Use offense to inform defense.
Find flaws before the bad guys do.

Covering the Tracks on Mac OS X Leopard

GCIH Gold Certification

Author: Charles Scott, cscott@infosec.utexas.edu

Adviser: Don C. Weber

Accepted: June 16, 2008

OUTLINE

1. Abstract

Many systems administrators are not aware of the subtle differences between Mac OS X and its Unix operating system brethren (Jepson, Rothman, & Rosen, 2008). Hackers can exploit this ignorance when hiding their presence on compromised systems (Skoudis, 2007). In this paper, I apply the "Covering the Tracks" techniques described in the SANS SEC 504 course to Mac OS X. Doing so highlights the ways in which Mac OS X and Unix diverge, increasing awareness of how an attacker might conceal himself in Mac OS X. The goal is to improve vigilance among systems administrators so they can better prepare for and identify intrusions on Mac OS X systems.

2. Introduction

Apple's Mac OS X is an increasingly popular operating system (Elmer-Dewitt, 2008). An offshoot of BSD, Mac OS X has a similar directory structure, analogous configuration files, and comparable logs and databases to other Unixes; however, Apple has made a few changes that make Mac OS X its own beast (Jepson, Rothman, & Rosen, 2008).

Ed Skoudis' material for "SANS Security 504: Hacker Techniques, Exploits & Incident Handling" provides examples and hands-on exercises demonstrating how an intruder might hide files, edit logs, modify accounting databases, and clear shell histories on a Unix system in order to cover their tracks (2007). Knowing how a hacker might do this keeps a systems administrator vigilant for such attacks and assists in the Preparation and Identification phases of the incident handling process (Skoudis, 2007).

This paper will apply the "Covering the Tracks" section of SEC 504 to Mac OS X Leopard. It will determine what works from these examples and how any discrepancies might be addressed. It will also illustrate Mac OS X specific functions and features that an intruder might have to tackle in order to hide what they have been doing, such as: The asl.db logging database, the HPF+ journaling file system, Spotlight metadata, and unique directories (Apple, 2004; Royer, 2008; Jepson, Rothman, & Rosen, 2008; Davisson, 2005). Note that only Mac OS X 10.5 Leopard is covered, as there are subtle differences in older versions of Mac OS X, and covering all of them is beyond the scope of this paper (DC1743, 2007). After reading this paper, a systems administrator or security professional will have a good idea of how an intruder might cover their tracks on a Mac OS X system.

3. A Quick Overview of Getting File Information in Mac OS X

For those that are new to Mac OS X, a quick overview of how it lets you view and search for file information might be in order.

The Finder

In Mac OS X, files and folders are most often manipulated with the Finder, which is the Mac graphical user interface (see Figure 1).

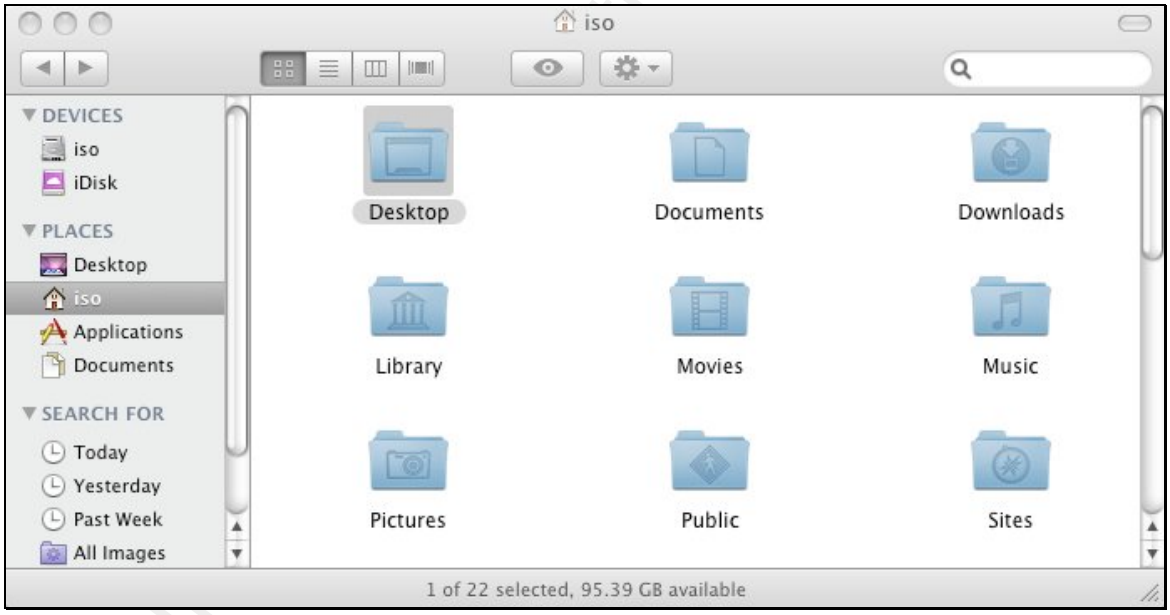

Figure 1: The Mac OS X Finder.

To get specific information on a file or folder, you select that object and then either right-click and select “Get Info”, or use the keyboard shortcut Command - I (⌘ - I). This shows information on the file similar to what you see in Figure 2, such as modification and

creation date, file size, permissions, and metadata that is not normally visible in a file system directory listing, such as file kind and label.

Figure 2: The "Get Info" feature of the Finder.

The Terminal App and Command-Line Interface

Command-line file operations under Mac OS X are performed using the Terminal application. Terminal is found in the Utilities sub-folder of the Applications folder. From there, files can be listed using the standard Unix `ls` command. The `ls` command with the "-l" switch will produce a listing of files with several of their attributes, such as modification time, size, and permissions. Figure 3 illustrates this.

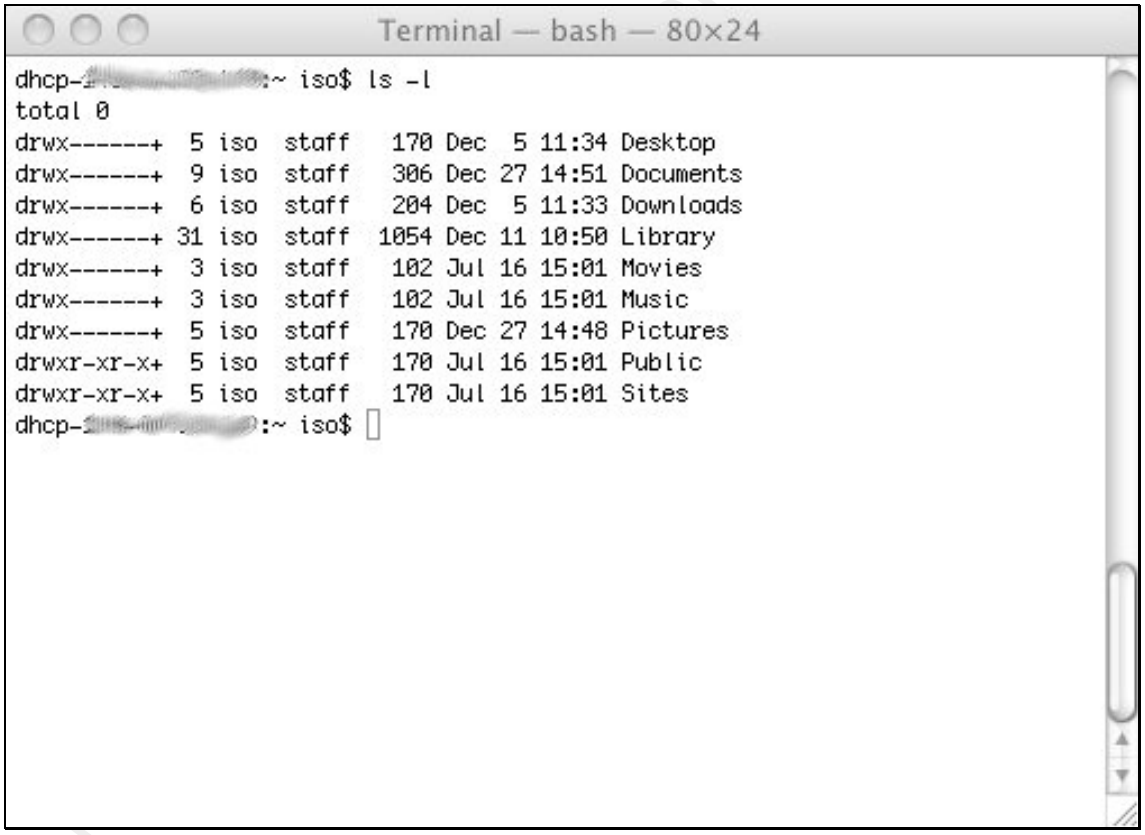

Figure 3: The Terminal app and "ls -l" output.

Spotlight

Spotlight is the Mac OS X file and directory searching tool that was introduced in Tiger. It goes beyond typical Unix file searching tools such as `find`, which is slow, and `locate`, which may miss recently added files, by storing file metadata in a quickly-accessible format (Jepson, Rothman, & Rosen, 2008). This is nice for users that need to find something posthaste, but could be the bane of an attacker if an administrator accidentally stumbles upon their tools in Spotlight's suggestions list when searching for something else.

Spotlight is accessed by clicking on the magnifying glass in the upper right-hand corner of the desktop. Doing so produces a search field, which can be filled with any text related to the file for which you are searching (see Figure 4).

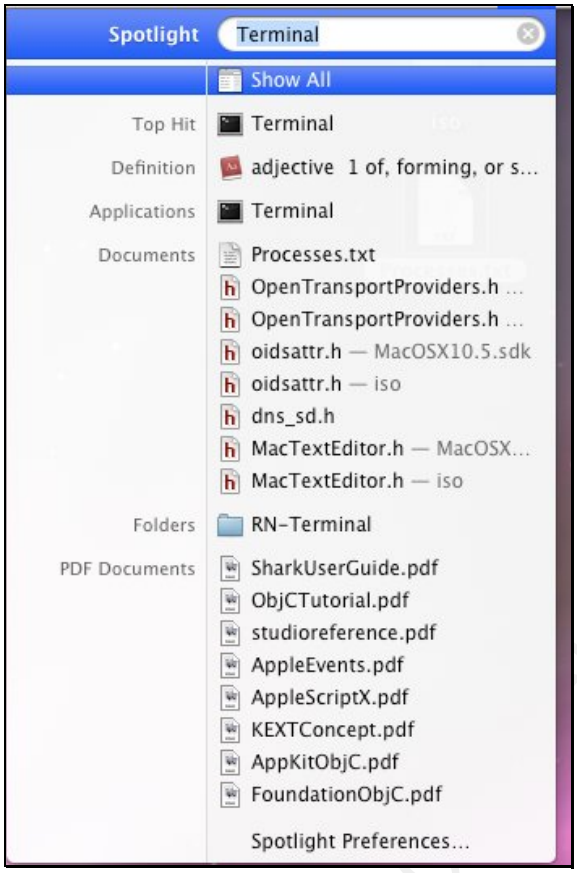

Figure 4: Spotlight search results.

Clicking on "Show All" in the results allows you to refine the search further by location, date, or size (see Figure 5).

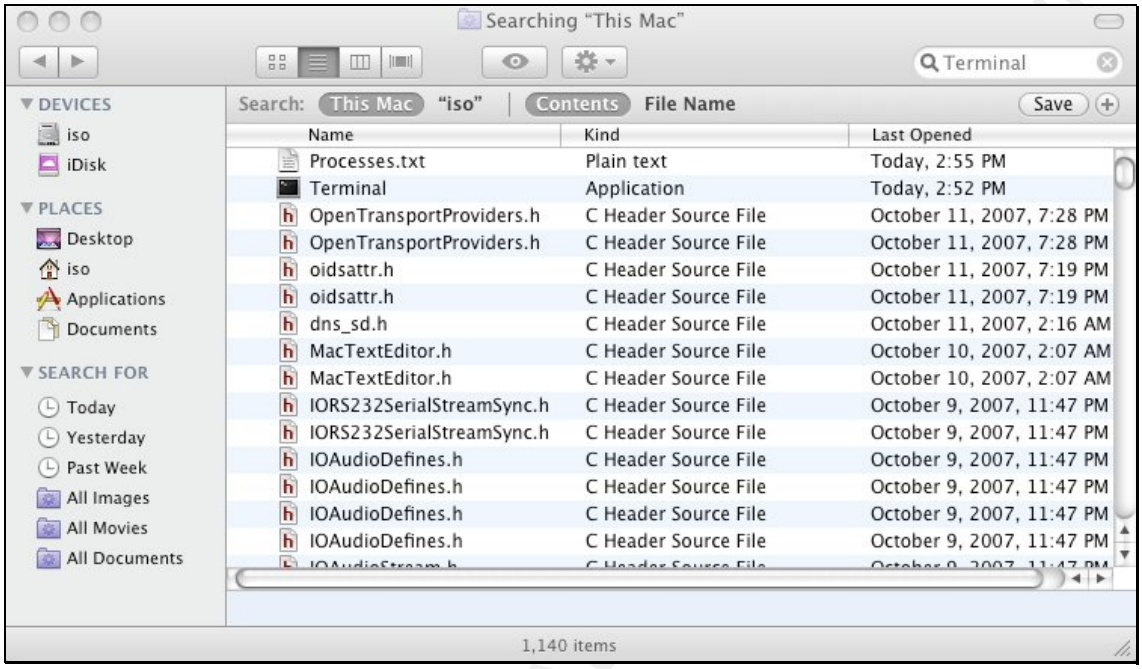

Figure 5: Narrowing Spotlight search results in the "Show All" window.

Spotlight is configured through the Spotlight control panel under System Preferences. One of the options is the ability to exclude certain folders from Spotlight's indexing. This is accomplished by dragging-and-dropping the folder into the Privacy area of the control panel (see Figure 6).

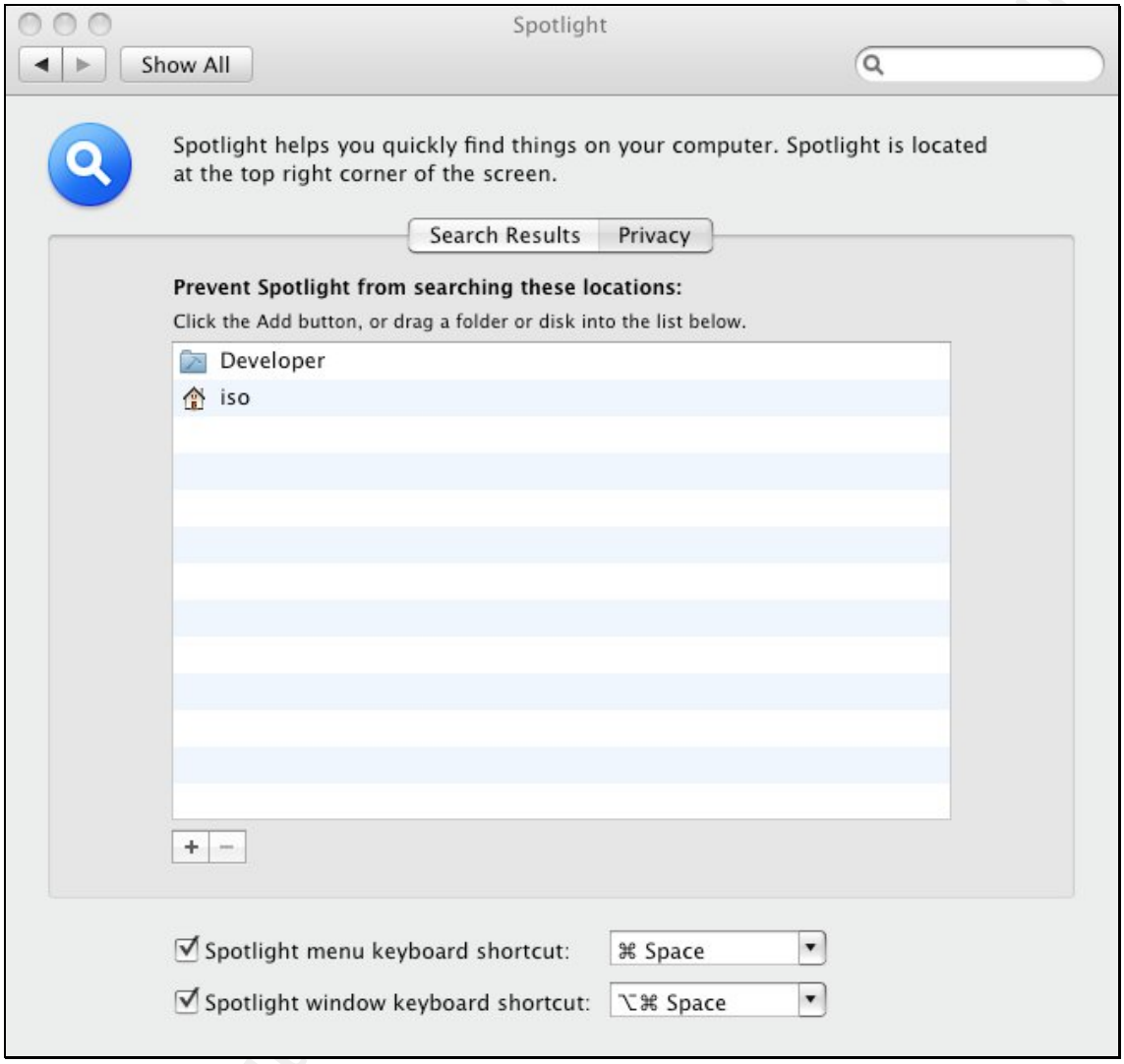

Figure 6: The Privacy settings in Spotlight.

4. Hiding Files and Directories in Mac OS X

A common method for hiding files and directories in Unix is to use a file or directory name that starts with a dot (Skoudis, 2007). While system applications such as SSH normally use this

to hide configuration, state, or temporary files from users (and thus prevent the user from accidentally deleting the file), the users themselves can also do this to remove a file from view. These files and directories can only be seen in an `ls` directory listing by using the "-a" switch. As a Unix-based operating system, Mac OS X also has the ability to hide files using a prepended dot. For instance, an attacker could hide a directory called "tmpx" by putting a dot before it, this hiding it from a standard `ls` listing, as well as the Finder.

```
$ mkdir .tmpx
$ ls
$ ls -a
.     ..    .tmpx
```

The "dot-dot-space" and "triple-dot" obfuscation methods demonstrated by Skoudis in SEC504 both work in Mac OS X as in other Unix systems, as shown in the following example (2007):

```
$ echo hidden data > ".. "
$ echo more hidden data > "..."
$ ls
$ ls -a
.     ..    ..    ...
$ cat ".. " "..."
hidden data
more hidden data
```

The idea behind these files is to make them look similar enough to the directory

shortcut representation schemes used by Unix that most users might overlook them. A vigilant Unix systems administrator using the "-a" switch, however, might spot them. There is one catch with Mac OS X: The Finder does show "dot-dot-space" files, even though an `ls` listing does not. The same is be true for "dot-dot-anything" files, such as "..ticktock" (see Figure 7). This author has not been able to uncover any reason why double-dot files are not hidden from the Finder the way dot files are. At any rate, a second step must be used to hide these files from the Finder.

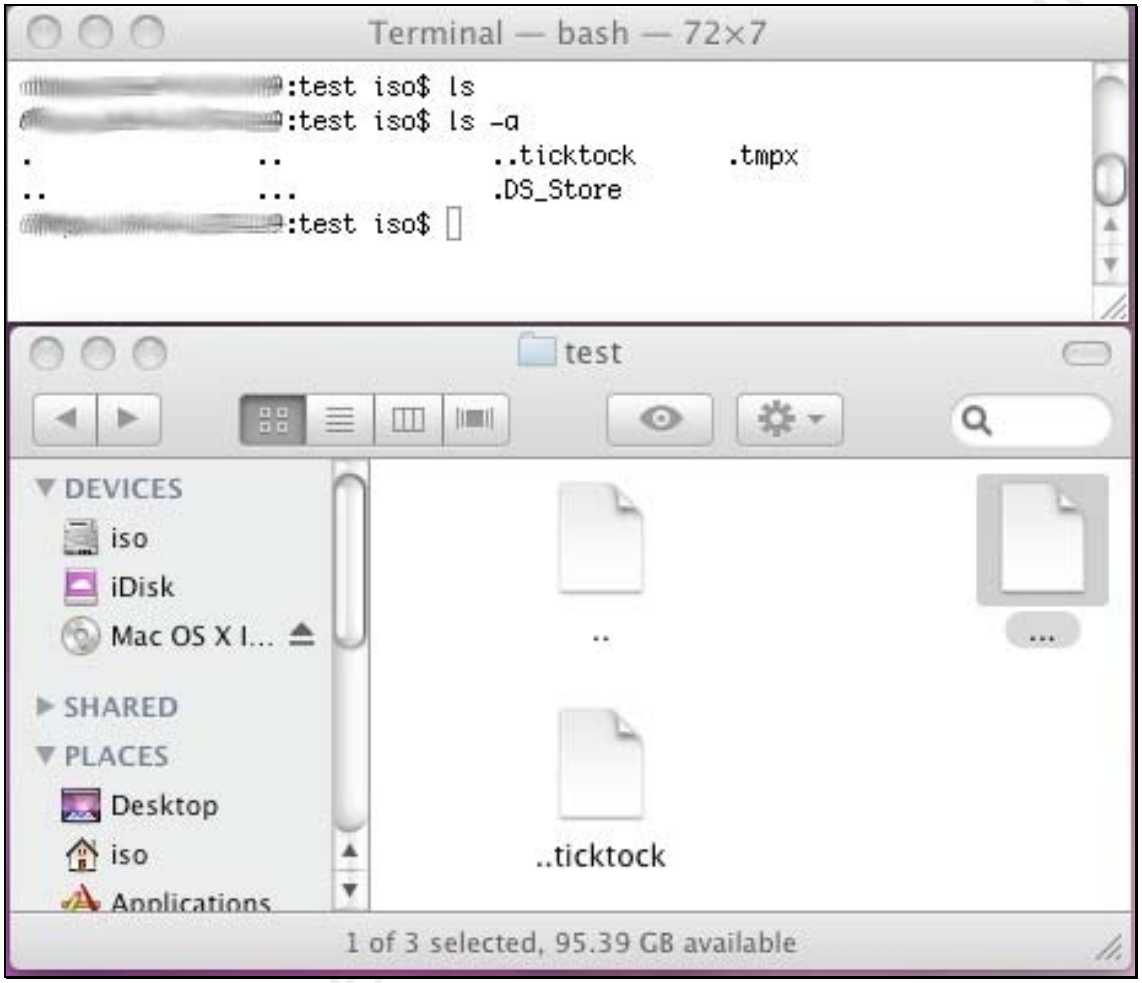

Figure 7: Double-dot files do are hidden from "ls", but not "ls -a" or the Finder.

Hiding Files from the Finder Using File Attributes

The Apple HFS+ file system used by Mac OS X provides for extended attributes that go beyond the typical Unix file attributes. By setting the "invisible" attribute, a file or directory becomes hidden from the Finder. These attributes are viewed and manipulated by two utilities: `GetFileInfo` and `SetFile`. Both of these are part of the Apple Developer Tools

available on the Leopard install DVD, or from Apple's Developer website (Jepson, Rothman, & Rosen, 2008). If an attacker does not find that the Developer Tools have already been installed on the compromised system, they can easily upload `GetFileInfo` and `SetFile` from an existing installation.

The existing attributes for a file can be viewed using the `GetFileInfo` command. For example, this is a view of the attributes for a "dot-dot-space" file.

```
$ GetFileInfo ".. "
file: "/Users/iso/test/.. "
type: ""
creator: ""
attributes: avbstclinmedz
created: 10/08/2008 11:44:41
modified: 10/08/2008 11:45:00
```

The "attributes" field shows all of the attributes that apply to that file. A lowercase letter means that the given attribute is turned off, or false, while an uppercase letter means that the attribute is turned on, or true. The letter "v" represents the "invisible" attribute, as in make the file invisible to the Finder (Royer, 2008). Here it shows that the invisible attribute is set to off, meaning that the file will appear in the Finder.

To turn the invisible attribute on, the `SetFile` command is used, along with the "-a" switch:

```
$ SetFile -a V ".. "
```

Now, as you can see in Figure 8, the "dot-dot-space" file no longer appears in the Finder.

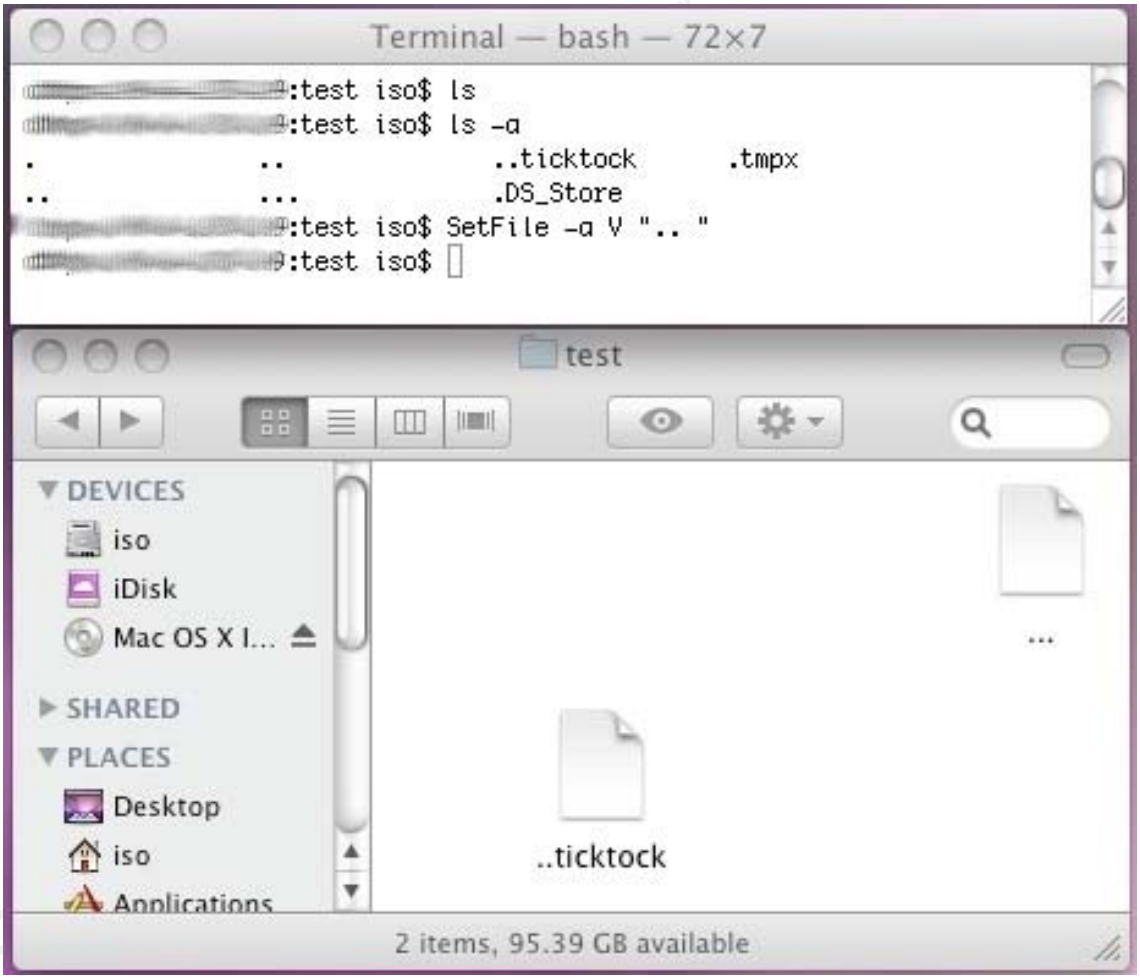

Figure 8: The double-dot-space file no longer appears in the Finder after using the SetFile command to turn on the "invisible" attribute.

In addition to making files invisible, `SetFile` can also be used to modify the creation and modification times of a file. File date and time modification is another trick that hackers use, especially on Trojaned binaries, to make it seem like a file has actually been on the system longer than it has, and thus more likely to escape the notice of a systems administrator.

Because `GetFileInfo` and `SetFile` come with the Developer's Tools, a valid question might be, "Should a systems or security administrator install the Developer's Tools in the first place?" On production systems the answer will generally be no. Some hardening guides recommend that compilers such as GCC be removed from production systems as a standard practice because attackers could use these tools to compile their rootkits or Trojans on your system (Turnbull, 2005). A better practice would be to have development tools installed on a well-secured development system that does not allow connections from the Internet (or restricts the connections to only those networks that need access), and has the same hardware and software specifications as the production system. Once the software you need has been compiled, copy the binaries from the development to the production system.

Unique Areas to Hide Files in Mac OS X

In addition to using dot names attackers will put files in locations where they will not be noticed, such as /dev, /tmp, /usr/local/man, and other areas where files are numerous and complex (Skoudis, 2007). Although Mac OS X is Unix under the hood, Apple has overlaid even more complexity to give it Mac features and personality. With added intricacy come even more opportunities to obfuscate files and directories, because even eagle-eyed Unix systems administrators might not be familiar with the more esoteric aspects of Mac OS X file structure (Jepson, Rothman, Rosen, 2008).

In addition to the standard Unix locations, the following directories are ideal places for an attacker to hide files. Mac OS X hides these directories from the Finder by default, making it even less likely a systems administrator would stumble upon them (Davisson, 2005). They are, however, visible from a Terminal window. The following table lists the common hidden directories in Mac OS X and their intended use.

/Network	Contains directories and shares that are network-mounted.
/Volumes	Contains visible file systems, including removable media and mounted disk images.
/cores	Contains core dumps from program crashes.

Ironically, the safest place to hide an executable on a Mac might be in the trash. Specifically, there is a directory created in the root level of all Mac disks called /.Trashes, which is used to temporarily store files on non-boot volumes that have been deleted by users (Davisson, 2005). Non-boot volumes include removable media, external hard disks, and network volumes.

What follows is a look at the permissions on the /.Trashes directory:

```
d-wx-wx-wt   2 root  staff        68 Oct 15 10:53 .Trashes
```

Notice that all users, including the "other" group, are only given write (w) and execute (x) permissions on the directory. Additionally, the sticky bit (t) is set, so that any directories or files created by a user can only be deleted or renamed by the owner of the file or directory, the owner of the .Trashes directory (in this case "root"), or the superuser (BSD, 1993). What this means from a practical standpoint for an attacker is that anything created in that directory cannot be seen or read by other users, or even the owner of the created object. The superuser would be able to see it, but because the .Trashes folder is not normally seen in the Finder, it is not likely that an Administrator will notice anything amiss.

What follows is an example of creating, hiding, and executing a very simple script in the .Trashes directory (the dollar-sign prompt indicates that the logged-in user is not the

superuser):

```
$ cat > /.Trashes/sekretskript
#!/bin/sh
echo "This works!"
^C
$ chmod 700 /.Trashes/sekretskript
$ ls /.Trashes
ls: .Trashes: Permission denied
$ /.Trashes/sekretskript
This works!
```

Finally, other locations that are not hidden but contain enough files that something might be overlooked include the following (Jepson, Rothman, Rosen, 2008):

/Applications	Mac OS X applications.
/Developer	Development tools (exists if XCode is installed).
/Library	Files that support installed applications.
/System/Library	More files that support installed applications.

Securely Removing Files

Mac OS X has a nice built-in tool for securely removing files called `srm` (secure remove), which overwrites, renames, and truncates a file before removing it (Apple, 2008). It behooves an attacker to use this tool when removing any files they have left behind, or log files they want to wipe, because it makes it forensically difficult to recover those files.

Using `srm` is very much like using the Secure Empty Trash option in the Finder; whereas, Secure Empty Trash uses a seven-pass overwrite with random data by default, `srm` uses a thirty-five-pass overwrite with random data by default.

For instance, if an attacker wants to remove a dot-dot-space directory that they have created, they would use something similar to the following:

```
$ srm -vrf ".. "
removing ..
```

The options for `srm` are very similar to those for `rm`, the standard Unix tool for removing files. The "-v" means to produce verbose output of what it is doing, which is why we saw the "removing .. " line. The "-r" option means to delete the entire directory hierarchy recursively, and the "-f" means to forcefully remove the directory without prompting for confirmation, regardless of the permission. There are other options, such as perform a

single-pass overwrite ("-s") or a seven-pass overwrite ("-m").

Writing to a deleted file

A common hacker trick is to delete a file after a process has opened it for writing (Siles, 2006). When this is done, the file will not be visible on the file system, but the process will still have the file descriptor open and can read from it and write to it. For example, in our Terminal window (which we will label "A") we can use the ls command to see that no file exists, then the `cat` command to redirect standard input to a file called "hidden-output.txt":

```
$ ls
$ cat > hidden-output.txt
```

Leaving `cat` running, we can open another Terminal window (we will call this one Terminal B) and see that the file exists on the file system, then tail that output:

```
$ ls
hidden-output.txt
$ tail -f hidden-output.txt
```

If we go back to Terminal A and type in "Some hidden output" and hit the Return key, that text will appear in Terminal B's tail output. Now, if we open up another Terminal window

(this one will be Terminal C) and delete the “hidden-output.txt” file, we can see that it no longer exists:

```
$ rm hidden-output.txt
$ ls
$
```

But we can still enter data in our Terminal A window and hit Return, and have it appear in Terminal B’s tail output. This shows that even though the file is deleted, the file descriptor is still available for reading and writing.

One way to see that a file is open is to use the `ps` command, where the file name may appear in the running command. Another way is to use the `lsof` (list open files) utility that comes with Mac OS X, which reveals what files handles processes have open:

```
$ lsof |grep hidden
cat  14209  iso  1w  REG  14,5  43  197792  /Users/iso/hidden-output.txt
```

5. Editing log files in Mac OS X

Like many Unix systems, Mac OS X uses the `syslogd` daemon to log system information in the /var/log directory (Jepson, Rothman, Rosen, 2008). Also like many Unix systems, the configuration file for `syslogd` is the /etc/syslog.conf file. An attacker would first

check the syslog.conf file to find out the names and locations of the logs being generated, then edit or remove those logs (Skoudis, 2007). On Mac OS X, an attacker is primarily concerned about two files: system.log and secure.log. The system.log file contains most of the system notices, kernel debug information, and login information. Authentication and authorization information is also logged to the secure.log file. In addition, several daemons and applications have their own log, such as mail.log and ftp.log, which may contain information logged during the attack. Also, there are logs for the application firewall (appfirewall.log) and BSD firewall (ipfw.log) (Maintain, 2008).

As with Unix, these logs are all ASCII and can be removed, edited with a text editor such as VI or EMACS, or modified with a script (Skoudis, 2007). In order to do this, however, an attacker would need to stop and restart `syslogd`. On other Unix systems running the `syslogd` initialization script, or simply killing the `syslogd` process and restarting it, accomplishes this. On Mac OS X, the best way to restart `syslogd` is to use the `launchctl` utility as the superuser (Simmons, 2008).

To stop `syslogd`, type the following:

```
$ sudo launchctl unload /System/Library/LaunchDaemons/com.apple.syslogd.plist
```

To start syslogd, type the following:

```
$ sudo launchctl load /System/Library/LaunchDaemons/com.apple.syslogd.plist
```

A Log File Unique to Mac OS X: asl.db

Besides the ASCII syslog files, Mac OS X's syslog also creates a database file called asl.db. This is a binary database file, so it cannot be edited through a standard text editor (Apple, 2004). The attacker has two choices here: They can delete asl.db, which might be noticed, or they can attempt to prune entries using the `syslog` utility.

In order to prune entries, first the attacker must shut down `syslogd` using the method described in the previous section. After `syslogd` has been shut down, they can prune certain things from syslog using the "-db" and "-p" switches, as well as a key/value expression (Apple, 2004). This, for instance, will remove all entries associated with `sshd`:

```
# syslog -db -p -k Sender sshd
```

This one will remove all authentication entries:

```
# syslog -db -p -k Sender com.apple.SecurityServer
```

While this command will remove sudo entries, which could contain a list of superuser commands used by the attacker:

```
# syslog -db -p -k Sender sudo
```

When finished, the attacker needs to restart the syslog daemon, as described in the

previous section. Additional commands for pruning asl.db can be found in Appendix A.

6. Clearing Shell Histories in Mac OS X

As mentioned in the SANS 504 course, the Unix shells often keep a record of each command entered. This is to make it easier for users to go back and quickly review or run previous commands, but it also provides a means for an investigator to go back and see what commands an intruder ran (Skoudis, 2007). Leopard uses BASH as the default shell, so methods an attacker might use to cover their tracks are very similar to those outlined in the course. As with many Unix systems, the default size of the .bash_history file is 500 lines.

There are two widely used methods to clear shell histories (Skoudis, 2007). The first is to kill, rather than exit, the shell so that it cannot write the most recent shell history. Here is an example where we are editing a script, then run it.

```
$ chmod 700 ./sekretskript
$ ./sekretskript
running script...
```

If we check the last five running history entries, we see the commands in there.

```
$ history 5
  376  finger
  377  w
  378  chmod 700 ./sekretskript
```

```
  379  ./sekretskript
  380  history 5
```

Now we kill our BASH process so that it does not write the shell history to disk:

```
$ kill -9 14322
Connection to x.x.x.x closed.
```

If we log in again and check the history, we do not see the most recent commands in there.

```
$ ssh iso@x.x.x.x
Password:
Last login: Thu Dec 11 16:07:48 2008 from ____.____.____
$ history 5
  344  diskutil
  345  diskutil
  346  ps ax
  347  sudo su
  348  history 5
```

The other way to avoid logging to .bash_history in the first place is to set the BASH history size (HISTSIZE) variable to zero. What follows is an example of this under Leopard:

```
$ export HISTSIZE=0
$ chmod 700 ./sekretskript
$ ./sekretskript
running script...
$ exit
logout
Connection to x.x.x.x closed.
```

Now if we log back in and check the previous history.

```
$ ssh iso@x.x.x.x
Password:
Last login: Thu Dec 11 16:12:05 2008 from x.x.x.x
$ history 5
    1  history 5
```

No history exists except the command we just typed.

As demonstrated, the shell history avoidance methods described in SANS 504 work very well under Leopard.

7. Editing Accounting Files in Mac OS X

On Unix systems the accounting files provide information on user logins. Traditionally, these have been separated into at least three files: utmp file, which contains information on currently logged-in users; the wtmp file, which contains entries about past logins; and lastlog, which contains the name, port, and last login time for each user (Skoudis, 2007).

Mac OS X up to version 10.4 also had utmp, wtmp, and lastlog files. These were deprecated, however, in 10.5 and replaced with /var/run/utmpx for currently logged in users and asl.db for the functions of wtmp and lastlog (DC1743, 2007). This means that tools created to edit the standard Unix accounting files, including those tools mentioned in the SANS 504 coursework, will not work under Mac OS X Leopard because of file type, location,

and API difference. Of course, one could simply shut down syslog and remove asl.db entirely, but this would likely be even more obvious than removing wtmp, because it contains other log entries besides just accounting.

Once again you can use the `syslog` command to prune entries out of asl.db, and therefore the virtual wtmp. This command removes wtmp entries caused when someone logs into the GUI at the initial login screen:

```
syslog -db -p -k Sender loginwindow
```

The following command removes wtmp entries generated by Terminal windows:

```
syslog -db -p -k Sender login
```

Finally, this using the sshd expression mentioned earlier will remove wtmp entries generated by SSH logins:

```
syslog -db -p -k Sender sshd
```

8. Hiding Files and Directories from Spotlight

It's fairly easy to configure Spotlight to avoid certain areas you deem "private" through the Spotlight control panel under the System Preferences GUI (Jepson, Rothman, & Rosen, 2008). For an attacker, however, this presents two problems: The first is that it is extremely obvious to an administrator that something fishy is going on when they see a new directory under the privacy settings that did not exist there before; the second is that the attacker may

not have access to the GUI, only a shell. Fortunately for an attacker, there are several ways to subvert Spotlight through the command-line.

One method is to go back to the tried-and-true: Start a file or directory name with a dot. Doing so not only hides the file or directory from the Finder, but also Spotlight. This is true even if you place a file without a dot in a directory hidden with a dot. Also, even though the "dot-dot-space" method does not seem to hide files in the Finder, Spotlight ignores such files.

Another way is to turn off Spotlight's indexing completely using the `mdutil` command (Jepson, Rothman, & Rosen, 2008). For example, the following will turn off indexing on all disk volumes (you must be superuser):

```
# mdutil -i off -a
/:
      Indexing disabled.
```

The "-i" option is for indexing and the "-a" stands for all volumes.

The final way to hide files and directories from Spotlight is to make them invisible using file attributes. This not only renders them invisible to the Finder but to Spotlight as well. How to do this is described in the "Hiding Files from the Finder Using File Attributes" section of this paper.

9. Preparing for and Identifying the Techniques in the Paper

Most of the attacks described in this paper rely on the fact that system administrators are not omniscient and may miss hidden or modified files, not review logs regularly, or otherwise not know when something changes. There are actions a systems administrator can take, however, that will reduce the chances of this happening, or at least noticing when they do.

Look into MacPorts

If you want to compile your favorite security tool under Mac OS X, it can sometimes be a challenge to find the right dependencies and compiler settings—especially if the tool was originally designed for another flavor of Unix. MacPorts is an open source software packaging system for the Mac OS X designed to make it easy to compile, install, and configure a wide variety of open source software available for other Unixes under Mac OS X (MacPorts Project, 2008). There are a number of detection tools available through MacPorts, some of which we will cover in this section, so it is highly recommended you install it on your system.

Check for modified files

A host-based intrusion detection system (HIDS) or file integrity checker can assist a systems administrator in finding files that have changed. One such tool is OSSEC HIDS,

which combines a rootkit checker, file integrity checker, and log file analyzer all in one free and open source package. It works on most flavors of Unix, including Mac OS X (Third Brigade, 2008). OSSEC's file integrity check can alert an administrator that a file has changed by sending an e-mail. While you would not want to run this in a directory that contains content that changes often, such as /tmp or /.Trashes, it might be useful to add a few directories that are Mac OS X specific to her list of monitored directories. To do this under OSSEC, open your ossec.conf file (typically in /var/ossec/etc) in a text editor. Under the <syscheck> section, look for the <directories> lines and add a line similar to this:

```
<directories check_all="yes">/Applications,/Developer</directories>
```

This will check the /Applications and /Developers directories (which should rarely change) for anything new or changed.

Other integrity checkers available for Mac OS X include Tripwire and AIDE, both of which are available directly through MacPorts (Gray, et al., 2008).

Look for Strange Files in Strange Places

One way to combat hidden files and directories is to actively look for them. This can be as simple as running the `find` command with a syntax similar to the following (BSD, 2006):

```
# find / -name "..." -print
```

```
/private/tmp/...

/Users/iso/...
```

This command will search the entire file system for triple-dot files or directories; two were found in this example. This command can be automated through a cron job and scripted to e-mail the output to an administrator.

Another way to find suspicious files is to use a rootkit detector, such as the one included with the OSSEC HIDS mentioned in the previous section. OSSEC includes a file called rootkit_files.txt, normally found in /var/ossec/etc/shared, that contains a list of specific files and directories that are included with common rootkits (Third Brigade, 2008). Here is an example from rootkit_files.txt:

```
usr/include/.../       ! Bobkit Rootkit ::rootkits/bobkit.php
```

This looks for a triple-dot directory in /usr/include, which is indicative of the Bobkti Rootkit.

If you want to have OSSEC warn you about any triple-dot files or directories that show up, you can add lines to rootkit_files.txt that looks like this:

```
*/…         ! Triple-dot file found

*/…/        ! Triple-dot directory found
```

These will search for any instance of a triple-dot file or triple-dot directory across the entire file system.

Other rootkit detectors available for Mac OS X include the venerable (in Internet terms) chkrootkit, which is also available in MacPorts (Murilo & Steding-Jessen, 2007), and the relative newcomer Rootkit Hunter (Boelen, 2008), which is not in MacPorts so it must be compiled by hand.

Log syslog messages elsewhere

A good way to prevent an attacker from compromising the integrity of your logs is to send a copy of the logs elsewhere. In Unix systems, this is normally accomplished by configuring the syslog daemon to send the logs to another machine; this can be done with Mac OS X as well (Gray, et al., 2008).

Simply edit the /etc/syslog.conf file and use a syntax similar to this:

```
*.notice;authpriv,remoteauth,ftp,install.none;kern.debug;mail.crit \
@1.2.3.4
```

This will send the information normally sent to the /var/log/system.log file to the remote syslog server with the IP address 1.2.3.4. A hostname can also be used in lieu of an IP address. Of course, you will also need to configure the receiving server to allow the

connection.

While a user with sufficient privileges may just be able to turn off syslog, all messages up to that point will have been copied to the remote syslog server for the administrator to review later. The attacker would also have to break into the remote syslog server in order to completely cover their tracks.

10. Conclusion

Mac OS X administrators should not think that they are immune to the subterfuge techniques used by Unix intruders. As demonstrated, most of the methods for covering tracks in Unix described in the SANS 504 course also work under Mac OS X. Likewise, seasoned Unix administrators should not believe they have all their bases covered just because they are aware of how an attacker would cover their tracks in Unix; there are many obscure hiding places for data on Macs that do not exist in the Unix world. Knowing what works and what does not will help any administrator prepare for an attacker and identify when their Mac OS X system might be compromised.

11. Appendix A: Useful syslog commands for pruning data from asl.db

Remove SSH connection entries, as well as SSH remote login entries in wtmp:

```
# syslog -db -p -k Sender sshd
```

Remove all authentication entries:

```
# syslog -db -p -k Sender com.apple.SecurityServer
```

Remove SUDO entries:

```
# syslog -db -p -k Sender sudo
```

Remove wtmp entries caused by a GUI login:

```
# syslog -db -p -k Sender loginwindow
```

Remove wtmp entries generated by Terminal windows:

```
# syslog -db -p -k Sender login
```

Remove entries with a specific process ID number:

```
# syslog -db -p -k PID <process ID number>
```

For instance, if the syslog entry shows this:

```
Fri Jan  2 10:45:17 localhost ntpd[25] <Notice>: time reset +0.247008 s
```

The process ID number is the number 25 located within the brackets. The process number can also be retrieved with the `ps` command.

Remove entries for a specific host (useful if the host they are being removed from is a remote syslog server):

```
# syslog -db -p -k Host <hostname>
```

Entries can also be removed based on time. For instance, if an attacker wants to remove all SUDO entries for the past 30 minutes, the following command will do so:

```
# syslog -db -p -k Sender sudo Time ge -30m
```

The “ge” stands for “greater than or equal to,” but an attacker could also use any of the following expressions:

eq equal

ne not equal

gt greater than

ge greater than or equal to

lt less than

le less than or equal to

In addition, the “m” stands for “minutes,” but an attacker could also use “s” for seconds, “h” for hours, “d” for days, or “w” for weeks (Apple, 2004).

12. References

Apple, Inc. (2004), man page for `syslog` retrieved on Mac OS X 10.5.

Apple, Inc. (2004), man page for `syslogd` retrieved on Mac OS X 10.5.

Apple, Inc. (2008). Mac OS X Security Configuration For Version 10.5 Leopard. PDF file retrieved on December 8, 2008 from the Apple, Inc. web site: http://images.apple.com/server/macosx/docs/Leopard_Security_Config_20080530.pdf

Boelen, M. (2008). Rootkit Hunter. Retrieved on December 27, 2008 from the Rootkit.nl web site: http://www.rootkit.nl/projects/rootkit_hunter.html

BSD (1993), man page for `sticky` retrieved on Mac OS X 10.5.

BSD (2006), man page for `find` retrieved on Mac OS X 10.5.

Davisson, G. (2005). Mac OS X Hidden Files & Directories. Retrieved on October 9, 2008 from the Westwind Computing web site: http://www.westwind.com/reference/OS-X/invisibles.html

DC1743 (2007). wtmp. Retrieved on September 30, 2008 from the Forensics from the Forensics From the Sausage Factory Blog: http://forensicsfromthesausagefactory.blogspot.com/2008/05/wtmp.html

Elmer-DeWitt, P. (2008). "Analyst: Apple's U.S. consumer market share now 21 percent." Retrieved on January 7, 2009 from the Fortune magazine's CNN web site: http://apple20.blogs.fortune.cnn.com/2008/04/01/analyst-apples-us-consumer-market-share-now-21-percent/

Gray, A., Korty, A., Edge, C., Steers, C., Fryer, N., & Conrad, E. (2008). Mac OS X 10.5 Security Checklist 1.1. Retrieved on December 9, 2008 from the SANS Institute S.C.O.R.E web site: http://www.sans.org/score/macosxchecklist.php

Jepson, B., Rothman, E., & Rosen, R. (2008). Mac OS X for Unix geeks, 4th edition. Sebastopol, CA: O'Reilly Media, Inc..

Maintain (2008). Manage log files. Retrieved on December 10, 2008 from the Maintain web site: http://www.maintain.se/cocktail/help/tiger/files/logs.html

Murilo, N. & Steding-Jessen, K. (2007). chkrootkit README. Retrieved on December 27, 2008 from the chkrootkit.org web site: http://www.chkrootkit.org/README

Royer, N. (2008). Methods of Showing and Hiding Files in Mac OS X. Retrieved on December 9, 2008 from the Project Mouse web site: http://projectmouse.org/2012/MethodsofShowingandHidingfilesinMacOSX

Siles, R. (2006). Hacking Challenges: A Training Tool. Retrieved on January 7, 2009 from the RaDaJo blog: http://radajo.blogspot.com/2006/09/hacking-challenges-training-tool.html

Simmons, D. (2008). Taming Leopard's Syslogd. Retrieved on November 17, 2008 from Daniel Simmons' Blog web site: http://www.spockboy.com/blog/archives/8

Skoudis, E. (2007). Hacker Techniques, Exploits, & Incident Handling. Bethesda, MD: The SANS Institute.

Third Brigade, Inc. (2008). OSSEC Frequently Asked Questions. Retrieved on November 14, 2008 from the OSSEC web site: http://www.ossec.net/wiki/index.php/FAQ

Third Brigade, Inc. (2008). OSSEC Manual. Retrieved on December 10, 2008 from the OSSEC web site: http://www.ossec.net/main/manual/

Turnbull, J. (2005). Hardening Linux. Berkeley, CA: Apress, Inc.

Last Updated: December 12th, 2018

Upcoming SANS Penetration Testing

Mentor Session AW - SEC542	Oklahoma City, OK	Dec 19, 2018 - Feb 01, 2019	Mentor
SANS Bangalore January 2019	Bangalore, India	Jan 07, 2019 - Jan 19, 2019	Live Event
SANS vLive - SEC504: Hacker Tools, Techniques, Exploits, and Incident Handling	SEC504 - 201901,	Jan 08, 2019 - Feb 14, 2019	vLive
Mentor Session @ Work - SEC560	Louisville, KY	Jan 10, 2019 - Mar 14, 2019	Mentor
Mentor Session - SEC542	Denver, CO	Jan 10, 2019 - Mar 14, 2019	Mentor
SANS Threat Hunting London 2019	London, United Kingdom	Jan 14, 2019 - Jan 19, 2019	Live Event
SANS Amsterdam January 2019	Amsterdam, Netherlands	Jan 14, 2019 - Jan 19, 2019	Live Event
SANS Miami 2019	Miami, FL	Jan 21, 2019 - Jan 26, 2019	Live Event
Cyber Threat Intelligence Summit & Training 2019	Arlington, VA	Jan 21, 2019 - Jan 28, 2019	Live Event
SANS Las Vegas 2019	Las Vegas, NV	Jan 28, 2019 - Feb 02, 2019	Live Event
SANS Security East 2019	New Orleans, LA	Feb 02, 2019 - Feb 09, 2019	Live Event
Community SANS Minneapolis SEC504	Minneapolis, MN	Feb 04, 2019 - Feb 09, 2019	Community SANS
Security East 2019 - SEC504: Hacker Tools, Techniques, Exploits, and Incident Handling	New Orleans, LA	Feb 04, 2019 - Feb 09, 2019	vLive
SANS SEC504 Stuttgart 2019 (In English)	Stuttgart, Germany	Feb 04, 2019 - Feb 09, 2019	Live Event
Security East 2019 - SEC542: Web App Penetration Testing and Ethical Hacking	New Orleans, LA	Feb 04, 2019 - Feb 09, 2019	vLive
Mentor Session - SEC560	Fredericksburg, VA	Feb 06, 2019 - Mar 20, 2019	Mentor
Mentor Session - SEC560	Boca Raton, FL	Feb 07, 2019 - Feb 22, 2019	Mentor
SANS Anaheim 2019	Anaheim, CA	Feb 11, 2019 - Feb 16, 2019	Live Event
SANS London February 2019	London, United Kingdom	Feb 11, 2019 - Feb 16, 2019	Live Event
SANS Northern VA Spring- Tysons 2019	Vienna, VA	Feb 11, 2019 - Feb 16, 2019	Live Event
Mentor Session: SEC560	Columbia, MD	Feb 16, 2019 - Mar 23, 2019	Mentor
SANS Scottsdale 2019	Scottsdale, AZ	Feb 18, 2019 - Feb 23, 2019	Live Event
SANS New York Metro Winter 2019	Jersey City, NJ	Feb 18, 2019 - Feb 23, 2019	Live Event
SANS Dallas 2019	Dallas, TX	Feb 18, 2019 - Feb 23, 2019	Live Event
SANS Secure Japan 2019	Tokyo, Japan	Feb 18, 2019 - Mar 02, 2019	Live Event
SANS Zurich February 2019	Zurich, Switzerland	Feb 18, 2019 - Feb 23, 2019	Live Event
Mentor Session - SEC504	Vancouver, BC	Feb 23, 2019 - Mar 23, 2019	Mentor
SANS Riyadh February 2019	Riyadh, Kingdom Of Saudi Arabia	Feb 23, 2019 - Feb 28, 2019	Live Event
SANS Reno Tahoe 2019	Reno, NV	Feb 25, 2019 - Mar 02, 2019	Live Event
SANS Brussels February 2019	Brussels, Belgium	Feb 25, 2019 - Mar 02, 2019	Live Event
Mentor Session - SEC542	Seattle, WA	Feb 26, 2019 - Apr 02, 2019	Mentor