SUN BLUEPRINTS
[illegible]
APPROVED

Using Computer Forensics When Investigating System Attacks

Joel Weise and Brad Powell,
Sun Client Solutions Security Expertise Center

Sun BluePrints™ OnLine—April 2005

`http://www.sun.com/blueprints`

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95045 U.S.A.
650 960-1300

Part No. 819-2262-10
Revision 1.0, 3/22/05
Edition: April 2005

Using Computer Forensics When Investigating System Attacks

This Sun BluePrints™ Online article describes how to use computer forensics when investigating attacks on a computer system. Computer forensics is an approach that helps investigators identify the source of an attack on an organization's systems and helps with assessing and recovering from any damage resulting from such an attack.

Computer forensic investigations must be conducted in such a way that the information collected could be introduced as evidence in a court of law during the criminal prosecution of the attacker. Failure to follow guidelines for handling evidence might preclude an organization from being able to successfully prosecute the attacker(s). Although not all computer-forensic investigations lead to prosecution, organizations should always collect evidence using a methodology that can stand up in a court of law.

This article contains the following sections:

About This Article

This article explains how to conduct a computer forensic investigation of a system in response to the suspicion, or actual occurrence, of an attack on that system. It discusses computer forensic analysis at different levels and provides information that is useful to a wide audience, including CIOs, DSOs, auditors, and system administrators.

This article helps organizations prepare systems for faster recovery and recommends ways of preserving evidence so that it can possibly be used in a prosecution. This article describes a range of options for responding to a computer attack, including the ramifications of each option, and provides recommendations for determining the best course of action given the specific circumstances of the attack. It provides a list of tools that are useful for investigating attacks on Sun Solaris™ systems. Finally, this article walks readers through a step-by-step example of a computer forensic investigation.

Note – Although this article uses some legal terms and concepts when describing computer forensics, this article does *not* provide legal advice—consult appropriate legal counsel instead. The laws governing computer forensics are evolving, complex, and can vary between legal jurisdictions. Organizations should retain legal counsel to assist with interpreting applicable laws, developing company policies and procedures for responding to an attack on IT resources, and conducting a computer forensic investigation.

This article assumes that:

- You have explicit permission from your organization to conduct a computer forensic analysis on your employer's property, or on your own property (where you have explicit rights to do so). This article therefore does not address such issues as warrants, searching and seizing computers or data, privacy, or electronic surveillance. These issues should rightfully be left up to local law enforcement When in doubt, seek legal advice before proceeding with forensic activities.
- Your intention is generally to determine the cause of an attack, to rectify any problems arising from that attack, to prevent future attacks, and to pursue a criminal prosecution of the perpetrator(s), if possible.
- The records gathered during a computer forensic investigation meet the standards of authenticity described in "Authenticity" on page 5.

For additional information about the topics covered in this article, refer to the "Resources for Additional Information" on page 35.

Terminology

This section provides definitions for key terms used in this article—computer forensics, computer attacks, and evidence.

Computer Forensics

Forensic science has been defined as "... any science used for the purposes of the law... [providing] impartial scientific evidence for use in the courts of law, and in a criminal investigation and trial...." (http:// www.thinkquest.org).

According to Marcus Ranum, "Network forensics is the capture, recording, and analysis of network events in order to discover the source of security attacks or other problem incidents" (http://searchnetworking.techtarget.com).

For the purposes of this article, we expand on these definitions to define *computer forensics* as:

> "the capturing, processing, preservation, and analysis of information obtained from a system, network, application, or other computing resource, to determine the source of an attack on those resources."

These activities are undertaken in the course of a computer forensic investigation of a perceived or actual attack on computer resources.

The primary goals of the computer forensic analysis process are:

- To help participants determine what undesirable events occurred, if any.
- To gather, process, store, and preserve evidence to support the prosecution of the culprit(s), if desired.
- To use that knowledge to prevent future occurrences.
- To determine the motivation and intent of the attackers.

This article focuses primarily on the first two goals. The topic of preventing future attacks, part of a broader discussion on designing and implementing comprehensive security protections inside an organization, is outside the scope of this article.

Computer Attacks

An *attack* is defined in this article as any kind of malicious activity targeted against computer system resources, including (but not limited to) a break-in (any unauthorized access), virus infestation, data alteration or destruction, or distributed denial of service attacks.

Evidence

Evidence is defined in this article as any physical or electronic information (such as written or electronic documentation, computer log files, data, reports, physical hardware, software, disk images, and so on) that is collected during a computer forensic investigation. Evidence includes, but is not limited to, computer-generated files (such as log files or generated reports) and human-generated files (such as spreadsheets, documents, or email messages).

The purpose of gathering evidence is to help determine the source of the attack, recover from any damage resulting from the attack, and to introduce the evidence as testimony in a court of law during a prosecution of the perpetrator(s). In order to support a prosecution, the evidence must be *admissible* in court and be able to withstand challenges as to its authenticity.

Key Issues in Computer Forensic Investigations

While conducting computer forensic gathering and analysis, participants need to be mindful of the legal aspects and ramifications of their actions. This section describes the key issues that participants should understand before proceeding to undertake investigation activities.

Whether to Prosecute the Perpetrator(s)

In the event of a system attack, an organization must determine whether to prosecute the attacker(s). The decision to pursue prosecution could add considerable time and effort to a computer forensic analysis, because precautions must be taken to adequately preserve computer images and data so that they can later be admitted as evidence in a court of law.

If your organization is absolutely certain that it will not attempt to prosecute the attacker(s), then some considerations for managing the information gathered during the computer forensic investigation might not be necessary. However, we recommend that, even if the current intent is not to pursue prosecution, your organization should keep your options open and treat all gathered information as though it might eventually be used as evidence in a court of law.

Proper Handling of Evidence

An important part of any computer forensic investigation is the manner in which evidence is collected and preserved for the purpose of supporting prosecution in a court of law.

Authenticity

When handling evidence, the goal is to ensure that, if the evidence is challenged, it can be persuasively demonstrated that the evidence is *authentic* or untampered with: that it originated from the attacked system, and that it accurately represents the state of the system at a given point in time. The standard of authenticity mandates that computer records cannot have been altered, deleted, or damaged after they were created. It is essential that evidence be untainted so that it does not become inadmissible. The baseline to follow is that, absent specific evidence that tampering has occurred, the records can be considered admissible.

Proponents submitting the evidence must be able to demonstrate that the evidence is what it is claimed it to be. This, of course, does not imply that a judge or jury will believe the claims. It simply means that the evidence can be introduced. A challenge to the correctness of the records normally only affects their weight in terms of how they are considered by a judge or jury.

Reliability

The reliability of computer records depends on the reliability of the computer system that is under investigation. If a system is shown to be reliable and uncorrupted, then the records produced from that system are normally considered to be reliable as well, and thus they should be considered authentic and admissible as evidence. Taking a copy of a system and showing that it can be reconstituted to a reliable state implies that, at some point, the system was able to produce accurate records of its condition.

Chain of Custody

In the course of conducting a computer forensic investigation, tracking the *chain of custody* is essential for preparing evidence for possible prosecution and for mitigating any challenge to the evidence during that prosecution. A chain of custody is a process used to maintain and document, in detail, the chronological history of the investigation, including the collection, handling, and preservation of evidence, along with a record of anyone who has come into contact with that evidence.

A chain of custody shows that the evidence was collected from the system in question, and that it was stored and managed without alteration. This can entail labeling evidence, having a signature log kept for each time any evidence changes hands, and other similar activities.

Note – Data that cannot be recovered, backed up, or is known to be missing should also be identified and documented to ensure a complete record.

Keeping the Original System Intact

In a computer forensic investigation, it is necessary to maintain the integrity of the original system file image, its data, and its memory image. This bolsters the evidence in the face of a legal challenge. For example, an attorney might suggest that the original image was modified and thus does not accurately reflect the true condition of the system.

Making a Copy of the Compromised System

A key task in a computer forensic investigation is to safely make two precise and complete copies of the disk image and memory dump: one copy is submitted as evidence, while the other copy is used for analysis. *Under no circumstances should anyone perform the investigation on the original system or on the copy to be used for evidence*, as this would taint the evidence and jeopardize your legal case. Investigators can safely use various hardware devices to make copies of system images. These devices disable the ability to write or change existing data, which helps ensure the authenticity of the original data while allowing the investigation of data on the copy.

Cryptographically Signing Data

We strongly recommend that investigators generate a cryptographic signature and/or hash of the original system image and use it as a means of demonstrating that the image is not modified. The cryptographic signing of data must be performed in such a way that the signature itself is not introduced onto the evidence. For example, avoid tainting the evidence by generating an md5.signature and then storing the signature on the disk being examined.

Physically Securing Data

All of the data recovered from the compromised system should also be physically secured. We recommend that investigators make two copies of the data to be used during the investigation, and then seal and lock up the original data (if on disk or tape) in a secure vault, if one is available.

Guidelines for Computer Forensic Analysis

This section presents some of the most important guidelines to consider before starting a computer forensic investigation. These questions and guidelines should help organizations identify what has happened to a system and determine how to approach a forensic investigation.

Note – Laws vary in different legal jurisdictions. Therefore, users should seek legal counsel to supplement the information presented in this article. These guidelines are industry best practices that are modeled on those provided by United States Department of Justice.

Questions to Consider

Consider the following questions related to computer forensic analysis:

- Is there a link between data and the actions of a hacker or other adversary?
- What is the liability of an organization that thinks its data might have been compromised?

- When should other individuals or organizations (that might have been adversely impacted by this intrusion) be notified?
- When is it necessary to bring in law enforcement?
- Is there proprietary data that could be released as part of evidence that might adversely affect the enterprise?
- Has the system really been attacked? Ensure that your organization is not simply investigating a performance problem, a misconfigured system, or a prematurely released application.

Answering some of these questions might require support from the company's Human Resources, Public Relations, or Legal departments.

Key Guidelines

When performing a computer forensic analysis, consider the following top four guidelines:

- Guideline #1—Don't Panic
- Guideline #2—Treat Everything as Evidence
- Guideline #3—Create a Plan for Handling the Situation
- Guideline #4—Isolate the System

Guideline #1—Don't Panic

When performing computer forensic activities, it's easy to get caught up in the situation and to start searching the suspected compromised system. In one word—*don't*. There is little to be gained by doing this and, in fact, there is much to lose. Over the years, Sun has conducted many computer forensic investigations that were initially thought to be intrusions but that instead turned out to be system failures or miscommunications between two or more system administrators.

It is best not to assume anything at this point. This is the time for rational and planned actions, not for any quick responses that might damage evidence. Remember, every time a file or directory is opened, the system records this change. A *timeline* is the chain of events that is recorded by the computer system, starting with older events first. Poking around at this point damages the timeline, which is critical in determining when the system was (potentially) broken into. Avoid this, and instead consider isolating the system and getting it into a state in which the investigation can proceed without damaging evidence.

Guideline #2—Treat Everything as Evidence

In the early stages, it is yet to be determine whether or not a crime has, in fact, been committed. Regardless, investigators need to act as if one has been committed. Until it has been determined with certainty whether a security incident has occurred, it must be assumed that one has, and act accordingly.

Be sure to document everything done from this point on. Using a new pad of paper, start a log book and document everything by hand: dates, times, locations, conversations, people in contact with the compromised system, all activities involved with the investigation, and so on. Think of this as your new runbook and document everything done to the system. There is a reason investigators fence off a crime area—they must preserve the scene and not allow changes to occur that might taint or destroy evidence.

Guideline #3—Create a Plan for Handling the Situation

Your organization should already such have a plan in place prior to any computer security incidents, but (sadly) this is rarely the case. If you are reading this article before needing to deal with an actual security incident, take this opportunity to inform your management of the need to develop policies and guidelines for handling these types of incidents. There are so many different scenarios for which to prepare—too many to list in this article, so we will make some general comments and move on.

Most computer security incidents go unreported, and even fewer get anywhere near a courtroom. There are plenty of reasons for this. In the corporate environment, coming forward and admitting a computer security incident to law enforcement can be a daunting and possibly career limiting move to many IT security organizations. If it was your responsibility to prevent these situations, coming forward can mean accepting that your IT security protections are not perfect. Although no "perfect" security solution exists, IT departments like to believe that they have good solutions in place and often won't admit that they are vulnerable in this area. Further, when breaches to security are made public, the organization's stock price, customer trust, brand name, and reputation can all suffer.

By getting management involved early in the process of making reasonable decisions on how to handle computer security incidents, you can significantly reduce the negative impact of these situations on your organization. Letting intruders continue their crimes will never improve the landscape of electronic security.

While your organization might never prosecute these types of incidents, conducting a full investigation enables them to know exactly what happened and helps you take measures to ensure that other systems will not suffer the same fate in the future. Thorough investigations can also benefit your organization by prompting it to change the way it configures and operates systems in order to prevent future intrusions, and to prevent culprits from using these systems as jumping off points for attacking others.

In addition, consider reporting the problems experienced in an attack to the applicable software vendors. This type of information can help vendors improve their products, thereby decreasing the chance that others will suffer the same fate.

Guideline #4—Isolate the System

The compromised system needs to be isolated from the network to prevent further access by the suspected intruder, and also to prevent the system from attacking other systems. Automated processes, such as worms or viruses, must not be allowed to infect other systems attached to the network.

Isolating the system does not mean turning it off or shutting it down. In fact, quite the contrary. Shutting the system down will destroy the memory state and kill running processes that can provide valuable information about what the suspected intruder was up to.

Deciding How to Respond to An Attack

In the event of a suspected attack on a computer system, the first step in preparing for the investigation is deciding how to respond to the attack. Your organization has a range of responses to consider, including:

- Doing nothing.
- Performing an analysis as fast as possible so that the compromised system can be repaired and put back into production, allowing business processes to resume.
- Performing as detailed an analysis as possible, properly collecting and preserving all evidence in anticipation of possible prosecution.

Your organization must decide its response on a case-by-case basis. However, this article makes the following recommends:

- Whenever possible, perform as detailed and comprehensive an investigation as possible.
- Your organization should assume that the information gathered during the investigation will, sometime in the future, need to be admitted as evidence in a court of law for the criminal prosecution of the person(s) participating in the attack

Once your organization has decided on how to approach an investigation, investigators can take any of the following specific actions to conduct the investigation:

- Do Nothing
- Reinstall and Move On
- Investigate for Yourself
- Call for Help

The following sections describe some of the positive and negative implications of these various actions.

Do Nothing

We do not consider this to be a viable option and strongly recommend any other approach instead. Nonetheless, this approach is taken more often than it should be, with victims of attacks hoping that attackers will get bored and go away. Many home users use this tactic, thinking they have no real value on their systems or wireless access points, thus they do not consider it much of an issue.

The negative consequence to this approach is that your site might be used as a staging point to attack others. You might be the one who receives the knock on the door by the local police department with a search warrant because your system was used to stage attacks upon other systems. There might be legal ramifications that can leave your organization liable if one of its systems was, in fact, used for illegal purposes.

Reinstall and Move On

This approach is probably the fastest way to recover from an incident with minimal interruption to system operations. Unfortunately, it has become the de facto way in which most computer incidents are handled. In this case, an organization just chalks up the intrusion to the cost of doing business, reinstalls the OS, and gets the system back into production as soon as possible. Often, little or no negative publicity about the incident becomes public.

The negative consequence of this approach is that it emboldens attackers. They might attack again, and one cannot be certain to have closed all the holes. Intruders often leave backdoors that are removed by reinstalling the OS. However, most often, during the initial break in an attacker will gather and retain enough information about your organization to be able to attack more efficiently again. For example, attackers might have already sniffed or cracked passwords that will allow them back into your systems. Without knowing for certain which initial security failure(s) allowed them access the first time, one cannot be sure to have closed all access points.

Investigate for Yourself

The positive aspect of this approach is that no outsider needs to be contacted. Depending on the level of expertise that is available from in-house resources, your organization might be able to complete the investigation in a timely and efficient manner. The downside of this approach is that, even if the investigation is successful, others do not know about the attack scenario and do not benefit from the results of the investigation.

Call for Help

Calling for outside help is the most practical of the four options, and it is the approach that we recommend for most scenarios. Many sites are not able to have an onsite specialist who knows computer forensic methodology. Computer forensics is a discipline that can take years to really understand intimately. It can be a daunting task to know all of the different techniques required to perform an investigation on all of the different types of operating systems.

Bringing in a trustworthy confidential investigator, when needed, might be less expensive than trying to keep a resident expert on the payroll (assuming that your system is not broken into on a regular basis). A hired consultant who knows computer forensic techniques will often be able to detect, isolate, and help your organization recover from attacks in a timely manner.

Certification of computer forensic investigators is a trend that is underway but still developing. If possible, hire a certified computer security forensic investigator. As of the writing of this article, few certification programs exist for computer forensic investigators, so finding a certified investigator might prove difficult.

Seeking bids for this type of consulting service just after an attack has occurred is not the right time to do so. When an attack has occurred, time is very important—both for the investigation, as well as for the recovery from the incident. This article urges organizations to find competent computer forensic investigators who can fulfill their needs in advance—well before an attack occurs.

Preparing for Forensic Analysis

Assuming that a computer forensic analysis investigation will be performed, certain policies, procedures, and processes must be in place in order to obtain useful information. Before initiating a computer forensic analysis, consider the following questions:

- Is there an incident response process in place?
- Is there a designated incident response team?
- Will the investigation be performed internally, or outsourced to forensic experts?
- Are there adequate audit logs and can they be trusted?
- Has the system really been attacked? Or do the symptoms point to other issues, such as application performance problems or even something as simple as an system administrator entering an incorrect command?
- Are system and data files backed up so that they can be used as baseline references?
- Is there sufficient space available to perform a forensic analysis? In general, sufficient space is a minimum of two to three times the disk space of the suspect system.
- Does your organization have the necessary tools to perform the forensic analysis?
- Does your organization have proficiency in the proper usage of the forensics tools? Even the most sophisticated tools can corrupt evidence if not handled correctly.

For this article, we make the following assumptions about the environment and use the following tools to retain system state while performing the analysis.

Assumptions

The process for performing a forensic analysis, described in "Conducting a Forensic Investigation" on page 15, makes the following assumptions:

- You have (or the investigator has) explicit permission from your organization to conduct a computer forensic analysis on your employer's property, or on your own property (where you have explicit rights to do so).
- No wireless connectivity is involved.
- Only Sun Solaris™ systems are involved in the breach.
- Your organization has access to someone with the technical skills required to perform a forensic analysis.

- There are adequate system resources and storage to perform the various tasks needed, such as backups.
- You have (or the investigator has) access to the tools described in the following section.

Tools

Before getting started, assemble the following tools that enable you to retain system state and running processes while collecting the evidence for forensic analysis:

- A UNIX®/Linux system with at least twice to three times as much free disk space as the disk(s) of the compromised system. This capacity is needed to store root, swap, proc, and any other locations where the intruder might have hidden files. This can be a laptop on which to record the system disk image for archival purposes, and a separate system at the office that has lots of free disk space for the actual forensic analysis.
- A forensic tools CD or, at a minimum, the installation CD for the operating system. It is useful to have a forensic CD that can be mounted on the suspect system and booted for collecting the disk image. Specific forensic tools, such as The Coroner's Toolkit (TCT) and `mac-robber`, are discussed later in this article.
- A crossover Ethernet cable or a 10/100 megabyte Ethernet mini-hub, as well as other network gear, if required (ATM network gear, or fiber assuming the system does not have Ethernet).

Note – In place of the three tools above, you could use a hardware disk imaging device, as mentioned previously.

- Make sure that you provide a signed copy of all tools used during the investigation, and that you store a copy together with the evidence in order to prove the correctness of data gathering or analysis. The best way is to provide a copy of the forensics CD and its MD5 checksum together with the sums of the data copies.
- At a minimum, the software tools required (for disk imaging in the event that a hardware disk imaging device is not available) would include the following:
  - `dd(1)` command
  - `netcat` `nc(1)` command
  - `sh(1)` (`/sbin/sh` is often statically compiled on the Solaris Operating System, or Solaris OS)
  - `md5(1)`
  - `cp(1)`

    - `memdump(1)`

    All software tools should be statically compiled, if possible, and should be the appropriate binary for the operating system being copied. If static compiling is not possible, then the `LD_LIBRARY_PATH` should be set to point to the path of the forensic tools CD. In addition, set `LD_NOAUXFLTR=yes`, which will instruct the dynamic loader not to load alternative (optional) libraries.

The following tools, while not required to dump the image, are helpful to have for the purpose of querying running processes on the suspect system:

- `netstat(1)`
- `tcpdump(1)`
- `ps(1)`

Conducting a Forensic Investigation

To conduct a forensic investigation, complete the following primary steps:

- Step 1: Isolate the System and Data

  Collect information by interviewing all system administrators and anyone else who might have come into contact with the system.
- Step 2: Collect Non-Technical Information
- Step 3: Preserve Evidence and Create Copies for Analysis
  - Memory and /proc.
  - Create a duplicate of the system.
  - Validate copies.
- Step 4: Prepare for an Analysis
- Step 5: Perform the Analysis
- Step 6: Perform Recovery

The rest of this section describes these steps in detail.

Step 1: Isolate the System and Data

The purpose of the isolation step is twofold:

- To prevent the corruption of other systems, reducing the risk of a cascading failure throughout an organization's IT infrastructure.
- To maintain the state of the system so that an exact image of the system is preserved for possible prosecution.

Isolation involves the following tasks:

- Sever network connectivity from the system(s) in question by physically disconnecting the network cables.
- Ensure that no one other than the forensic analyst touches anything. This includes everyone—from the CEO and CTO to maintenance personnel and interns.

Step 2: Collect Non-Technical Information

The purpose of this step is to ensure that the required minimum information is recorded in paper note form, such as developing a timeline of events. The premise here is to start a log book using a fresh pad of paper to gather information from the system administration staff and any other persons involved. It is important to collect and record (in writing) as much of this information as possible, as soon as possible, to ensure that the people involved don't forget facts and details, and to document the dates and times when the suspected break-in was detected. The investigator must bear in mind that any entries in the notebook could end up in court, so it is important to keep precise handwritten notes, untainted by assumptions or premature conclusions.

Recording this information is critical in a courtroom. It also helps narrow the search parameters when examining the electronic data. Identifying the approximate date and time at which the suspected break-in occurred provides a reference point that helps focus the search to the relevant time, preventing the need to review logs that are not chronologically pertinent (such as those from years past). This makes analysis efforts much more manageable and improves the likelihood of discovering when the initial system penetration occurred.

At a minimum, document the following information:

- Date and time when the suspected break-in occurred.
- The names, job functions, and contact information of all parties involved, including how they were involved.
- Dates, times, and names of when management got involved.
- Dates and times when the investigation started and ended.

- Dates, times, names, and contact information of the people performing the investigation.
- Every step, process, and procedure performed on the data that was gathered and processed in the investigation.

Step 3: Preserve Evidence and Create Copies for Analysis

The purpose of this process is twofold:

- To ensure that the original version of the attacked/corrupted system is copied and stored as evidence for possible future prosecution.
- To have a separate system copy that can be searched, investigated, and analyzed—to figure out what happened and when.

Note – Before performing this procedure, make sure that the system has been *disconnected* from all other networks.

Step 3a: Copy Volatile Data

1. **Mount (`vold`) the forensic CD or use NFS mount.**

   Isolate the system using a mini hub or Ethernet crossover cable. Do this to collect data on the running processes, perform `memdump(1)`, and copy the `/proc` file system of the running processes, as well as the kernel state table, before shutting down the system (which would result in a loss of this potentially valuable data).

2. **Run a statically compiled shell from the CD.**

   This will help keep the suspect system libraries from being used or touched (`/cdrom/sbin/sh`).

Note – An advanced attacker might have modified shared libraries instead of modifying (trojaning) a binary command. Modifying shared binaries enables the attacker to alter the correct behavior of all dynamically linked binaries that use such a library. This attack is much more powerful and makes it more difficult to detect subverting the integrity of a system.

---

3. **Set the `LD_LIBRARY_PATH` to point to the libraries on the CD for anything that is not statically compiled. In addition, set `LD_NOAUXFLTR=yes`, which will instruct the dynamic loader not to load alternative (optional) libraries.**

4. **Start `netcat` (`nc`) in listening mode on the safe system—for example, on a laptop or other device that has not been compromised—so that it will collect the data dumped from the suspect system.**

   ```
   safe-system# nc -l -p 9000 > /largedisk/memory-dump -w 999
   ```

5. **Copy the system memory state using `memdump(1)`.**

   ```
   suspect-system# memdump | nc safe-system 9000
   ```

6. **Copy the entire `/proc` file system.**

   ```
   safe-system# nc -l -p 9001 | cpio -i -d >> /largedisk/proc -w 999
   ```

   ```
   suspect-system# /cdrom/bin/find /proc/*/object/ -type f | /cdrom/bin/
   cpio -o 2> /dev/null | /cdrom/bin/nc -w 999  | nc safe-system 9001
   ```

7. **Snoop anything coming from the system (assume no encrypted traffic).**

   Run `snoop` (or `tcpdump`) on the safe-system (not on the suspect system) to collect any outbound activity originating from the suspect system. Let this run for a period of time to determine whether the suspect system is making any outbound calls. This can give pointers to the origination of the attackers.

8. **After allowing `snoop(1)` or `tcpdump(1)` to run for a period of time—gathering any data possible, finishing the memory dump (`memdump`), and saving a copy of the /proc file system—put the suspect system into stop or bootprompt mode.**

   Execute a system stop on the suspect system using the Stop-A or L1-A key sequence.

   ```
   Stop -A
   ```

Note – Additional and advanced data gathering methods, which involve running kernel dissection without tainting evidence, can be run prior to the volatile data collection. Such techniques are outside the scope of this article.

---

9. **Boot from the forensic CD or from the system install disk (assuming that `dd` and `netcat` [`nc`] are available on system install disk).**

   Doing this allows investigators to start working from a known good mini-root kernel and run off of the CD-ROM for the remaining steps.

   ```
   boot cdrom -S
   ```

10. **Optionally, if the CD image supports booting into a windowing system, or if it allows setting up Ethernet hardware and IP addresses to the same Class C network as the safe system, then that CD image can be used to help save some steps.**

11. **At this point, you should be running from the CD on the suspect system. You should have plumbed the Ethernet hardware and set the suspect system's IP address to the same network on which as the safe system is running (`192.168.1.0` in the example below).**

    If not, run `ifconfig` on the suspect system (running from the CD) to enable the Ethernet hardware and to turn on the network.

    ```
    ifconfig hme0 plumb
    ```

    ```
    ifconfig hme0 192.168.1.10 netmask 255.255.255.0 up
    ```

    Now the entire disk can be copied from the suspect system.

12. **Netcat listening on the safe system.**

    ```
    safe-system# nc -l -p 9010 > /largedisk/disk-image -w 999
    ```

    ```
    suspect-system# /cdrom/sbin/dd  bs=1024 if=/dev/rdsk/c0t0d0s0 | /
    cdrom/sbin/nc 192.168.1.11 9010
    ```

    Repeat the preceding step as needed for all other disks/partitions.

13. **Make a second copy of the disk. The following command will suffice:**

    ```
    /cdrom/forensic/bin/cp image1 image2
    ```

Note – Perform all forensic work on the second copy.

Step 3b: Validate Copies

The purpose of the validation process is to ensure that a confirmed, legitimate copy of the system has been made. Using an MD5 checksum, verify that copies of the system exactly match the original state of the suspect system. It is especially important that you create a baseline in the event that the copy of the system will be used as forensic evidence in a prosecution.

To validate the new copy, use CD statically compiled binaries:

1. **Create an MD5 checksum of copy 1.**

    ```
    safe-system# /cdrom/bin/md5-static suspect-drive-root-partition
    >suspect-drive-root.md5
    ```

2. **Create an MD5 checksum on the second copy, and then compare it to the first copy to ensure they have the same MD5 signature.**

    This ensures that the two copies are exact copies of the suspect system.

3. **Securely store the original copy of the system.**

Once the copies have been validated, the original copy of the system must be securely stored to ensure that it is preserved for future use in prosecution, if desired.

Note – It is best to preserve the actual disk from the suspect system. Label it, bag it in an anti-static bag, and lock it up. If this is not possible, then—at a minimum—set aside a copy of the disk images that were created so that they are never touched. Any analysis is performed on the second copy so that the first copy is untouched and secured, thus preserving the chain of custody of the image.

---

4. **Perform an MD5 checksum of the objects (running binaries from the suspect system).**

   The following MD5 checksum of the objects (running binaries from the suspect system) are collected, as the MD5 signature can subsequently help determine whether the process running was a Solaris binary as shipped by Sun. This helps eliminate processes that were supposed to be running by comparing them to the Solaris Fingerprint Database.

Note – On Linux, md5 has a file size limit of 2 gigabytes, so this might only work on the Solaris OS.

```
/cdrom/md5 /largedisk/proc/object/* >/tmp/proc-md5-sigs
```

Step 4: Prepare for an Analysis

Note – This section is only a short primer and does not describe the full computer forensic process. Entire books have been and are being written about all of the possible steps and procedures involved in this process.

At this point, the information collected in the preceding steps is ready for inspection. The following steps can also be performed by a third party or at another physical location.

1. **On a Solaris system, use `lofiadm` to create an association.**

   ```
   safe-system# lofiadm -a /someplace-with-lots-of-space/suspect-
   drive-root-partition-copy2
   /dev/lofi/1
   ```

   In the preceding example, note that `lofiadm` responded with `/dev/lofi/1`. This is the newly created association between the `suspect-drive-root-partition-copy2` file and a loopback mount point.

2. **Mount (READ ONLY) the `dd` image file (called `suspect-drive-root-partition-copy2` in this section) in read-only mode using the loopback file association that `lofiadm` created.**

   ```
   # mount -o ro /dev/lofi/1 /mnt
   ```

   Important – Note the `-o ro` (or "option read-only") mount.

   ---

3. **Repeat the preceding steps for other disk slices as needed.**

   At this point, after the `suspect-drive-root-partition` file and the `suspect-drive-root-partition-copy2` file exist as local files, inspect the image. The `file(1M)` command indicates that the image is "data," so mount the "data" file as a UNIX file system to examine it.

   Remember, when finished with your investigation, use the `-d` option of `lofiafm` to remove the association. The following example is just a reminder. Do not type the following commands at this time.

   ```
   # umount /mnt
   ```

   ```
   # lofiadm -d /someplace-with-lots-of-space/suspect-drive-root-
   partition-copy2
   ```

4. **After verifying that the disk image can be mounted, shut down the suspected system, install a new disk, and allow system administration to begin recovering and reintroducing the new system into production.**

   Do this only if the system administration staff needs to start the rebuild process. The longer that the suspect system can be left in its isolated state, the better, as its system memory might provide information that can be extremely useful. Having a root disk mirror is a best practice for organizations.

Step 5: Perform the Analysis

The purpose of the analysis step is to determine whether a compromise has occurred, what exactly was damaged, and to obtain any evidence indicating who the culprit(s) might be, as well as the methods they used to attack the system.

Tools are needed to complete the analysis phase. Fortunately, these tools are free and readily available. The tools used in this example are for the Solaris OS and include The Coroner's Toolkit (TCT), mac-robber, Sleuthkit, and Autopsy. For more information about these tools, see "Tools Resources" on page 36.

This section walks through an example of examining a file system. During an actual forensic analysis, you would likely examine the memory dump as well.

Note – During this task, work from the second copy with read-only capability so that timestamps are undisturbed and evidence is not corrupted.

1. **Using the MD5 checksum and the modified, access, and change (MAC) time, create an MD5 checksum for every file created in the last 15 days.**

   The MD5 checksums reveal pretty quickly whether the intruder left a smoking gun to examine, because it allows checking the MD5 signatures against the Solaris Fingerprint Database.

   ```
   safe-system# find /mnt -mtime -15 -exec md5 {} \ ; >mtime- minus15-
   md5
   ```

2. **Grab the MAC timestamps and make MD5 signatures of all the binaries.**

   ```
   safe-system# mac-robber /mnt1 >/usr/tmp/body.mac
   ```

   ```
   safesystem# mactime -b /usr/tmp/body.mac 03/25/2003 >timeline.03-25-
   2003
   ```

   The `mactime` tool gives a timeline of what has changed on the system, and the sequence of those changes.

   One can use `mac-robber`, a forensics and incident response tool, to collect the MAC times from allocated files. It recursively reads the MAC times of files and directories and prints them in time-machine format to STDOUT. This is the same format that the TCT `mactime` tool reads. The `mac-robber` tool is based on the TCT `grave-robber` tool when using the `-m` flag, except that it does not require Perl.

   The loopback mount of the suspect disk has already been mounted using the `lofiadm` command, so you can assume that `/mnt` is the mount point in the examples.

3. **With the TCT `mactime` tool, analyze the `body.mac` file created previously with `mac-robber`.**

   ```
   safe-system# mactime -b /usr/tmp/body.mac 10/06/2002 >timeline.10-06-
   2002
   ```

   ```
   safe-system# vi timeline.10-06-2002
   ```

   Notice that the `timeline.$DATE` file shows a chronological ordering of the files. The quick and dirty way to identify potential attacks is to look for anything that was modified starting with the date/time when the break in was detected by the system administration staff, and try to work backwards from there to determine when the initial system penetration occurred. This information is useful when determining which back up tapes can be safely used to restore the system.

This technique does not always work—intrusions can go undetected for months before a user or administrator notices anything. By starting with 15 days from the event, you are hoping to get lucky. If you don't find the break in during this period of time, continue going back in additional increments of 15 days until you find answers or run out of time and/or resources.

The following example is an excerpt of a `mac-robber` timeline from an actual break-in, after which the intruder left behind the following files and directories. One can deduce the order by which things were installed based on the modification (m); access (a); or creation (c); date of these files.

```
Date                      size  mac perms      owner: group       filename
Sep 17 02 11:31:19       11848 .ac -r-xr-xr-x root:  staff   /usr/bin/w
                          4692 .ac -rwxr-xr-x root:  staff   /usr/lib/vold/nsdap/cleaner
                          8616 .ac -r-xr-xr-x root:  staff   /usr/bin/sun3x
                           534 .ac -rwxr-xr-x root:  staff   /usr/lib/vold/nsdap/defines
                           512 .ac drwxr-xr-x root:  staff   /dev/prom
                         17440 .ac -r-xr-xr-x root:  staff   /usr/bin/mc68000
                          8000 .ac -r-xr-xr-x root:  staff   /usr/bin/mc68010
                         21424 .ac -rwxr-xr-x root:  staff   /usr/lib/vold/nsdap/sn2
                         26372 .ac -r-xr-xr-x root:  staff   /usr/bin/mc68020
                         18584 .ac -r-xr-xr-x root:  staff   /usr/bin/mc68030
                        136288 .ac -rwxr-xr-x root:  staff   /usr/bin/wget
                          8332 .ac -rwxr-xr-x root:  staff   /usr/lib/vold/nsdap/pg
                         52272 .ac -r-xr-xr-x root:  staff   /usr/bin/mc68040
                           512 .ac drwxr-xr-x root:  staff   /dev
                       1819109 .ac -rw-r--r-- root:  staff   /usr/lib/vold/nsdap/sn.l
                          8024 .ac -rwxr-xr-x root:  staff   /usr/lib/vold/nsdap/utime
                       1820577 .ac -rw-r--r-- root:  staff   /dev/prom/sn.l
                           804 .ac -rwxr-xr-x root:  staff   /usr/lib/vold/nsdap/basepatch
                         96252 .ac -r-xr-xr-x root:  staff   /usr/bin/sun2
                         19452 .ac -r-xr-xr-x root:  staff   /usr/bin/sun3
                        558868 .ac -rw-r--r-- root:  staff   /usr/bin/sshd
                          4388 .ac -rwxr-xr-x root:  staff   /usr/lib/vold/nsdap/patcher
                          4868 .ac -rwxr-xr-x root:  staff   /usr/lib/vold/nsdap/crypt
                           335 .ac -rwxr-xr-x root:  staff   /usr/lib/vold/nsdap/sniffload
                          4817 .ac -rwxr-xr-x root:  staff   /usr/lib/vold/nsdap/findkit
                           512 .ac drwxr-xr-x root:  staff   /usr/bin
                           174 .ac -rwxr-xr-x root:  staff   /usr/lib/vold/nsdap/README
Sep 17 02 11:31:20         512 .ac drwxr-xr-x root:  staff   /usr/lib/vold
                           512 .ac drwxr-xr-x root:  staff   /usr/lib
                           512 .ac drwxr-xr-x root:  staff   /usr
```

The preceding example is only a small portion of the time slice. During a typical computer forensic investigation, expect to sift through much more data than is shown in this short example.

The obvious fact that these files were accessed (a) and created (c) is very revealing and gives us clues about where to look more closely. The first questions to ask might be: "Why would mc6800, a Motorola (sun3) system binary, be accessed on a Solaris 2.6 SPARC architecture system? And why would it even still be there?" Most likely, the attacker moved aside old or unpatched binaries to hide some tools. Reading `/usr/lib/vold/nsdap/patcher`, a script left behind by the intruder, verifies that this is what happened.

Note that, in this example, the sequence occurs from the oldest directory to the newest, according to the following timeline:

- `/usr/lib/vold/nsdap` is created first
- `/usr/lib/vold/nsdap/README` is created next
- `/usr/bin` is accessed
- `/findkit` is created and run, and
- finally, `/sniffload`, `/crypt`, `/patcher`, and `/sshd` are installed

Next,

- `/usr/bin/sun3` and `/usr/bin/sun2` are modified (trojaned)
- `/usr/lib/vold/nsdap/basepatch` is run
- `/dev/prom/sn.l` (sniffer log) is created

And so on. Then, `wget` is used to get the latest Sun patch bundle after the original unpatched versions are placed in Trojan hiding places, and then the cleaner script is run.

One might ask why the attacker did this. By performing these actions, the attacker caused the system to now pass a `patchdiag` or patch checker audits, appearing to be up-to-date on patches. This can also ensure that other hackers cannot steal what the attacker has already taken. (Sun doesn't currently have MD5 signatures in patches or in `patchadd`.) The system in this state will have down-reved binaries that are exploitable or, in some cases, just plain Trojans that hide the attacker's presence but appear to be up-to-date in patching. This also prevents a system administrator from reinstalling a system patch that would destroy the attackers backdoors, because the system (`patchadd`) will not install a patch that it believes has already been applied.

In this situation, the PatchDiag auditing tool would have been totally fooled. The `sidekick.sh` script that checks the MD5s of `suid` might have caught this if the auditor had actually run the MD5 outputs against the Solaris Fingerprint Database, as this clever trick is no match for the Solaris Fingerprint Database. Sun has taken an MD5 cryptographic checksum of every Solaris binary that Sun has ever shipped and put these MD5 checksums into an online database that can be queried by anyone wishing to discover from which OS or patch a Solaris binary came from. One can use this database to validate binaries. For more information about the Solaris Fingerprint Database, go to the following URL:

```
http://sunsolve.sun.com/pub-cgi/show.pl?target=content/content7
```

Note – This example demonstrates why auditors can no longer rely on `patchdiag` or `patchadd` to ensure that systems are up to date.

---

4. **Returning to the analysis tasks started previously, because it is already known how many files were modified in the past fifteen days, one could do an MD5 checksum of everything that changed in that period.**

   Use this data to determine whether the binaries from the example image were shipped by Sun, and from which OS revision they came from. In addition, it helps determine whether they were not Sun-shipped binaries but are, instead, backdoors left by the intruder to hide their presence.

   ```
   safe-system# find /mnt -mtime -15 -exec md5 {} \; >mtime-minus15-md5
   ```

5. **Analyze the MAC times in the `timeline.xDATE` file.**

   At this point, simply use an editor (`vi`) on the example file (`timeline.01-25-2003`) that was created. Look for files that have been introduced into the system recently. This will show a timeline of events. Reviewing the MAC times provides a picture of when the system recorded the changes.

   The `sfpC.pl` perl script is used to easily query the Solaris Fingerprint Database. Use it against the collected MD5 signatures to determine whether the system binaries are valid Sun binaries. The following example checks for the past 15 days.

   ```
   safe-system# sfpC.pl mtime- minus15- md5
   ```

   In a previous step, the MD5.signatures of all of the `/proc/objects/*` were collected. Run these against the Solaris Fingerprint Database using `sfpC.pl` to determine the binary to which the `a.out` in `/proc` refers. Conversely, one can determine which binaries were not valid, Sun-shipped binaries.

   False positives can result if the system was running any third party software, as these MD5 signatures will not be reflected in the Solaris Fingerprint Database. Therefore, if a file shows up as having no match in the database, do not jump to any conclusions. These results indicate which binaries might need further investigation.

   The following command shows whether the binaries were shipped by Sun and from which OS release.

   ```
   safe-system# sfpC.pl /tmp/proc-md5-sigs
   ```

   Items from `/proc/objects/*` should still match if they were Sun shipped binaries, so the `a.out` that matches the Sun shipped `sendmail` (as an example) might be perfectly fine if it was running. What one looks for are binaries that have no exact match, or for binaries that match with an older version of Solaris that at one time had a security vulnerability. Therefore, be sure to look for older (Solaris 2.6) binaries on newer systems (Solaris 8). Any binaries that do not match might be Trojans or might not be Sun binaries.

```
safe-system# sfpC.pl mtime-minus15-md5
```

The following example is the output of the Solaris Fingerprint Database returned by `sfpC.pl`:

```
    b1a9e23c8cb35e34d82e043164c7954e - (/mnt/usr/bin/w) - 1 match(es)

    canonical-path: /usr/bin/uptime
    package: SUN
--------------------------------------------------------------------------------
    b08a8c2486d048bcd63e79ea1092722a - (/mnt/usr/bin/m68k) - 1 match(es)

    canonical-path: /usr/bin/su
    package: SUNWcsu
    version: 11.6.0,REV=1997.07.15.21.46
    architecture: sparc
    source: Solaris 2.6/SPARC

/mnt/usr/bin/m68k is really the original /usr/bin/su

--------------------------------------------------------------------------------
    e643207218360aa62d57ee078ddce916 - (/mnt/usr/bin/sshd) - 0 match(es)

   Not found in this database.

Not a Sun shipped binary; and added to the system during the intrusion

--------------------------------------------------------------------------------
    e0686aeb9f46ff8a2491a09bf2879055 - (/mnt/usr/bin/sun2) - 1 match(es)

    canonical-path: /usr/bin/passwd
    package: SUNWcsu
    version: 11.6.0,REV=1997.07.15.21.46
    architecture: sparc
    source: Solaris 2.6/SPARC
    patch: 106271-08
```

```
/mnt/usr/bin/sun2 =  original /usr/bin/passwd

--------------------------------------------------------------------------------
    0236041c43c1c7c96cb724f0e41ad5a0 - (/mnt/usr/bin/sun3) - 1 match(es)

    canonical-path: /usr/sbin/ping
--------------------------------------------------------------------------------
    4cd4af91f1c1ea0453f53a8c1b45c546 - (/mnt/usr/bin/wget) - 0 match(es)

   Not found in this database.
Not a Sun shipped binary; and added to the system during the intrusion




--------------------------------------------------------------------------------
   30e2cbc4f807920d7ac008db86b64baf - (/mnt/usr/bin/sun3x) - 1 match(es)

    canonical-path: /usr/bin/strings
    package: SUNWtoo
    version: 11.6.0,REV=1997.07.15.21.46
    architecture: sparc
    source: Solaris 2.6/SPARC

/usr/bin/sun3x = original Solaris /usr/bin/strings
--------------------------------------------------------------------------------
    1eabd3dbc0746c8a4b5467f99a4f8823 - (/mnt/usr/bin/mc68000) - 1 match(es)

    canonical-path: /usr/bin/ls
    package: SUNWcsu
    version: 11.6.0,REV=1997.07.15.21.46
    architecture: sparc
    source: Solaris 2.6/SPARC

/mnt/usr/bin/mc680000 =original Solaris /usr/bin/ls
--------------------------------------------------------------------------------
    e1693a0caf8551857bc1ae6e2f9a0df3 - (/mnt/usr/bin/mc68010) - 1 match(es)
    canonical-path: /usr/bin/du
    package: SUNWcsu
    version: 11.6.0,REV=1997.07.15.21.46
    architecture: sparc
    source: Solaris 2.6/SPARC

/mnt/usr/bin/mc68010 = original Solaris /usr/bin/du
--------------------------------------------------------------------------------
   162134f40dbc47eb5ed113592bfed10 - (/mnt/usr/bin/mc68020) - 1 match(es)

    canonical-path: /usr/bin/ps
    package: SUNWcsu
    version: 11.6.0,REV=1997.07.15.21.46
    architecture: sparc
    source: Solaris 2.6/SPARC

/mnt/usr/bin/mc68020 = /usr/bin/ps
--------------------------------------------------------------------------------
```

By examining the output of the Solaris Fingerprint Database from `sfpC.pl`, one can discern between files that "match" (most likely legitimate processes) and those with "zero (0)" or no matches in the Solaris Fingerprint Database. The non-matching files in this example are files that were either started or left behind by the suspected intruder.

```
--------------------------------------------------------------------------------
    ac9ef190901c01a2c88674c4af38f063 - (/mnt/usr/bin/mc68030) - 1 match(es)

    canonical-path: /usr/bin/find
    package: SUNWcsu
    version: 11.6.0,REV=1997.07.15.21.46
    architecture: sparc
    source: Solaris 2.6/SPARC

/mnt/usr/bin/mc68030 = /usr/bin/find
--------------------------------------------------------------------------------
    6249c10975ac50b8fffe408342191e5b - (/mnt/usr/bin/mc68040) - 1 match(es)

    canonical-path: /usr/bin/netstat
    package: SUNWcsu
    version: 11.6.0,REV=1997.07.15.21.46
    architecture: sparc
    source: Solaris 2.6/SPARC

/mnt/usr/bin/mc68040 = /usr/bin/netstat
--------------------------------------------------------------------------------
    130a263319a918bd3e2bbf5cc5b2c1fa - (/mnt/usr/lib/vold/nsdap/pg) - 0 match(es)

Not found in this database.
--------------------------------------------------------------------------------
    cd63323c0eb2c25da98641ad701ee5a7 - (/mnt/usr/lib/vold/nsdap/sn2) - 0 match(es)

Not found in this database.
--------------------------------------------------------------------------------

    60811226259b44dc48bf63503f681d6b - (/mnt/usr/lib/vold/nsdap/sn.l) -0 match(es)

Not found in this database.
```

Other Solaris Commands That Help Computer Forensic Investigations

The following examples show how trojan binaries may differ from the following commands shipped by Sun.

```
/usr/ccs/bin/what(1)
/bin/strings(1)
```

Example use of system command `what(1)`:

```
root# /usr/ccs/bin/what /bin/login
```

Output from `/bin/login` from a Sun shipped (valid) binary:

```
/bin/login:
Sun OS 5.8 Generic 111085-02 November 2001
root# /usr/ccs/bin/what /mnt1/bin/login
```

Output from the suspect system's `/bin/login` binary:

```
/mnt1/bin/login:
Copyright (c) 1980, 1987, 1988 The Regents of the University of
California.
login.c          5.32.1.1 (Berkeley) 1/28/89
Sun OS 5.5 Generic November 1995
```

In this example, note that validation using MD5 showed that this file was not a Sun shipped binary, and using `what(1)` showed that this binary was actually created from source code from a BSD system.

One can perform the same trick against other suspect system binaries and obtain similar results. For example:

```
root# what /mnt1/bin/ls
```

Output of `what(1)` on the `ls` binary on the suspect system:

```
ls:

Sun OS 5.6 Generic August 1997

root# what /bin/ls

/bin/ls:

Sun OS 5.8 Generic February 2000
```

In this example, note that the intruder installed a Solaris 5.6 (Sun OS 5.6) `ls` binary in place of the Solaris 8 (Sun OS 5.8) system binary shipped by Sun in the Solaris 8 OS.

Another example shows using `what(1)` on the `/bin/password` binary:

```
root# what /mnt1/bin/passwd
```

Output from the `/bin/password` binary on the suspect system:

```
/mnt1/bin/passwd:

Sun OS 5.6 Generic August 1997

root# what /bin/passwd
```

Output from a known good (valid) Solaris `/bin/password` binary:

```
/bin/passwd:

Sun OS 5.8 Generic 111659-05 December 2001
```

The following example uses the `strings(1)` command to examine several suspect binaries:

```
root# strings -a /mnt1/bin/login |more
```

Output using `strings(1)` on the suspect `/bin/login` binary:

```
/var/spool/lp/.lpr

[file]

find

file_filters

[ps]

ps_filters

[netstat]
```

```
netstat
net_filters
```

Output using `strings(1)` on the suspect `/bin/password` binary:

```
Root# strings /mnt1/bin/passwd |more
/usr/lib/libX.a/uconf.inv
[file]
find
file_filters
[ps]
ps_filters
[netstat]
netstat
Net_filters
```

Ah ha! Note the reference to `/usr/libX.a/uconf.inv`. What is that doing in there?

This final example uses `strings(1)` with the "`-a`" argument:

```
Root# strings -a  /mnt1/bin/passwd |more
```

Output from the `strings -a` command on the suspect `passwd` binary:

```
lsof_filters
shell
/usr/lib/libX.a/bin/passwd
bexter
/bin/sh
```

Hmm. What in the heck is the word "`bexter`" doing in `/bin/passwd`? Again, note the reference to the `/usr/lib/libX.a` directory. This is not normal.

Inode Analysis

Unix and Solaris filesystems (UFS) use inodes to allow the file system to keep track of files. Inodes are created sequentially (inode #5 is created before inode #6). Use this fact to your advantage while conducting a computer forensic investigation.

When a file is created, a new inode is assigned to that file. The inode assignment is what the filesystem uses to keep track of where the file is located on the physical disk media. When a system is freshly installed, files are copied to the filesystem from the installation media (install CD-ROM) and the inodes that are consequently created are done so in sequential order. Thus, one would expect that, when the `/bin` directory was created and populated, the pattern of inodes for each of the files in /bin would show that the first file created would have a lower inode number, and that all the files in /bin that were created by the installation would have their inodes in the same sequential range. Inode numbers are not reused unless the inode table on a system has been exhausted. In common practice, the higher the inode number, the newer the file creation time. This is a rule of thumb (Your Mileage May Vary).

Invoke the "ls" command with the "-i" argument to show the corresponding inode associated with the file.

```
safe-system# ls -i /bin/am*"
```

Condensed output from `ls -i /bin/*`:

```
inode file
81609 amicert
81610 amicertify
81611 amidecrypt
81612 amiencryp
81613 amikeystore
81614 amilogin
81615 amilogout
81616 amisign
81617 amiverify
```

In this example output, note that, in a system installation, the inode numbers are sequential beginning with 81609. Each subsequent binary is created with an inode +1 as the files were copied onto the filesystem.

If the inodes are not in sequential order, this indicates that changes were made to the filesystem. In addition, this can point to where the file was originally created on the filesystem, and where it first appeared on the system.

When looking at suspect files, one could deduce not only at what relative order they were created, but also possibly where on the file system they were first created by examining inodes that are around the same number range.

```
safe-system# ls -li /usr/bin/lo*'
81413 -r-xr-xr-x   1 root     bin       172768 Jan  5  2000 /usr/bin/localedef*
81414 -r-xr-xr-x   1 root     bin         7204 Jan  5  2000 /usr/bin/logger*
81415 -r-s--x--x   1 root     bin        29200 Jan  5  2000 /usr/bin/login*
81416 -rwxr-x---   1 root     bin        12924 Jan  5  2000 /usr/bin/logins*
```

Notice how all the inodes are in sequential order. This is typical and normal.

Now let's look at the inode layout from our suspect system.

```
safe-system# ls -li /mnt1/usr/bin/lo*
81413 -r-xr-xr-x    1 root     bin       172768 Jan  5  2000 /usr/bin/localedef*
81414 -r-xr-xr-x    1 root     bin         7204 Jan  5  2000 /usr/bin/logger*
769843 -rwsr-xr-x   1 root     bin        28008 Mar 18 09:31 /usr/bin/login*
81416 -rwxr-x---    1 root     bin        12924 Jan  5  2000 /usr/bin/logins*
```

In this example, note that the inode numbering is out of sequence for `/usr/bin/login`. This is yet another strong indication that the original `/bin/login` program was replaced by another version, and the inode number 769843 also potentially tells us where the file was created.

Referring to the previous loop back mount of the suspect partition, look for an inode number sequence that roughly matches the same range:

```
safe-system# find /mnt1 -type f -exec ls -li {} >inode-list \;
```

In this example, there is a `/usr/lib/libX.a` directory whose contents has similar inode numbers. There are also other binaries with similar inodes. These are also suspected of having been modified or of even being Trojans.

```
769828 -rwsr-xr-x 1 root   bin      18844 Mar 18 09:31 ls
769835 -rwsr-xr-x 1 root   bin     101744 Mar 18 09:31 passwd
769842 -rwsr-xr-x 1 root   bin       9492 Mar 18 09:31 ps
769826 -rwsr-xr-x  1 root  bin      17156 Mar 18 09:30 su
```

These results point to a new directory to examine: `/usr/lib/libX.a`. Note that this is a directory and not what would be expected. One would expect that a "`libX.a`' would be an archive library or a "file" in a broader terminology, but instead it is a "directory".

Thus far, the forensic analysis supports the conclusion, with some degree of certainty, that changes to the system were not the result of an accidental modification by an administrator, and that there is at least a partial list of the files that have been replaced. Although not all of files that were examined or potentially modified is yet known, one can assume to be proceeding on the right track.

At this point, an investigator can make a decision to conduct a more in-depth analysis using more sophisticated forensic tools, or simply to re-install the production machine and get it back into production.

Internet Search Engines

One last tip. Internet search engines can provide pointers to others who might have suffered the same break-in and might have possibly already completed some of the same analysis that your organization is undertaking. Why not take advantage of this wealth of information? One can find information about what toolkit was used, and links to analysis that has already been done by others.

```
http://www.google.com
Search for  ' /usr/lib/libX.a bexter rootkit/
```

Step 6: Perform Recovery

This article describes the steps for performing a forensic analysis. The recovery efforts that ensue after the forensic analysis is a standard process that should be defined in your organization (and is beyond the scope of this article). In general, to ensure that your forensic analysis is a worthwhile effort, it is recommended that, when reinstalling your system, your organization should patch it, harden it, minimize it, and apply a defense in depth strategy. For more information about these tasks, refer to the Sun BluePrints OnLine Web site at:

`http://www.sun.com/software/security/blueprints/`

Conclusions and Recommended Best Practices

Hacking computer systems is especially commonplace in today's modern and network connected world. In most cases, hacking is illegal. Because perfect computer security is virtually impossible in a networked environment, it should be assumed that everyone is a potential target of hacker attacks. For this reason, organizations should be proactive, consider the recommendations offered in this article, and take steps to be prepared.

Prosecuting a hacker can be very difficult. Success depends largely on how well evidence is gathered, processed, and presented so that it is admissible in a court of law. Therefore, one recommended approach is to conduct a computer forensic investigation according to the process outlined in this article.

For computer security, the best offense is a good defense. Your organization should always assume the existence of hackers who want to attack your systems. Given that stance, this article offers the following recommendations:

- Implement the most comprehensive auditing that your organization can afford. Review the audit trails regularly.
- Implement real time monitoring systems that can detect anomalous behaviors and intrusions.
- Define, document, and publish a comprehensive security policy for your organization. Make sure that it addresses audit trails, monitoring, intrusion detection, and other security risks.

- Make sure that all systems are bannered so that there are effectively no trespassing signs posted.
- Inventory all computer processing resources, including all data, and classify these according to a schema appropriate for the enterprise.
- Implement a comprehensive and holographic security architecture that protects hardware, software, network, application, data, and operational / resource management.
- Implement and regularly test a disaster recovery and business resumption/ continuity plan.
- Make sure that patches on all systems are up to date.
- Coordinate computer forensic investigations with a CERT (Computer Emergency Response Team).
- Implement a security awareness and training program so that personnel are trained to appropriately deal with a system intrusion.

Resources for Additional Information

Related Resources

TABLE 1 Resources

Resource	URL
Department of Justice Search & Seizure Manual dated January 2001	http://www.cybercrime.gov/ searching.html#FED_GUID
Computer Forensics Incident Response Essentials by Warren G. Kruse II and Jay G. Heiser. Published by Addison Wesley	
Handbook of Computer Crime Investigation - Forensic Tools and Technology by Eoghan Casy, et. al. Published by Academic Press.	
Computer Incident Response by Scott Grace	http://www.sans.org/infosecFAQ/ incident/IRCF.htm

References

TABLE 2 References

Reference	URL
NCIS Computer Crime	http://www.ncis.navy.mil/activities/ CompCrim/ CompCrim.html
HTCIA (High Technology Computer Investigation Association)	http://htcia.org/
Computer Forensics Online Magazine	http://www.shk-dplc.com/cfo/
CyberCrime	http://www.cybercrime.gov/

Tools Resources

TABLE 3 Tools Resources

Tool Resource	URL
Sun Service Security	http://www.sun.com/service/sunps/security
Project Honeynet	http://project.honeynet.org
The Corners Toolkit	http://www.fish.com/tct
Autopsy and task tools from @stake	http://www.atstake.com/research/tools/ forensic/
Bootable X86 CD	http://biatchux.dmzs.com/

About the Authors

Joel Weise

Joel Weise has worked in the field of data security for over 20 years. As a Senior Security Architect for Sun Professional Services, he designs system and application security solutions for a range of different enterprises, from financial institutions to government agencies. He specializes in cryptography and public key infrastructures.

Prior to joining Sun, Joel was a Senior Project Manager for Visa International, where he was responsible for developing cryptographic standards, designing key management and cryptographic systems, and architecting security solutions for chipcard, Internet, and other new products.

Brad Powell

Brad Powell has been in the computer and network security field for over 18 years. As Chief Security Architect for Sun Professional Services, he designs security solutions such as firewalls and security architectures. Other duties include security assessments, penetration studies, and computer forensics for banks, industry, and government agencies.

Formerly at Sun, Brad held the position of Network Security Engineer, designing Sun's firewall, security architecture, and network security policies. Duties also included electronic intrusion detection and prevention, implementing security solutions on thousands of internal Sun networks, computing platforms, and applications, and assisting law enforcement agencies worldwide in investigating computer crime. Brad is a member of the HoneyNet project and author of many freeware security tools. For more information, see:

http://www.fish.com/~brad

http://www.honeynet.org

Ordering Sun Documents

The SunDocs^{SM} program provides more than 250 manuals from Sun Microsystems, Inc. If you live in the United States, Canada, Europe, or Japan, you can purchase documentation sets or individual manuals through this program.

Accessing Sun Documentation Online

The `docs.sun.com` web site enables you to access Sun technical documentation online. You can browse the `docs.sun.com` archive or search for a specific book title or subject. The URL is `http://docs.sun.com/`

To reference Sun BluePrints OnLine articles, visit the Sun BluePrints OnLine Web site at: `http://www.sun.com/blueprints/online.html`