff880) = 0
3054  recvfrom(3, "\0PV\300\0\1\0PV\300\30\276\10\0E\20\1H\0\0\0\20\21\227"... , 1564,
0, {sin_family=AF_UNIX, path="eth0"}, [18]) = 342
3054  ioctl(3, 0x8906, 0xbffff880) = 0
3054  recvfrom(3, "\0PV\300\0\1\0PV\361\210\33\10\0E\0\0JV9@0@21\322\226"... , 1564, 0,
{sin_family=AF_UNIX, path="eth0"}, [18]) = 88
3054  ioctl(3, 0x8906, 0xbffff880) = 0
3054  recvfrom(3, "\377\377\377\377\377\377\0PV\300\0\1\10\6\0\1\10\0\6\4"... , 1564, 0,
{sin_family=AF_UNIX, path="eth0"}, [18]) = 60
3054  ioctl(3, 0x8906, 0xbffff880) = 0
3054  recvfrom(3, "\0PV\300\0\1\0PV\361\210\33\10\6\0\1\10\0\6\4\0\2\0PV\361"... , 1564,
0, {sin_family=AF_UNIX, path="eth0"}, [18]) = 42
3054  ioctl(3, 0x8906, 0xbffff880) = 0
3054  recvfrom(3, "\0PV\361\210\33\0PV\300\0\1\10\0E\0\0008\253\261\0\0\200"... , 1564, 0,
{sin_family=AF_UNIX, path="eth0"}, [18]) = 70
3054  ioctl(3, 0x8906, 0xbffff880) = 0
3054  recvfrom(3, "\0PV\300\0\1\0PV\361\210\33\10\0E\0\0JV9@0@21\322\226"... , 1564, 0,
{sin_family=AF_UNIX, path="eth0"}, [18]) = 88
3054  ioctl(3, 0x8906, 0xbffff880) = 0
3054  recvfrom(3, "\0PV\361\210\33\0PV\300\0\1\10\0E\0\0008\253\262\0\0\200"... , 1564, 0,
{sin_family=AF_UNIX, path="eth0"}, [18]) = 70
3054  ioctl(3, 0x8906, 0xbffff880) = 0
3054  recvfrom(3, "\0PV\300\0\1\0PV\361\210\33\10\0E\0\0HV9@0@21\322\230"... , 1564, 0,
{sin_family=AF_UNIX, path="eth0"}, [18]) = 86
3054  ioctl(3, 0x8906, 0xbffff880) = 0
3054  recvfrom(3, "\0PV\361\210\33\0PV\300\0\1\10\0E\0\0008\253\263\0\0\200"... , 1564, 0,
{sin_family=AF_UNIX, path="eth0"}, [18]) = 70
3054  ioctl(3, 0x8906, 0xbffff880) = 0
3054  recvfrom(3, "\0PV\300\0\1\0PV\361\210\33\10\0E\0\0HV9@0@21\322\230"... , 1564, 0,
{sin_family=AF_UNIX, path="eth0"}, [18]) = 86
3054  ioctl(3, 0x8906, 0xbffff880) = 0
3054  recvfrom(3, "\0PV\361\210\33\0PV\300\0\1\10\0E\0\0008\253\264\0\0\200"... , 1564, 0,
{sin_family=AF_UNIX, path="eth0"}, [18]) = 70
3054  ioctl(3, 0x8906, 0xbffff880) = 0
```



```

3054 recvfrom(3, "\\0PV\\300\\0\\1\\0PV\\361\\210\\33\\10\\0E\\0\\0JV:@\\0@\\21\\322\\225"... , 1564, 0,
{sin_family=AF_UNIX, path="eth0"}, [18]) = 88
3054 ioctl(3, 0x8906, 0xbffff880) = 0
3054 recvfrom(3, "\\0PV\\361\\210\\33\\0PV\\300\\0\\1\\10\\0E\\0\\0008\\253\\265\\0\\0\\200"... , 1564, 0,
{sin_family=AF_UNIX, path="eth0"}, [18]) = 70
3054 ioctl(3, 0x8906, 0xbffff880) = 0
3054 recvfrom(3, "\\0PV\\300\\0\\1\\0PV\\361\\210\\33\\10\\0E\\0\\0JV:@\\0@\\21\\322\\225"... , 1564, 0,
{sin_family=AF_UNIX, path="eth0"}, [18]) = 88
3054 ioctl(3, 0x8906, 0xbffff880) = 0
3054 recvfrom(3, "\\0PV\\361\\210\\33\\0PV\\300\\0\\1\\10\\0E\\0\\0008\\253\\266\\0\\0\\200"... , 1564, 0,
{sin_family=AF_UNIX, path="eth0"}, [18]) = 70
3054 ioctl(3, 0x8906, 0xbffff880) = 0
3054 recvfrom(3, "\\0PV\\300\\0\\1\\0PV\\361\\210\\33\\10\\6\\0\\1\\10\\0\\6\\4\\0\\1\\0PV\\361"... , 1564,
0, {sin_family=AF_UNIX, path="eth0"}, [18]) = 42
3054 ioctl(3, 0x8906, 0xbffff880) = 0
3054 recvfrom(3, "\\0PV\\361\\210\\33\\0PV\\300\\0\\1\\10\\6\\0\\1\\10\\0\\6\\4\\0\\2\\0PV\\300"... , 1564,
0, {sin_family=AF_UNIX, path="eth0"}, [18]) = 60
3054 ioctl(3, 0x8906, 0xbffff880) = 0
3054 recvfrom(3, "\\377\\377\\377\\377\\377\\377\\0PV\\300\\0\\1\\10\\0E\\0\\0N\\253\\325"... , 1564, 0,
{sin_family=AF_UNIX, path="eth0"}, [18]) = 92
3054 ioctl(3, 0x8906, 0xbffff880) = 0
3054 recvfrom(3, "\\377\\377\\377\\377\\377\\377\\0PV\\300\\0\\1\\10\\0E\\0\\0N\\253\\326"... , 1564, 0,
{sin_family=AF_UNIX, path="eth0"}, [18]) = 92
3054 ioctl(3, 0x8906, 0xbffff880) = 0
3054 recvfrom(3, "\\377\\377\\377\\377\\377\\377\\0PV\\300\\0\\1\\10\\0E\\0\\0N\\253\\330"... , 1564, 0,
{sin_family=AF_UNIX, path="eth0"}, [18]) = 92
3054 ioctl(3, 0x8906, 0xbffff880) = 0
3054 recvfrom(3, "\\377\\377\\377\\377\\377\\377\\0PV\\300\\0\\1\\10\\0E\\0\\0N\\253\\344"... , 1564, 0,
{sin_family=AF_UNIX, path="eth0"}, [18]) = 92
3054 ioctl(3, 0x8906, 0xbffff880) = 0
3054 recvfrom(3, "\\377\\377\\377\\377\\377\\377\\0PV\\300\\0\\1\\10\\0E\\0\\0N\\253\\345"... , 1564, 0,
{sin_family=AF_UNIX, path="eth0"}, [18]) = 92
3054 ioctl(3, 0x8906, 0xbffff880) = 0
3054 recvfrom(3, "\\377\\377\\377\\377\\377\\377\\0PV\\300\\0\\1\\10\\0E\\0\\0N\\253\\347"... , 1564, 0,
{sin_family=AF_UNIX, path="eth0"}, [18]) = 92
3054 ioctl(3, 0x8906, 0xbffff880) = 0
3054 recvfrom(3, "\\377\\377\\377\\377\\377\\377\\0PV\\300\\0\\1\\10\\0E\\0\\0\\364\\253"... , 1564, 0,
{sin_family=AF_UNIX, path="eth0"}, [18]) = 258
3054 ioctl(3, 0x8906, 0xbffff880) = 0
3054 recvfrom(3,

```

References

- @stake. "@stake's Pocket Security Toolkit." Version 3.0.
URL: http://www.atstake.com/research/tools/pst/pocket_security_toolkit_3.iso
(18 September 2002).
- @stake. "The @stake Sleuth Kit (TASK)." Version 1.50. 18 July 2002.
URL: <http://www.atstake.com/research/tools/task> (26 August 2002).
- @stake. "Autopsy Forensic Browser." Version 1.60. 18 July 2002.
URL: <http://www.atstake.com/research/tools/autopsy> (26 August 2002).
- @stake. "NetCat 1.10 for Unix." Version 1.10.
URL: <http://www.atstake.com/research/tools/nc110.tgz> (26 August 2002).
- Abell, Vic. "LSOF." Version 4.64.
URL: <ftp://ftp.cerias.purdue.edu/pub/tools/unix/sysutils/lsof/lsof.tar.gz>
(3 September 2002).
- ADM. "ADM." URL: <http://adm.freelsd.net> (3 September 2002).
- ADM. "The ADM CreW official ftp site." URL: <http://adm.freelsd.net/ADM>
(3 September 2002).
- ADM. "ADMsniff." Version 1.0. URL: <http://adm.freelsd.net/ADM/ADMsniff.tar.gz>
(3 September 2002).
- Dittrich, Dave. "Re: 'ls: File too large'." lists.jammed.com mail list archives.
25 April 2002. URL: <http://lists.jammed.com/forensics/2002/04/0033.html>
(5 September 2002).
- GIAC. "sn.zip." GIAC Certified Forensic Analyst (GCFA) Practical Assignment.
Version 1.0. URL: <http://www.giac.org/gcfa/sn.zip> (30 August 2002).
- Google. "Google." URL: <http://www.google.com> (3 September 2002).
- Library of Congress. "USA Patriot Act". 26 October 2001.
URL: http://thomas.loc.gov/cgi-bin/toGPO/http://frwebgate.access.gpo.gov/cgi-bin/getdoc.cgi?dbname=107_cong_public_laws&docid=f:publ056.107.pdf
(17 September 2002).
- Office of the Law Revision Counsel. "U.S. Code." URL: <http://uscode.house.gov>
(17 September 2002).

Office of the Law Revision Counsel. "18 U.S.C. Section 1030." The Computer Fraud and Abuse Act. URL: <http://uscode.house.gov/DOWNLOAD/18C47.DOC> (17 September 2002).

Office of the Law Revision Counsel. "18 U.S.C. Section 2511." The Electronic Communications Privacy Act.
URL: <http://uscode.house.gov/DOWNLOAD/18C119.DOC> (17 September 2002).

SecurityFocus. "ADMSniff 0.8." Version 0.8.
URL: <http://online.securityfocus.com/tools/215/scoreit> (3 September 2002).

Stearns, William. "Appttrace." Version 0.1. URL: <ftp://ftp.stearns.org/pub/appttrace> (3 September 2002).

TCPDump.org. "LibPCAP." Version 0.7.1.
URL: <http://www.tcpdump.org/release/libpcap-0.7.1.tar.gz> (3 September 2002).

TCPDump.org. "TCPDump." Version 3.7.1.
URL: <http://www.tcpdump.org/release/tcpdump-3.7.1.tar.gz> (3 September 2002).

VMWare. "VMWare Workstation 3.1." URL: <http://www.vmware.com> (18 September 2002).