First Responders Evidence Disk

By

Jesse Kornblum

From the proceedings of

The Digital Forensic Research Conference

DFRWS 2002 USA

Syracuse, NY (Aug 6th - 9th)

Preservation of Fragile Digital Evidence by First Responders

Special Agent Jesse Kornblum
Air Force Office of Special Investigations
`jesse.kornblum@ogn.af.mil`
8 August 2002
Digital Forensics Research Workshop

Introduction

The nature of computer based evidence makes it inherently fragile. Data can be erased or changed without a trace, impeding an investigator's job to find the truth. The efforts of first responders are critical to ensure that the evidence is gathered and preserved in a simple, secure, and forensically sound manner. This paper describes the challenges first responders face and some strategies for dealing with them. As an example, the paper also details a sample tool for first responders to incidents on Windows based computers.

The Fragility of Digital Evidence

Unlike evidence encountered during traditional investigations, digital evidence can be very fragile. Once a crime scene has been secured, the evidence of a traditional crime such as fingerprints or firearms tends to obey The Dead Body Theorem ("It's not going anywhere"). Even if the integrity of the evidence is threatened preserving the evidence can be accomplished quickly and with a minimum of expertise on the investigator's behalf. For example, if rain starts to fall on footprints in the dirt, an investigator can cover the area with a tarp.

When a computer is involved in the crime scene, however, the situation is not as clear. The very existence of evidence may not be obvious upon initial examination. There are no bullet holes to show where an intruder has gained unauthorized access nor blood stains to show where information has been destroyed. In order to uncover these details, an investigator must perform a full forensic analysis of the computer.

Performing such an analysis is a slow and difficult process that begins with first taking the computer into evidence. Upon that seizure, however, the computer is usually turned off and taken back to a laboratory. When that happens, all of the data on the computer that's not saved to the hard drive is lost forever. In modern systems, that can be a substantial amount of data and can be comprised of several parts.

Even without the power being turned off, evidence can still be destroyed. If the computer is still connected to the network, an individual could delete evidence either as an overt act or by inadvertent use. That is, an intruder could deliberately erase log files, but a legitimate user could cause data to be written to the system that overwrites evidence of a crime. In addition, the programmable nature of the computer allows an individual to instruct the computer to erase information without any human interaction. For example, an intruder could use a compromised machine to launch another attack and then automatically erase all evidence of the attack and initial intrusion upon completetion.

Beyond outright destruction, an investigator must also be careful that digital evidence is not tainted before it can be secured. Most often tainting occurs because of the good intentions of a first responder. That is, the first people to discover the crime look around to determine what happened, and in doing so, alter data on the system. The first responder must do something to determine that an incident has occurred. In doing so, however, the first responder shouldn't do anything to compromise the evidence.

The great majority of computer security incidents will never be taken to court, but the possibility still exists that they might, and thus proper precautions should be taken. Our goal is to create a procedure that will allow a first responder to find out if an incident has occurred, determine the nature of the incident, and to do so in a manner that is sound enough to be taken to court if necessary. Because we cannot train every system administrator or first responder how to conduct a completely sound investigation, it is imperative that we provide them with a tool can accomplish these goals automatically; a point and shoot tool.

Types of Fragile Evidence

We are concerned with two major types of fragile evidence:

- **Transient data** - Information that will be lost at shutdown, such as open network connections, memory resident programs, etc.
- **Fragile data** - Data that is stored on the hard disk, but can easily be altered, such as last accessed time stamps.
- **Temporarily accessible data** - Data that is stored on the disk, but that can

only be accessed at certain times.

The obvious danger facing transient data is that it will be lost. If the machine is turned off, which many nervous managers order, this data is gone forever. Sometimes programs keep data in memory for faster access times. Other times, however, transient data could include programs that are running but have been removed from the disk. That is, if a user launches a program and then, while the program is running, erases the program from the disk, the program will exist only in memory. An attacker could use this to hide evidence of an illicit program such as a sniffer. That is, the user could compile a sniffer on the victim, run it, and then delete the file from the disk. When the sniffer is stopped or the machine is turned off, there is no evidence of its existence in the active file system.

In addition, transient data could include things that simply end over time. Network connections are closed or timeout, users log out, and cached data expires. All of these things could hold potential evidence of a security incident.

Fragile data are those things stored on the hard drive but that can be easily altered, especially by a first responder trying to determine if an incident has occurred. These could include access dates on files or temporary files. Once these files have been altered by a first responder, there is no way to recover the original data. For example, let's say that an intruder has attempted to hide her tracks by removing entries from a log file. In editing the file she changes the access times of the file, leaving an indication of the time of her actions. When the first responder tries to examine the same file to see if there is any evidence of an intrusion, he changes the access time of the file again. The original access time, which could have been used as evidence, is gone forever.

Finally, we must consider data that is stored on the hard drive, but isn't accessible all the time. A good example of this kind of data are encrypted file systems. These filesystems store data on the hard drive encrypted and only after an access device (e.g. password, fingerprint, smart card) is presented. If the first responder turns the computer off, it may be difficult or impossible to obtain the access device later on, especially if the computer's authorized user is uncooperative or dead. Therefore, the computer should be examined for encrypted file systems before being turned off. If encrypted file systems are found, the contents of them should be preserved immediately.

Methods of Preservations

In order to preserve fragile data, it has to be transported to a non-volatile medium as quickly as possible without disrupting any other part of the system. The victim's own hard drives are not a safe place to store information as we don't want to overwrite what could turn out to be crucial evidence. We also have to be careful not to use too much memory while transferring information. Not only could the use of virtual memory overwrite data on the disk, but we also want to save the data in memory too.

For small amounts of data, a floppy disk is a logical place to store data. Floppies are cheap, readily available, easy to transport, easy to sanitize, and can be readily write protected after use. On the other hand, their small storage space limits their usefulness for forensics. Hard drives would be useful, but attaching a hard drive can't be done without shutting down the machine; something we're trying to avoid. Even connecting a removable device like a USB or Firewire drive is not advised as this will change the system state.

The best method of extracting data from a victim machine is therefore via the network connection. These days most computers have a network connection and for the most part, intruders gain access to the system through the network connection. (Tools to respond to standalone computers require special consideration and are beyond the scope of this paper.) Ironically, the same path used by the intruder can also be used by a first responder too.

In order to not only protect the system from further attack and to keep the existence of the investigation private, it is wise to disconnect the victim from the network upon the discovering a possible incident. The victim can instead be connected to a private hub and the data collected sent to another machine on the private hub. The second machine, or receiver, may need to be configured for the victim's environment, but this shouldn't be hard to do. That is, the first responder shouldn't have to change the IP address or network settings to use the tool.

The first data sent from the victim to the receiver should be the contents of the victim's memory. These transfers should be done in small batches to avoid overwriting the remainder of memory. This may take more time, but it's more forensically sound. If the first responder is more worried about finding malicious logic, such as a timed action to erase data, this consideration can be overlooked. Once the RAM is transferred, other data can be transferred without regard to size limits.

First Responder Tools

There are several primary goals to keep in mind during the tool development process. They are:

1. **The forensic integrity of the system must be maintained.** This means not only that data shouldn't be written to the system, but also that

executable programs should not be run on the system either.

2. **The tool should do all evidence handling without any intervention from the user**. Because we don't have the opportunity to train every first responder, we have to encode all of the logic for proper evidence handling inside the tool itself.
3. **The tool should gather all of the pertinent information that will be lost either during examination or transportation of the evidence.** We don't need to grab every bit of data from the machine at this time, only those things that we can't get later on.
4. **The tool should provide the first responder with enough information to determine if an incident has occurred.** Unless the first responder gains something by using this tool, they're not going to want to run it. Again, because we can't always train the first responder, we have to give them a reason to use our tool.

As stated before, we are concerned with maintaining the forensic integrity of the victim at all costs. To that end it is our goal to modify as little information as possible on the machine. We accept that by doing any kind of analysis of the system we are going to change some things, but we want to keep that to a minimum. One thing that can be overlooked as part of this goal is that we must also be careful not to trust the applications on the system as well. Not only will accessing an application alter data on the disk, but it's possible that it may running an application may cause serious damage.

If an application has been altered or replaced with a trojan, it could cause immediate and serious damage to the system. For example, if an attacker could replace the "who" command with one that deletes files on the system, a first responder could inadvertently destroy data. Even without a malicious individual replacing an executable, running a program from the disk could change the state of the system. For example, a program could delete temporary files left from a previous execution or change registry entries.

Secondly, the tool should not leave any of the evidence handling in the hands of the first responder. This isn't to say that the first responder can't control the tool, only that they should be physically separated from the evidence. In this manner, it can be guaranteed that the evidence was not tampered with. While a careful first responder could argue that they didn't do anything to alter the evidence, it is only by following a specific set of procedures can it be convincingly argued. Further, if those procedures call for using tools that don't alter the data (or alter it in a known manner), it can be used as proof in a court of law.

The above is not to say that the agent should not be in control of the tool; it doesn't have to be completely automated. But the tool should be designed in such a way that it's impossible to alter the system in an unknown manner. That way, should the case ever go to court, it can be argued that the first responder did not alter the system during the initial response. Thus, an area of potential "reasonable doubt" can be eliminated and more of the truth can be brought to the court room.

Next, it is the goal of our tool to gather and protect the information that can be easily lost. The tool does not have to preserve all of the evidence, only the fragile pieces. Attempting to preserve all of the evidence may prove more time consuming (and more complicated) than a first responder can afford. The job of a first response tool is only to protect those data not covered by the Dead Body Theorem.

Finally, the tool must be useful. This sounds like a trivial point, but unless the first responder has a reason to run the tool, they're not going to. Yes, the use of a first response tool can be dictated as part of an incident response plan, and this is a Good Thing(tm), but that doesn't guarantee that the tool will be used. People can overlook the proper procedure or panic in a stressful situation. By creating a tool that generates meaningful output to the first responder, we can increase our chances of use by the first responder. They are more likely to use something that's useful.

Implementing these goals and developing a useful tool is left as an exercise to the reader.

FRED - A Sample Preservation Tool

The First Responder's Evidence Disk (FRED) is a simple example of a real world incident response tool developed by the Air Force Office of Special Investigations (AFOSI). In the summer of 2000, we drafted the requirements for a tool for our first responders that would preserve fragile evidence on possibly compromised computers. Our first responders included both local system administrators and members of the Air Force Computer Emergency Response Team (AFCERT) based at Lackland AFB, TX. In a typical scenario, either the local system administrator or an AFCERT member would determine that a possible incident has occurred. After that determination, the AFCERT Incident Response Team (IRT) coordinates efforts between reviewing AFCERT data and the local system administrator to determine if in fact an incident has occurred. If they concurred that it had, the event was assigned an AFCERT incident number and the relevant information was turned over to AFOSI.

Many AFOSI agents were finding that as part of the IRT process, the system administrators were running a lot of commands on the victim system. As a result, by the time AFOSI was informed that a victim existed, the computer had been accessed by multiple system administrators, and, often enough, turned off to prevent further compromise.

AFOSI began to work with the AFCERT IRT to develop a process better suited for investigations. As part of that process, we developed the FRED to help the IRT preserve evidence while helping them make their incident determination faster. FRED was developed with the following primary goals:

1. **Don't alter anything on the victim system** - No data is written to the victim
2. **Don't trust the victim system** - Every effort is made to avoid using programs from the victim system.
3. **Fit on a floppy disk** - Almost every computer has a floppy disk, floppies can hold a small, but useful amount of data. Although more tools could fit on a CDROM, getting the data off the machine without violating the first principle becomes much more difficult.

Some of these requirements are specific to the distributed nature of the Air Force situation. A smaller organization would not need to limit itself by using a floppy disk. Many people have suggested that more tools could be added to the FRED toolkit if it was moved to a CDROM, but we have decided not to pursue this option. The distributed nature of the Air Force means that our incident response tool may need to be sent around the world at a moments notice. Although our bandwidth is increasing over time, it still requires much more effort to burn a CDROM than to copy a file to a floppy. To make things even simpler, FRED uses a self extracting archive to write itself to a floppy drive.

FRED uses commercial off the shelf (COTS) tools to do its analysis. Some of them were obtained at a cost to the Air Force, but the majority of them were freely obtained. The total cost of FRED, to include development, documentation, testing, and distribution, was less than $2,000.

The primary focus of FRED is to gather information useful for determining if an intrusion has occurred. As a side benefit, information is gathered regarding the general well being of the machine too. A full description of FRED's activities is given in the FRED documentation which should be made available with this paper. In brief, FRED records basic system information (e.g. date and time), network connections, active processes, active DLLs, open ports, and takes MD5 hashes of many critical system files. The heart of FRED is a batch file that calls the COTS tools and writes the results back to an audit file on the floppy. (The contents of the batch file are given in Appendix A).

The response to FRED has been fantastic. The AFCERT IRT enjoys the new capability to examine evidence from each potential victim, and AFOSI enjoys getting an untainted version of the data. There have been some procedural questions regarding when FRED should be run and by whom, but these are issues of policy, not technology. Overall, FRED has been a great success for AFOSI.

Appendix A - FRED Batch file

The FRED engine consists of two batch files. The first is trivial, the second is the heart of the program.

The makeline batch file:

```
@echo -------------------------------------------------- >> audit.txt
```

The FRED batch file:

```
@echo FRED v1.1 is running...

@echo FRED v1.1 - 2 April 2002 > audit.txt
@echo. >> audit.txt

@call makeline
@echo START TIME >> audit.txt
@call makeline
time /t >> audit.txt
@time /t
date /t >> audit.txt
@date /t
@echo. >> audit.txt
@echo. >> audit.txt

@call makeline
@echo PSINFO >> audit.txt
@call makeline
psinfo >> audit.txt
@echo. >> audit.txt
@echo. >> audit.txt

@call makeline
@echo NET ACCOUNTS >> audit.txt
@call makeline
net accounts >> audit.txt
@echo. >> audit.txt
@echo. >> audit.txt

@call makeline
@echo NET FILE >> audit.txt
@call makeline
net file >> audit.txt
@echo. >> audit.txt
```

```
@echo. >> audit.txt

@call makeline
@echo NET SESSION >> audit.txt
@call makeline
net session >> audit.txt
@echo. >> audit.txt
@echo. >> audit.txt

@call makeline
@echo NET SHARE >> audit.txt
@call makeline
net share >> audit.txt
@echo. >> audit.txt
@echo. >> audit.txt

@call makeline
@echo NET START >> audit.txt
@call makeline
net start >> audit.txt
@echo. >> audit.txt
@echo. >> audit.txt

@call makeline
@echo NET USE >> audit.txt
@call makeline
net use >> audit.txt
@echo. >> audit.txt
@echo. >> audit.txt

@call makeline
@echo NET USER >> audit.txt
@call makeline
net user >> audit.txt
@echo. >> audit.txt
@echo. >> audit.txt

@call makeline
@echo NET VIEW >> audit.txt
@call makeline
net view >> audit.txt
@echo. >> audit.txt
@echo. >> audit.txt

@call makeline
@echo ARP (arp -a) >> audit.txt
@call makeline
arp -a >> audit.txt
@echo. >> audit.txt
@echo. >> audit.txt

@call makeline
@echo NETSTAT (netstat -anr) >> audit.txt
@call makeline
netstat -anr >> audit.txt
@echo. >> audit.txt
@echo. >> audit.txt

@call makeline
@echo LOGGED ON >> audit.txt
```

```
@call makeline
psloggedon >> audit.txt
@echo. >> audit.txt
@echo. >> audit.txt

@call makeline
@echo LISTDLLS >> audit.txt
@call makeline
listdlls >> audit.txt
@echo. >> audit.txt
@echo. >> audit.txt

@call makeline
@echo FPORT (fport /p)>> audit.txt
@call makeline
fport /p >> audit.txt
@echo. >> audit.txt
@echo. >> audit.txt

@call makeline
@echo PSLIST (pslist -x) >> audit.txt
@call makeline
pslist -x >> audit.txt
@echo. >> audit.txt
@echo. >> audit.txt

@call makeline
@echo NBTSTAT >> audit.txt
@call makeline
nbtstat -c >> audit.txt
@echo. >> audit.txt
@echo. >> audit.txt

@call makeline
@echo HIDDEN FILES (dir /s /a:h /t:a c: d:) >> audit.txt
@call makeline
dir /s /a:h /t:a c: >> audit.txt
dir /s /a:h /t:a d: >> audit.txt
@echo. >> audit.txt
@echo. >> audit.txt

@call makeline
@echo MD5SUM >> audit.txt
@call makeline
md5sum c:/*.* >> audit.txt
md5sum c:/winnt/*.* >> audit.txt
md5sum c:/winnt/system/*.* >> audit.txt
md5sum c:/winnt/system32/*.* >> audit.txt
md5sum d:/*.* >> audit.txt
md5sum d:/winnt/*.* >> audit.txt
md5sum d:/winnt/system/*.* >> audit.txt
md5sum d:/winnt/system32/*.* >> audit.txt

@call makeline
@echo END TIME >> audit.txt
@call makeline
time /t >> audit.txt
@time /t
date /t >> audit.txt
@date /t
```

```
@echo.
@echo.
@echo.
@echo.
@echo.
@echo FRED is done.
@echo.
@echo The MD5 sum of the audit log is:
@md5sum audit.txt > audit.md5
@type audit.md5
@echo.
@echo ** WRITE THIS NUMBER DOWN AND INCLUDE IT ON THE EVIDENCE TAG **
@echo            (this value also saved to a:\audit.md5)
@echo.
@echo Remove FRED from the computer and write protect it NOW.
```