# Global Information Assurance Certification Paper

# Solaris 10
# Filesystem Integrity Protection
# Using Radmind

GIAC Unix Security Administrator (GCUX)
Practical Assignment
Version 3.0 - Option 1



G. Samuel Wilson, Jr.
21 March 2005

## Table of Contents

## List of Figures

**Abstract**

This report is intended to provide information of value to security engineers who are choosing among various solutions to protect their Solaris systems from undesirable changes. In particular, the open-source product "Radmind" is described so it may be effectively compared to other, perhaps more well-known, commercial and open-source filesystem integrity applications.

Radmind seems to be most popular in the Mac OS X community, and much of the on-line documentation is heavily Mac OS X flavored. Therefore, a second objective of this report is to provide support for Solaris security administrators who choose to use Radmind, in the form of a "step-by-step" guide for the installation, configuration, and operation of Radmind on a Solaris 10 system. For this guide, a Solaris 10 server and client were used, but the guide should also be useful for older versions of the Solaris operating system or for other UNIX flavors.

## 1 Introduction

All security administrators must be concerned about unexpected and improper changes to the systems for which they are responsible. For certain systems operated by the US government, this concern is formally captured in the DCID 6/3. At all protection levels, system developers are required to incorporate "features and procedures" that protect the operating system and other security-relevant software from unapproved changes.[1] A filesystem integrity application is a valuable tool that can be a large part of the response to this requirement. There are a number of both commercial, off-the-shelf (COTS) products and open-source, "freeware," applications available to the system developer.[2]

Security engineers designing or developing systems for the US government also find that government customers, especially within the defense and intelligence communities, frequently specify the inclusion of COTS products only and exclude open-source applications. In some situations, though, a customer's budgetary restraints, the features of a particular open-source application, or other considerations may result in a contractor being directed or at least permitted to consider an open-source solution for inclusion in the information system under development.

With care, open-source solutions can cut costs without necessarily increasing the risks to the system. However, even where permission to use freeware products is granted, there may continue to be a stipulation that the products originate from within the United States. Permission to use software written by non-US citizens or controlled by entities outside of the United States can sometimes be obtained. Doing so generally requires additional paperwork to be filed and approved, which may pose problems to the schedule and budget of the program.

Three fairly well known and popular open-source file integrity checkers are Tripwire, AIDE, and Samhain. Tripwire is the "grand-daddy" among filesystem integrity checkers,[3] and although it has been very successfully incarnated as a commercial product,[4] there are still open-source versions available.[5] A more modern product very similar to the open-source version of Tripwire is AIDE (Advanced Intrusion Detection Environment),[6] which was discussed in depth in the "Securing UNIX" track course materials.[7] Another product, Samhain,[8] was recently the subject of a SANS GCUX practical.[9]

For each of these three, there is a barrier that may inhibit its use in a system being designed or developed for a US government customer.

There are several open-source versions of Tripwire. However, version 1.2 is old (1994), no longer supported, and hard to find. As of this writing, it may still be downloaded as a part of YASSP.[10] Version 1.3, the Academic Release Version of Tripwire, was not truly free for use in commercial projects and seems in any case to have completely disappeared from the web. More recently, the Tripwire Open Source Project has released Linux versions under the General Public License, but the last update is also several years old (2001).

The primary developers of AIDE and Samhain are citizens of Finland[11] and Germany,[12] respectively. The author intends them no disrespect whatsoever, but for the purposes of this study, the non-US source of these products must be considered, simply because they would be ineligible to be included in some US government information systems without undergoing a lengthy approval process.

Radmind has been mentioned in several forums recently as a possible alternative to Tripwire, AIDE, and Samhain.[13, 14, 15] It is a product of the University of Michigan and is still actively under development.[16] Therefore, it is a reasonable alternative as an open-source filesystem integrity checker where there is some concern about the age of the product or the country of origin.

Fyodor, in the Tripwire entry on his list of "Top 75 Network Security Tools", says:

> "UNIX users may also want to consider AIDE, which has been designed to be a free Tripwire replacement. Or you may wish to investigate Radmind."[3]

It is hoped that this report will provide some assistance to that investigation.

## 2 Environment

Although the environment and the particular network architecture are not of great consequence to this investigation, the setup envisioned is that typical of GIAC Enterprises as described in many of the papers submitted for the GCFW certification. One or more servers can be accessed from the Internet or another untrusted network. Private servers reside behind a firewall on a trusted network. Radmind client tools will be used to protect the public server, while the Radmind server is on the private network.

The Radmind server listens for TCP traffic from the clients on port 6662 by default. (It can be configured to use a different port number.) The firewall must be configured to open that port to TCP traffic inbound from a client to the Radmind server. The server never initiates a TCP session with its clients, so no outbound port has to be opened.

## 3 Overview of Radmind

The University of Utah maintains a web page with documentation about Radmind that includes the following definition.

> **"radmind** (**r**emote **admin**istration **d**aemon) is a collection of Unix command line tools that allow system administrators to manage file systems of multiple machines..."[17]

Radmind works much like Tripwire, Aide, or Samhain in that it checks the attributes of filesystem objects, compares those attributes to previously stored values, and flags any differences. Radmind offers a significant additional feature: if any filesystem object has been changed, it can restore that object back to its previous condition.

For directories, Radmind checks permissions, user ID, and group ID. For files, Radmind checks permissions, user and group ID's, modification time, and size. At the operator's discretion, it can also compute and compare a checksum for files. Radmind also handles other filesystem object types; the complete list is given in the man page for the `fsdiff` command.[18]

The Radmind executable runs on a central server. On the client hosts, Radmind installs a set of command-line tools, which are used to communicate with the server and to perform the filesystem checks and updates. Radmind can be compiled with support for Transport Layer Security, which enables the server and clients to authenticate each other and for the TCP traffic generated by Radmind to be encrypted.

Some of the important terms used frequently in all Radmind documentation are defined here to allow the reader to become familiar with them before they are encountered in the step-by-step guide below. A good introduction to Radmind can also be found at the University of Utah's web site, although it is oriented strongly toward Mac OS X.[19]

A **transcript** is a list of filesystem objects, formatted in a particular way. Each object occupies one line of the file. The contents of each line are: object type, directory path, and a list of the object's attributes. Lines starting with '#' are comments. Blank lines are allowed. Radmind transcripts have a `.T` filename extension.

If the transcript is an **applicable transcript**, each line may be preceded by a plus or minus sign. Applicable transcripts contain a list of filesystem objects that have been changed. The applicable transcript is the list of changes that must be made to restore the client back to a previous state.

A transcript may also be a positive or negative. A **positive transcript** is a list of filesystem objects and attributes that define the state to which the client will be restored if any changes are made. A **negative transcript** is similar to an exclusion list, in that directory contents will not be checked for directories on the negative transcript. However, the directory itself must exist and its permissions, user ID, and group ID will

be checked.

A transcript together with the filesystem objects that are listed in it forms a **loadset**. The loadset that fully describes the initial state of the client is called the **base load**. When new software is installed on a client or it changes in some other way, the security administrator may use Radmind to create a new loadset called an **overload**, which is then used together with the base load. This action may be less time-consuming than rebuilding an entire new base load for the client.

A set of transcripts is listed in a **command file**. By using both positive and negative transcripts together in a command file, the security administrator can achieve good control over just what actions Radmind will perform. Radmind command files have a `.K` filename extension.

Five of the Radmind command line utilities will be demonstrated in the step-by-step portion of this paper. They are `fsdiff`, `ktcheck`, `lcreate`, `lapply`, and lcksum. A brief explanation of the purpose of each utility is given here.

The **`fsdiff`** command reads a command file to get a list of transcripts. It checks the client's filesystem against those transcripts. Discrepancies are written to standard output but may be redirected to a file with the `-o` option. The fsdiff command may also be used to create transcript files.

The **`ktcheck`** command compares the client's command files and transcripts, as stored on the Radmind server, to the local command files and transcripts. If there are differences or if the local transcripts are not found, then they are retrieved from the server.

The **`lcreate`** command uploads a loadset to the server.

If there are changes to the filesystem, **`lapply`** accepts an applicable transcript as input and makes the indicated changes to restore the filesystem.

On the server, **`lcksum`** is used to verify that the attributes of uploaded files agree with what is shown in the matching transcript, i.e., that the uploaded load set is consistent.

Additional commands not demonstrated in this paper are `lmerge`, `lfdiff`, and `twhich`. These commands are used to merge loadsets, compare a single client file to its counterpart on the server, and to show the transcript associated with a file. One additional command, `applefile`, does not apply to Solaris clients.

# 4 Step-by-Step Guide for Radmind on Solaris

This section of the paper is intended to demonstrate the basic features of Radmind while at the same time providing a guide that can be followed to set up and operate Radmind.

What will not be done within the scope of this paper is to attempt to develop a definitive negative or positive transcript or a base load for a Solaris 10 machine. In any case, the "best" set of transcripts for a given machine at a given site may vary quite a bit depending on the network architecture, the threat environment, and the needs of the information system's users. However, the development of a basic negative transcript that could be used as a starting point by a Solaris 10 security administrator would be a good subject for some future work.

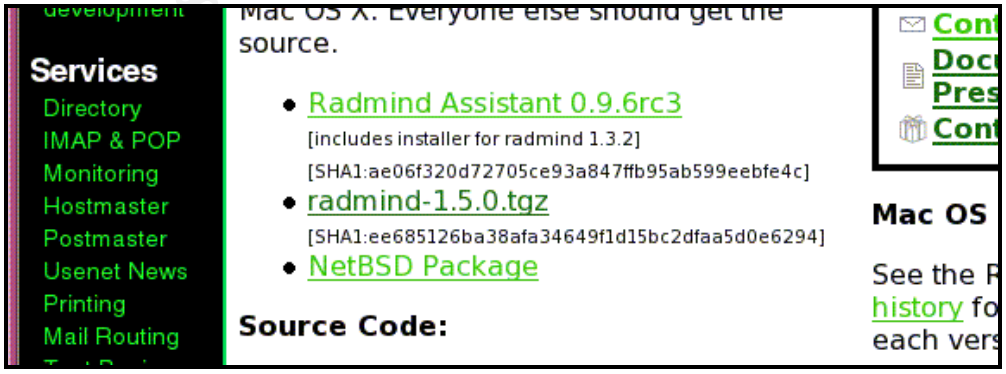
## 4.1 Installing Radmind

The download and installation of Radmind proved to be very straightforward for Solaris 10. No significant problems were encountered in making the executables from the source files. The `gcc` compiler is required, but it is a part of the Solaris 10 operating system, as is `openssl`. If those packages were not installed when the box was built, the administrator should add them before building Radmind.

### 4.1.1 Downloading Radmind

Radmind may be downloaded as `radmind-1.5.0.tgz` from the Research Systems Unix Group of the University of Michigan.[20] The SHA-1 hash value of the download is published on the download page, as shown in Figure 1. It is important to check the actual SHA-1 hash value of the file against this published value after the file has been downloaded. This provides some assurance that the download has not been compromised.

**Figure 1: Detail of Radmind Download Page Showing SHA-1 Checksum Value**




Unfortunately, Solaris 10 does not provide a SHA-1 utility. Jacksum[21] was downloaded

and installed in order to check the SHA-1 value of the Radmind download. The output from `jacksum` shown in Figure 2 affirms that actual SHA-1 hash matches the published SHA-1 hash. It is unlikely the Radmind download files have been trojaned, so the installation can proceed.

**Figure 2: Jacksum Output of SHA-1 Checksum Value for `radmind-1.5.0.tgz`**

```
SERVER:/tmp# jacksum -a sha1 radmind-1.5.0.tgz
ee685126ba38afa34649f1d15bc2dfaa5d0e6294 radmind-1.5.0.tgz
```

Figure 3 shows the set of files extracted from the download.

**Figure 3: Files in the Radmind Distribution**

```
SERVER:/usr/products/radmind-1.5.0# ls
COPYRIGHT      base64.h       daemon.c       list.h         pathcmp.h      tls.c
Makefile.in    ca.sh*         fsdiff.c       llist.c        progress.c     tls.h
OS_X/          cksum.c        hardlink.c     llist.h        progress.h     transcript.c
README         cksum.h        install-sh*    lmerge.c       ra.sh*         transcript.h
SPEC           code.c         ktcheck.c      lmerge.h       radstat.c      twhich.c
VERSION        code.h         lapply.c       logname.c      radstat.h      update.c
aclocal.m4     command.c      largefile.h    logname.h      retr.c         update.h
applefile.c    command.h      lcksum.c       lsort.c        rmdirs.c       version.c
applefile.h    config.h.in    lcreate.c      man/           rmdirs.h       wildcard.c
argcargv.c     configure*     lfdiff.c       mkdirs.c       root.c         wildcard.h
argcargv.h     connect.c      libsnet/       mkdirs.h       root.h
base64.c       connect.h      list.c         pathcmp.c      stor.c
```

### 4.1.2 Building Radmind on the Server

The `configure` script is run to prepare to build the executable. There are several relevant options for `configure`, but the only two which will be used here are the following:

`--with-authlevel=2`

This option turns on TLS, which allows the Radmind clients and servers to authenticate each other and to pass messages with encryption. The default is authlevel=0, which does not use TLS. At authlevel=1, the client can verify that the server is authenticated and the connection will be encrypted. Authlevel=2 provides the same protections as authlevel=1 and adds the capability for the server to authenticate the clients.

`--with-server=SERVER`

This option sets a default server. Specifying a default server here allows the operator to avoid naming the server with the `-h` option on the Radmind command line tools.

Figure 4 shows partial output from the three steps to build Radmind: `configure`,

make, and `make install`. The `make` command outputs a few warnings but no errors. The `make install` command issues an inconsequential error due to trying to make a directory that already exists. No changes to the downloaded files were needed to build Radmind on Solaris 10.

**Figure 4: Selected Output During Build of Radmind**

```
SERVER:/usr/products/radmind-1.5.0# ./configure -—with-authlevel=2 \
? --with-server=SERVER
checking for gawk... no
checking for mawk... no
checking for nawk... nawk
checking for gcc... gcc
...
...[snip]...
...

SERVER:/usr/products/radmind-1.5.0# make
cd libsnet; make  CC=gcc
./libtool—quiet—mode=compile \
gcc -c  -Wall -Wmissing-prototypes -I. snet.c
building profiled snet.o
...
...[snip]...
...
gcc  -Wall -Wmissing-prototypes -g -O2 -I./libsnet -I. \
     -D_RADMIND_HOST=\"SERVER\" \
     -D_RADMIND_AUTHLEVEL=2 \
     -D_RADMIND_COMMANDFILE=\"/var/radmind/client/command.K\" \
     -c ./ktcheck.c
./ktcheck.c: In function 'createspecial':
./ktcheck.c:123: warning: int format, pid_t arg (arg 5)
./ktcheck.c: In function 'main':
./ktcheck.c:569: warning: int format, pid_t arg (arg 5)
gcc  -Wall -Wmissing-prototypes -g -O2 -I./libsnet -I.  -c  retr.c
retr.c: In function 'retr':
retr.c:124: warning: int format, pid_t arg (arg 5)
gcc  -Wall -Wmissing-prototypes -g -O2 -I./libsnet -I. \
     -c ./progress.c
...
...[snip]...
...

SERVER:/usr/products/radmind-1.5.0# make install
cd libsnet; make  CC=gcc
mkdir tmp
mkdir tmp/man
...
...[snip]...
...
mkdir tmp
mkdir: Failed to make directory "tmp"; File exists
*** Error code 2 (ignored)
...
...[snip]...
...
```

After the completion of these commands the main Radmind executable has been created in `/usr/local/sbin`, Radmind utilities have been created in `/usr/local/bin`, and man pages for each of these executables have been placed in

/usr/local/man. The security administrator must set the 'path' and 'manpath' environment variables to include these locations. Changes to the filesystem made by installing Radmind are shown in Figure 5.

**Figure 5: Additions to Filesystem Due to Installation of Radmind**

```
SERVER:/var/radmind# ls
cert/       client/     postapply/  preapply/

SERVER:/var/radmind# cd /usr/local/bin
SERVER:/usr/local/bin# ls
fsdiff*   ktcheck*  lcksum*   lfdiff*   ra.sh*
jacksum*  lapply*   lcreate*  lmerge*   twhich*

SERVER:/usr/local/bin# ls -al
total 1566
drwxr-xr-x   2 root     root         512 Mar 11 14:35 ./
drwxr-xr-x   5 root     root         512 Mar 11 14:35 ../
-rwxr-xr-x   1 root     root       62152 Mar 11 14:35 fsdiff*
-rwxr-xr-x   1 root     root         637 Mar 11 11:49 jacksum*
-rwxr-xr-x   1 root     root      138932 Mar 11 14:35 ktcheck*
-rwxr-xr-x   1 root     root      135352 Mar 11 14:35 lapply*
-rwxr-xr-x   1 root     root       69324 Mar 11 14:35 lcksum*
-rwxr-xr-x   1 root     root      127116 Mar 11 14:35 lcreate*
-rwxr-xr-x   1 root     root      117772 Mar 11 14:35 lfdiff*
-rwxr-xr-x   1 root     root       31956 Mar 11 14:35 lmerge*
-rwxr-xr-x   1 root     root        9141 Mar 11 14:35 ra.sh*
-rwxr-xr-x   1 root     root       59132 Mar 11 14:35 twhich*

SERVER:/usr/local/bin# cd /usr/local/sbin
SERVER:/usr/local/sbin# ls -al
total 356
drwxr-xr-x   2 root     root         512 Mar 11 14:35 ./
drwxr-xr-x   5 root     root         512 Mar 11 14:35 ../
-rwxr-xr-x   1 root     root      170084 Mar 11 14:35 radmind*
```

### 4.1.3 Building Radmind on the Clients

Radmind may be built independently on the clients, or the Radmind command line tools may be pushed out to the clients after Radmind is built on the server, if the hosts are of the same operating system. The `radmind` executable itself is not needed on the clients. If the package is built independently on a client, then the client's copy of the `radmind` executable may be safely removed. Removing it improves the security of the client because it cannot be exploited if it is not there.

## 4.2 Setting up Transport Level Security

Before starting up Radmind, because it has been built with TLS authentication level 2 turned on, it is necessary to set up the certification authority on the server and establish

keys for the server and clients to use. If no TLS is to be used then this section should be skipped. The instructions here follow very closely "Using TLS with Radmind," version 0.9.1, which can be found on the Radmind website.[22]

### 4.2.1 Establishing a Certificate Authority

To begin the process of setting up a certificate authority, a sample configuration file, `openssl.cnf`, was downloaded from the Radmind web site.[23] This configuration file was edited to set some defaults to the expected local values. For example, Organization Name was set to "GIAC Enterprises" instead of "University of Michigan," but otherwise, no changes were necessary to allow the certificate authority and keys to be created. The modified configuration file is shown in Appendix A: Sample TLS Configuration File.

**Figure 6: Setting Up a TLS Certificate Authority for Radmind**

```
SERVER:/var/radmind# mkdir CA
SERVER:/var/radmind# mkdir CA/certs
SERVER:/var/radmind# mkdir CA/crl
SERVER:/var/radmind# mkdir CA/newcerts
SERVER:/var/radmind# mkdir CA/private
SERVER:/var/radmind# echo "01" > CA/serial
SERVER:/var/radmind# touch CA/index.txt
SERVER:/var/radmind# cd CA
SERVER:/var/radmind/CA# openssl req -new -x509 -days 400 -keyout \
? private/CAkey.pem -out ca.pem -config openssl.cnf
Generating a 1024 bit RSA private key
..............++++++
.................................++++++
writing new private key to 'private/CAkey.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be
Incorporated into your certificate request.
What you are about to enter is what is called a Distinguished Name or a
DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value, If you enter '.', the
field will be left blank.
-----
Country Name (2 letter code) [US]:
State or Province Name (full name) [Texas]:
Locality Name (eg, city) []:Garland
Organization Name (eg, company) [GIAC Enterprises]:
Organizational Unit Name (eg, section) [Information
Security]:
Common Name (eg, YOUR name) []:GIAC Admin
Email Address []:admin@giacenterprises.com
```

The security administrator must enter a pass phrase, which is not echoed to the screen. It should be long enough to be difficult to guess but something easily remembered.

The result of these steps will be the creation of two new files, the certificate authority's certificate, `ca.pem`, and the associated key file, `CAkey.pem`.

### 4.2.2 Creating Certificates

The next step is to create a certificate. The pass phrase, which is not echoed to the screen, must match the one used above during the creation of the certificate authority. (Otherwise, the openssl command will go into a segmentation fault and dump core.) The Common Name used should be the domain name of the server.

**Figure 7: TLS Certificate Creation**

```
SERVER:/var/radmind/CA# openssl req -new -keyout
key.pem -out req.pem -days 360 -config openssl.cnf -nodes
Generating a 1024 bit RSA private key
..................++++++
.....................................++++++
writing new private key to 'key.pem'
-----
You are about to be asked to enter information that will be
incorporated
into your certificate request.
What you are about to enter is what is called a
Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [US]:
State or Province Name (full name) [Texas]:
Locality Name (eg, city) []:Garland
Organization Name (eg, company) [GIAC Enterprises]:
Organizational Unit Name (eg, section) [Information Security]:
Common Name (eg, YOUR name) []:SERVER.giacenterprises.com
Email Address []:admin@giacenterprises.com

SERVER:/var/radmind/CA# cat req.pem key.pem > new-req.pem
SERVER:/var/radmind/CA# openssl ca -policy
policy_match -out out.pem -config openssl.cnf -infiles new-req.pem
Using configuration from openssl.cnf
3821:error:0E06D06C:configuration file
routines:NCONF_get_string:no
value:/builds/on10_74l3/usr/src/common/openssl/crypto/conf/conf_lib.c:3
29: group=CA_default
name=unique_subject
Enter pass phrase for /var/radmind/CA/private/CAkey.pem:
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName           :PRINTABLE:'US'
stateOrProvinceName   :PRINTABLE:'Texas'
localityName          :PRINTABLE:'Garland'
organizationName      :PRINTABLE:'GIAC Enterprises'
organizationalUnitName:PRINTABLE:'Information Security'
commonName            :PRINTABLE:'SERVER.giacenterprises.com'
emailAddress          :IA5STRING:'admin@giacenterprises.com
Certificate is to be certified until Mar 12 19:20:59 2006 GMT (365
days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated

SERVER:/var/radmind/CA# cat out.pem key.pem > cert.pem
SERVER:/var/radmind/CA# rm req.pem new-req.pem out.pem
```

Following the execution of the above steps, new files `/var/radmind/CA/cert.pem`. `/var/radmind/CA/key.pem`, and `/var/radmind/CA/newcerts/01.pem` will have been created. The file `/var/radmind/CA/index.txt` will have been updated with the Distinguished Name and some additional information.

To make TLS active, the certificate authority's certificate, `ca.pem`, and the server's certificate, `cert.pem`, must be copied to `/var/radmind/cert` on the server. In addition, a copy of the certificate authority's certificate must be placed in `/var/radmind/cert` on each client.

Since `authlevel=2` has been chosen for this implementation of Radmind, client certificates must also be created and propagated to the clients. Certificates for the clients are created following the exact steps as in Figure 7 for the server, except that each client's own domain name is used in the Common Name field. Finally, for each client, a copy of its own certificate must be placed in `/var/radmind/cert` on the client.

### 4.3 Automating Startup on Boot

For Radmind to be an effective service, it must be active whenever the server itself is running. Either of two methods may be used to start the Radmind service at boot time. The first, using `/etc/init.d`, is now obsolete, but it is included for the benefit of those who may be installing Radmind on older Solaris systems.

#### 4.3.1 Automating Startup Using the Service Management Facility

The familiar init.d-style startup script (as shown in the next section) will work on a Solaris 10 server. However, its use is discouraged because the Solaris 10 Service Management Facility cannot handle services started from `/etc/rcS.d`, `/etc/rc2.d`, or `/etc/rc3.d` as effectively as those which are set up properly with `svccfg` and `svcadm`.

To ensure that the Radmind service starts at run-time, a service manifest must first be created in a subdirectory of `/var/svc/manifest/`. In this example, since Radmind is a network service and provides a security service to its clients, the service manifest is created in `/var/svc/manifest/network/security`. The service manifest, which is displayed as Figure 41 in Appendix A, consists of XML tags.

The next steps, as shown in Figure 8, are to import this definition into the services repository, to test the startup and shutdown, and to bring the service online with the service administrator. The `enable -t` subcommand to `svcadm` starts Radmind, but only temporarily. Without the `-t` option, the `enable` subcommand will store the necessary information so that Radmind is automatically started during subsequent reboots.

**Figure 8: Bringing the Radmind Service Online**

```
SERVER:/# svccfg import /var/svc/manifest/network/security
SERVER:/# svcs | grep radmind
SERVER:/# svcadm enable -t radmind
SERVER:/# svcs | grep radmind
online         13:41:31
svc:/network/security/radmind:default

SERVER:/# svcadm disable -t radmind
SERVER:/# svcs | grep radmind
SERVER:/# svcadm enable radmind
SERVER:/# svcs | grep radmind
online         13:41:56
svc:/network/security/radmind:default

SERVER:/# ps -ef | grep radmind
    root  1615     1   0 13:41:57 ?    0:00   /usr/local/sbin/radmind -w
2
```

### 4.3.2 Automating Startup Using `/etc/init.d`

The following material is included for the benefit of a security administrator who may be installing Radmind using a Solaris 8 or 9 host as the server. This method of enabling Radmind will work on Solaris 10 machines, but the methods in the previous section are preferred. In this case, a startup script such as the following must be created for Radmind. The `-w 2` option is used to ensure Radmind is using TLS level 2 security. This script is stored in `/etc/init.d/radmind` on the server.

**Figure 9: Radmind Startup Script**

```
#! /bin/sh
#
# start/stop the radmind server daemon

case "$1" in

'start')
     # Start the radmind server daemon
     if [ -f /usr/local/sbin/radmind ]; then
          echo "starting radmind server daemon"
          /usr/local/sbin/radmind -w 2&
     fi
     ;;

'stop')
     # Stop the radmind server daemon
     PID='/usr/bin/ps -e -u 0 | /usr/bin/fgrep radmind | /usr/bin/awk '{print $1}''
     if [ ! -z "$PID" ] ; then
          /usr/bin/kill ${PID} >/dev/null 2>&1
     fi
     ;;

*)
     echo "usage: /etc/init.d/radmind {start|stop}"
     ;;

esac
```

Figure 10 shows a test of the startup script. Afterwards, in order to have the Radmind server restarted each time the machine is booted up, the startup script must be copied to `/etc/rc3.d`.

**Figure 10: Testing the Radmind Startup Script**

```
SERVER:/etc/init.d# radmind start
starting radmind server daemon

SERVER:/etc/init.d#  ps -ef | grep radmind
    root  3939     1   0 15:02:30 ?     0:00  /usr/local/sbin/radmind -w
2

SERVER:/etc/init.d# radmind stop
SERVER:/etc/init.d# ps -ef | grep radmind
SERVER:/etc/init.d# cp radmind /etc/rc3.d/S99radmind
```

With this script in place in the `/etc/rc3.d` directory, each time the server is rebooted, the Radmind service will be started.

### 4.4 Radmind Operations

The normal sequence of actions to set up the Radmind client/server relationship starts

with the construction the base load on the client. The base load establishes the parts of the client filesystem that will be checked for undesired change. The copy of the base load stored on the server will be used to restore the client to its initial state, if a filesystem object is added, deleted, or modified. The security administrator builds a set of negative and positive transcripts, and then uses them as input to the `fsdiff` utility to construct the base load.

When the security administrator is satisfied that the base load includes the correct set of filesystem objects, he or she then uses the `lcreate` utility to copy the transcripts and files of the base load to the server.

On the client, the Radmind utilities `ktcheck`, `fsdiff`, and `lapply` are used together, either manually, or scripted and called from a cron table entry, to check the client's filesystem and to restore any files that have changed.

### 4.4.1 Using `fsdiff` to Create an Initial Negative Transcript

The first step in constructing the base load is to create an empty command file. Ultimately, the command file will contain a list of transcript files that are used to create, check, and refresh the base load. At this point, however, the transcript files do not yet exist, and the `fsdiff` command, which will be used to build them up, needs the command file to exist.

In the sequence shown in Figure 11, the `fsdiff` command is run on the full filesystem to get a feel for its behavior. The `-%` option tells `fsdiff` to issue occasional progress reports to the console, rather than to spew out every filesystem object touched. That full list of filesystem objects will be written to the output file specified with the `-o` option. Unfortunately, this `fsdiff` run ends unexpectedly.

**Figure 11: Initial Run of `fsdiff`**

```
CLIENT:/var/radmind/client# touch client-cmd.K
CLIENT:/var/radmind/client# fsdiff -C -c sha1 -K client-cmd.K -% \
? -o client-base.T /
%00 /
%02 /.Xauthority
%05 /.cshrc
%08 /.dt
%09 /.dt/sessionlogs
%10 /.dtprofile
...
...[snip]...
...
%72 /opt
%73 /opt/SUNWits
%74 /opt/etc
%75 /platform
%76 /platform/ncri86pc
%78 /proc
/proc/0/ctl: Invalid argument
```

Investigation reveals that certain files in the `/proc` directory have a file permission setting of `0400`. That is, they can be written to by the root user, but not read. Similar problems will occur for the `/system` directory. To avoid hitting these errors, it is necessary to list `/proc` and `/system` in a negative transcript.

A transcript contains a list of filesystem objects. For a negative transcript, Radmind will look at the attributes of each object but will not look into the contents of directories. To generate properly formatted lines for the negative transcript, `fsdiff` can be used with the `-1` option, as shown in Figure 12.

**Figure 12: Generating Lines for a Negative Transcript**

```
CLIENT:/var/radmind/client# fsdiff -1 /proc
d /proc                                  0555     0     0

CLIENT:/var/radmind/client# fsdiff -1 /system
d /system                                  0755     0     0
```

The output of `fsdiff -1` may be copied directly into the negative transcript file. The name of the negative transcript file is then placed in the command file, with an ‘n’ to designate it as a negative transcript. When these actions are taken and `fsdiff` is re-run, it runs all the way through the filesystem with no further errors.

**Figure 13: Sample Negative Transcript and Command Files**

```
CLIENT:/var/radmind/client# cat client-neg.T
# Example of negative transcript
d /proc                                 0555     0     0
d /system                               0755     0     0

CLIENT:/var/radmind/client# cat client-cmd.K
# Type          File
n               client-neg.T

CLIENT:/var/radmind/client# fsdiff -C -c sha1 -K client-cmd.K -% \
? -o client-base.T /
%00 /
%02 /.Xauthority
%05 /.cshrc
%08 /.dt
...
...[snip]...
...
```

The resulting output file, here named `client-base.T`, will contain all filesystem objects except those contained in /proc and /system. Many of these filesystem objects are volatile, such as system logs, or are not important enough to guard with Radmind.

### 4.4.2 Radmind Session with Simplified Sample Base Load

The task of the security administrator at this point is to determine the set of filesystem objects that are to be protected and those which will be ignored by Radmind, and to fine tune the transcript files to achieve the desired protection. In order to understand how to use Radmind effectively, it will be helpful to work through the session illustrated by Figure 14 through Figure 20. This example serves as an illustration of how the positive and negative transcript files work together with the `fsdiff` command. It also constitutes a tutorial into how to create and store the base load on the Radmind server, and then use it to check and restore the client filesystem.

For this sample session a directory `/foo` has been created on the client, which contains some log files which are frequently updated. Radmind should ignore these log files. The `/foo` directory also contains some static files, which ought to be protected by Radmind. There is a subdirectory, `/foo/bar`, which also contains some static files. There is an additional subdirectory `/foo/tmp` that contains a temporary file, which Radmind does not need to check.

The directory contents under the path /foo are listed for future reference in Figure 14. In addition, the contents of some of the files are displayed.

**Figure 14: Contents of the `/foo` Directory**

```
CLIENT:/var/radmind/client# ls -al /foo/*
-rw-r--r--   1 root     root          20 Mar 16 07:52 /foo/file1
-rw-r--r--   1 root     root          20 Mar 16 07:52 /foo/file2
-rw-r--r--   1 root     root          20 Mar 16 07:52 /foo/file3
-rw-r--r--   1 root     root          19 Mar 16 07:52 /foo/log1
-rw-r--r--   1 root     root          19 Mar 16 07:52 /foo/log2
-rw-r--r--   1 root     root          19 Mar 16 07:52 /foo/log3

/foo/bar:
total 10
drwxr-xr-x   2 root     root         512 Mar 16 08:05 ./
drwxr-xr-x   4 root     root         512 Mar 16 07:52 ../
-rw-r--r--   1 root     root          25 Mar 16 07:52 bfile1
-rw-r--r--   1 root     root          25 Mar 16 08:05 bfile2
-rw-r--r--   1 root     root          25 Mar 16 08:05 bfile3

/foo/tmp:
total 6
drwxr-xr-x   2 root     root         512 Mar 16 07:52 ./
drwxr-xr-x   4 root     root         512 Mar 16 07:52 ../
-rw-r--r--   1 root     root          25 Mar 16 07:52 tfile1

CLIENT:/var/radmind/client# cat /foo/file1
This is /foo/file1.

CLIENT:/var/radmind/client# cat /foo/log1
This is /foo/log1.

CLIENT:/var/radmind/client# cat /foo/bar/bfile1
This is /foo/bar/bfile1.
```

#### 4.4.2.1 Creating the Negative Transcript

Since there are files and directories to be ignored by Radmind, there must be a negative transcript. The first line in the negative transcript shown in Figure 15 is an entry for the root directory itself, which is included here only to save time by allowing Radmind to skip processing for the majority of the client's filesystem. For a realistic application of Radmind, it would be unlikely for the security administrator to place the root directory in the negative transcript. The other entry in this negative transcript is the `/foo` directory itself. Since it is explicitly listed in this transcript, Radmind will check for its existence and will check that its permissions haven't changed, but won't check its contents.

**Figure 15: Negative Transcript for the Sample Session**

```
CLIENT:/var/radmind/client# cat foo-neg.T
# Negative Transcript for /foo example
# To save time, don't try to process the whole directory tree
d /                                     0755     0     0
# /foo must exist, but because it contains a frequently changing set
# of log files, it must stay in the negative transcript.
d /foo                                  0755     0     0
```

#### 4.4.2.2 Creating the Positive Transcript

The negative transcript in Figure 15 excludes all files from being checked by Radmind. The files that should be checked must therefore be listed in a positive transcript. The subdirectory `/foo/bar` is listed in the positive transcript shown in Figure 16, which will also ensure that the files in the subdirectory are checked. The files directly under `/foo` must be listed individually. Note that if Radmind is to protect these files by monitoring a checksum or hash value, then the proper value for the file must appear in the positive transcript. The form of the `fsdiff` command used to generate lines for the positive transcript is `fsdiff -1 -c sha1 filename`.

**Figure 16: Positive Transcript for the Sample Session**

```
CLIENT:/var/radmind/client# cat foo-pos.T
# Positive transcript for /foo example
# /foo/bar contains static files which radmind will protect.
d /foo/bar                              0755     0     0
# /foo also contains some static files.  Since /foo itself is in the
# negative transcript, these files must be listed here individually.
f /foo/file1 0644 0 0 1110981147 20 1CTGgMeIxXN9MyN7+y4AdO3I1tA=
f /foo/file2 0644 0 0 1110981147 20 E6GYRjt3+DVdVc7F38Jx68AR8gg=
f /foo/file3 0644 0 0 1110981147 20 NzG/C1ZYFwMqL9ZE77wmmvAh7Do=
```

#### 4.4.2.3 Setting up the Command File

The command file is edited to contain the two transcript files and is listed in Figure 17. If any filesystem object is listed in more than one transcript, the latest transcript encountered takes precedence.

**Figure 17: Command File for the First Example**

```
CLIENT:/var/radmind/client# cat foo-cmd.K
# Type          Transcript
n               foo-neg.T
p               foo-pos.T
```

#### 4.4.2.4 Constructing the Base Load

Now that the negative and positive transcripts are in place, along with the command file, the `fsdiff` command is used to create the base load transcript. The resulting file is then displayed, showing that the three files from `/foo/bar` have been placed in it, along with their attributes, including the SHA-1 checksum.

**Figure 18: Creating the Base Load for the First Example**

```
CLIENT:/var/radmind/client# fsdiff -C -c sha1 -K foo-cmd.K \
? -% -o foo-base.T /
%00 /
%33 /foo/bar/bfile2
%66 /foo/bar/bfile3
%00 /foo/file1
%100

CLIENT:/var/radmind/client# cat foo-base.T
f /foo/bar/bfile1 0644 0 0 1110981147 25 JuJ6+IvWXzpxt3UNR4H9ErgqS7c=
f /foo/bar/bfile2 0644 0 0 1110981910 25 tcME4l36b/jc5cAyxZugkO/KBDI=
f /foo/bar/bfile3 0644 0 0 1110981918 25 2ACxCDkfnxemmHmyWJlOgV2Y8qA=
```

#### 4.4.2.5 Transferring the Base Load to the Server

Before the Radmind tools on the client can communicate with the Radmind server, a configuration file must be created on the server. The configuration file grants permission for the client to initiate Radmind requests and defines the command file the client will get from the server. The command file listed does not exist yet. It will be created in a few steps, when it is needed. If there are additional clients, they are added to this configuration file.

**Figure 19: Setting Up a Configuration File on the Server**

```
SERVER:/var/radmind# cat config
# Client                        command file
# ------                        ------------
CLIENT.giacenterprises.com      client-foo.K
```

The command `lcreate` is used to communicate with the server. The `-w 2` option turns on TLS to secure the communication path by encrypting the TCP/IP traffic. The client and server also authenticate each other using the certificates created earlier. A negative transcript must be signaled to the server with the `-N` flag.

**Figure 20: Transferring Transcripts and Files to the Radmind Server**

```
CLIENT:/var/radmind/client# lcreate -w 2 -h SERVER.giacenterprises.com \
? /var/radmind/client/foo-base.T
/var/radmind/client/foo-base.T: stored
/foo/bar/bfile1: stored
/foo/bar/bfile2: stored
/foo/bar/bfile3: stored

CLIENT:/var/radmind/client# lcreate -w 2 -h SERVER.giacenterprises.com \
? /var/radmind/client/foo-pos.T
/var/radmind/client/foo-pos.T: stored
/foo/file1: stored
/foo/file2: stored
/foo/file3: stored

CLIENT:/var/radmind/client# lcreate -w 2 -N -h \
? SERVER.giacenterprises.com \
? /var/radmind/client/foo-neg.T
/var/radmind/client/foo-neg.T: stored
```

#### 4.4.2.6 Validating the Base Load on the Server

The set of commands shown in Figure 21 are executed on the server rather than the client. The `lcreate` command on the server copied the transcripts and files from the client into a temporary Radmind directory on the server. Before they are moved to a permanent location, they should be validated with the `lcksum` utility. After these tests pass, all files and transcripts are moved up a level, out of the `/tmp` directory.

**Figure 21: Actions on the Radmind Server to Verify and Store the Base Load**

```
SERVER:/var/radmind/tmp/transcript# ls
foo-base.T   foo-neg.T    foo-pos.T
SERVER:/var/radmind/tmp/transcript# lcksum -c sha1 -n foo-base.T
foo-base.T: verified
SERVER:/var/radmind/tmp/transcript# lcksum -c sha1 -n foo-pos.T
foo-pos.T: verified
SERVER:/var/radmind/tmp/transcript# lcksum -c sha1 foo-neg.T
foo-neg.T: verified
SERVER:/var/radmind/tmp/transcript# mv /var/radmind/tmp/transcript/* \
? /var/radmind/transcript/
SERVER:/var/radmind/tmp/transcript# mv /var/radmind/tmp/file/* \
? /var/radmind/file/
```

The only remaining step is to edit the client's command file on the server to match the set of transcripts just brought over.

**Figure 22: Setting Up a Client Command File on the Server**

```
SERVER:/var/radmind/command# cat client-foo.K
# Type          File
p               foo-base.T
p               foo-pos.T
n               foo-neg.T
```

#### 4.4.2.7 Checking and Restoring the Client Filesystem

At this point, Radmind stands ready to check the client's filesystem and to restore it if unexpected changes are detected. It should be pointed out that Radmind provides passive rather than active protection. It will not prevent filesystem changes, or detect them as they happen. Only when Radmind is commanded to inspect the client's filesystem will it detect any changes. The security administrator may enter that command manually, but more commonly, it is automated with a script and an entry in the cron table.

For the purposes of this example, a script has been written which breaks the client filesystem in various ways. This script, `break.sh`, shown in Figure 23, is used to corrupt the contents of the `/foo` directory. It changes the contents of three of the files, changes permissions on three other files, and removes three more. It also removes the entire `/tmp` subdirectory, and finally adds a new subdirectory and two new files. The purpose is to show how Radmind performs in detecting and correcting these changes.

**Figure 23: Script Used to "Break" the `/foo` Directory**

```
#!/bin/sh
echo "This is BAD /foo/file1." > /foo/file1
echo "This is BAD /foo/log1." > /foo/log1
echo "This is BAD /foo/bar/bfile1." > /foo/bar/bfile1
chmod a+w /foo/file2 /foo/log2 /foo/bar/bfile2
rm /foo/file3
rm /foo/log3
rm /foo/bar/bfile3
rm -r /foo/tmp
touch /foo/file4
touch /foo/bar/bfile4
mkdir /foo/bad
touch /foo/bad/badfile
```

The contents of the `/foo` directory and the contents of the modified files following the running of the `break.sh` script are shown in Figure 24. The state of the `/foo` directory and its contents at this point may be compared to the previous state shown in

Figure 14.

**Figure 24: Contents of the `/foo` Directory after Changes**

```
CLIENT:/var/radmind/client# ./break.sh
CLIENT:/var/radmind/client# ls -al /foo/*
-rw-r--r--   1 root     root          24 Mar 16 08:20 /foo/file1
-rw-rw-rw-   1 root     root          20 Mar 16 07:52 /foo/file2
-rw-r--r--   1 root     root           0 Mar 16 08:20 /foo/file4
-rw-r--r--   1 root     root          23 Mar 16 08:20 /foo/log1
-rw-rw-rw-   1 root     root          19 Mar 16 07:52 /foo/log2

/foo/bad:
total 4
drwxr-xr-x   2 root     root         512 Mar 16 08:20 ./
drwxr-xr-x   4 root     root         512 Mar 16 08:20 ../
-rw-r--r--   1 root     root           0 Mar 16 08:20 badfile

/foo/bar:
total 8
drwxr-xr-x   2 root     root         512 Mar 16 08:20 ./
drwxr-xr-x   4 root     root         512 Mar 16 08:20 ../
-rw-r--r--   1 root     root          29 Mar 16 08:20 bfile1
-rw-rw-rw-   1 root     root          25 Mar 16 08:05 bfile2
-rw-r--r--   1 root     root           0 Mar 16 08:20 bfile4

CLIENT:/var/radmind/client# cat /foo/file1
This is BAD /foo/file1.

CLIENT:/var/radmind/client# cat /foo/log1
This is BAD /foo/log1.

CLIENT:/var/radmind/client# cat /foo/bar/bfile1
This is BAD /foo/bar/bfile1.
```

The `ktcheck` utility is used to fetch the command file (see Figure 17) from the Radmind server. Since the `-K` option is not used here to specify a particular file name, the file is stored as the default command file, `command.K`. Next, the `fsdiff` command is used, in a different way from previously, to check for differences between the base load on the Radmind server and the equivalent filesystem objects on the client.

The significant difference in how the `fsdiff` utility is used is the `-A` option. With this option, the output represents edits to the filesystem that are needed to make it match what it should be. A filesystem object preceded by a plus sign should be downloaded from the server because it is missing or its contents have changed. An object preceded by a minus sign is a new object that was not in the base load and should be deleted. An object not preceded by either sign has had its permissions or timestamp changed. Radmind will simply set those back to the original values without downloading the original file from the base load.

When the `fsdiff` command is used in this way, it shows that changes that should be made to restore the filesystem, but it does not make those changes. To restore the filesystem, this use of `fsdiff` must be combined with the `lapply` utility as shown in Figure 26. A security administrator might use Radmind in this way just to generate alerts that something unexpected has happened on the client, and choose not to immediately restore the client. One reason for doing so would be to collect evidence or determine what weakness led to a compromise.

**Figure 25: Checking the State of the `/foo` Directory**

```
CLIENT:/var/radmind/client# ktcheck -w 2 -c sha1 -h \
? SERVER.giacenterprises.com
/var/radmind/client/command.K: updated

CLIENT:/var/radmind/client# fsdiff -A -c sha1 /
foo-base.T:
+ f /foo/bar/bfile1 0644 0 0 1110981147 25 JuJ6+IvWXzpxt3UNR4H9ErgqS7c=
f /foo/bar/bfile2 0644 0 0 1110981910 25 tcME4l36b/jc5cAyxZugkO/KBDI=
+ f /foo/bar/bfile3 0644 0 0 1110981918 25 2ACxCDkfnxemmHmyWJlOgV2Y8qA=
- f /foo/bar/bfile4 0644 0 0 1110982801 0 2jmj7l5rSw0yVb/vlWAYkK/YBwk=
foo-pos.T:
+ f /foo/file1 0644 0 0 1110981147 20 1CTGgMeIxXN9MyN7+y4AdO3I1tA=
f /foo/file2 0644 0 0 1110981147 20 E6GYRjt3+DVdVc7F38Jx68AR8gg=
+ f /foo/file3 0644 0 0 1110981147 20 NzG/C1ZYFwMqL9ZE77wmmvAh7Do=
```

The output of `fsdiff` is piped into the `lapply` command, which performs the actions necessary to restore the client filesystem, including downloading files from the server.

**Figure 26: Restoring the Client Filesystem**

```
CLIENT:/var/radmind/client# fsdiff -A -c sha1 / | lapply -w 2 \
? -h SERVER.giacenterprises.com
/foo/bar/bfile1: created updating mode time
/foo/bar/bfile2: updating mode
/foo/bar/bfile3: created updating mode time
/foo/bar/bfile4: deleted
/foo/file1: created updating mode time
/foo/file2: updating mode
/foo/file3: created updating mode time
```

Figure 27 shows the contents of the `/foo` directory following the `fsdiff/lapply` command shown in Figure 26. The results are perhaps not exactly what were hoped for, but they are what should have been expected with a correct understanding of how Radmind works. On the positive side, the files `/foo/file1` and `/foo/bar/bfile1` have been restored to their original contents. The permissions on

/foo/file2 and `/foo/bar/bfile2` have been reset to the original values. The files `/foo/file3` and `/foo/bar/bfile3`, which had been deleted, have now been replaced. In addition, as expected, the log files have not been restored, nor has the `/foo/tmp` subdirectory.

What may be a little surprising is that the subdirectory and files that were added during the "compromise" were not deleted, with the single exception of `/foo/bar/bfile4`. The reason they were not deleted is because `/foo` was placed in the negative transcript. That placement serves as a signal to Radmind that the contents of `/foo` are not, in general, security-relevant. The only files underneath `/foo` in the directory tree which will be checked are those for which there is an overriding listing in the positive transcript, as is the case for `/foo/bar`.

**Figure 27: Contents of `/foo` Directory Following Restoration**

```
CLIENT:/var/radmind/client# ls -al /foo/*
-rw-r--r--   1 root     root           20 Mar 16 07:52 /foo/file1
-rw-r--r--   1 root     root           20 Mar 16 07:52 /foo/file2
-rw-r--r--   1 root     root           20 Mar 16 07:52 /foo/file3
-rw-r--r--   1 root     root            0 Mar 16 08:20 /foo/file4
-rw-r--r--   1 root     root           23 Mar 16 08:20 /foo/log1
-rw-rw-rw-   1 root     root           19 Mar 16 07:52 /foo/log2

/foo/bad:
total 4
drwxr-xr-x   2 root     root          512 Mar 16 08:20 ./
drwxr-xr-x   4 root     root          512 Mar 16 08:21 ../
-rw-r--r--   1 root     root            0 Mar 16 08:20 badfile

/foo/bar:
total 10
drwxr-xr-x   2 root     root          512 Mar 16 08:21 ./
drwxr-xr-x   4 root     root          512 Mar 16 08:21 ../
-rw-r--r--   1 root     root           25 Mar 16 07:52 bfile1
-rw-r--r--   1 root     root           25 Mar 16 08:05 bfile2
-rw-r--r--   1 root     root           25 Mar 16 08:05 bfile3

CLIENT:/var/radmind/client# cat /foo/file1
This is /foo/file1.
CLIENT:/var/radmind/client# cat /foo/log1
This is BAD /foo/log1.
CLIENT:/var/radmind/client# cat /foo/bar/bfile1
This is /foo/bar/bfile1.
```

The real problem in the session just completed is that the `/foo` directory contains a mixture of volatile files and static files. If some of these static files were executables or important system configuration files, then this directory might become an attractive location for an attacker to stash a malicious file. Better results may be obtained by arranging for the volatile log files to be in their own subdirectory, which allows the

directory containing the static files to be better protected.

The next few figures show the critical parts of a session similar to the session above, but where the log files have been moved from `/foo` to `/foo/logs`. Figure 28 shows that the new negative transcript lists only those subdirectories whose contents change regularly and which are very unlikely to contain executables or important system files. The `/foo` directory itself, which now contains only static files, has been moved from the negative transcript to the positive transcript. It is now not necessary to also list `/foo/bar` in the positive transcript since it is in the directory tree under `/foo`.

**Figure 28: Modified Transcripts and Command File**

```
CLIENT:/var/radmind/client# cat foo1-neg.T
# Negative Transcript for /foo example
d /                                     0755     0     0
d /foo/logs                             0755     0     0
d /foo/tmp                              0755     0     0

CLIENT:/var/radmind/client# cat foo1-pos.T
# Positive transcript for /foo example
d /foo                                  0755     0     0

CLIENT:/var/radmind/client# cat foo1-cmd.K
# Type          Transcript
n               foo1-neg.T
p               foo1-pos.T
```

For this second example, Figure 29 shows the initial contents of the `/foo` directory, as they are at the time that the base load is created.

**Figure 29: Contents of `/foo` Directory**

```
CLIENT:/var/radmind/client# ls -al /foo/*
-rw-r--r--   1 root     root          20 Mar 17 16:42 /foo/file1
-rw-r--r--   1 root     root          20 Mar 17 16:42 /foo/file2
-rw-r--r--   1 root     root          20 Mar 17 16:42 /foo/file3

/foo/bar:
total 10
drwxr-xr-x   2 root     root         512 Mar 17 16:52 ./
drwxr-xr-x   5 root     root         512 Mar 17 16:42 ../
-rw-r--r--   1 root     root          25 Mar 17 16:42 bfile1
-rw-r--r--   1 root     root          25 Mar 17 16:52 bfile2
-rw-r--r--   1 root     root          25 Mar 17 16:52 bfile3

/foo/logs:
total 10
drwxr-xr-x   2 root     root         512 Mar 17 16:42 ./
drwxr-xr-x   5 root     root         512 Mar 17 16:42 ../
-rw-r--r--   1 root     root          19 Mar 17 16:42 log1
-rw-r--r--   1 root     root          19 Mar 17 16:42 log2
-rw-r--r--   1 root     root          19 Mar 17 16:42 log3

/foo/tmp:
total 6
drwxr-xr-x   2 root     root         512 Mar 17 16:42 ./
drwxr-xr-x   5 root     root         512 Mar 17 16:42 ../
-rw-r--r--   1 root     root          25 Mar 17 16:42 tfile1

CLIENT:/var/radmind/client# cat /foo/file1
This is /foo/file1.
```

Figure 30 shows the contents of the `/foo` directory after a “compromise” similar to that in the early example. A new subdirectory has been added this time, `/foo/bad`, with contents `badfile`.

**Figure 30: Contents of `/foo` Directory After Compromise**

```
CLIENT:/var/radmind/client# ls -al /foo/*
-rw-r--r--   1 root     root          24 Mar 17 16:59 /foo/file1
-rw-rw-rw-   1 root     root          20 Mar 17 16:42 /foo/file2
-rw-r--r--   1 root     root           0 Mar 17 16:59 /foo/file4

/foo/bad:
total 4
drwxr-xr-x   2 root     root         512 Mar 17 16:59 ./
drwxr-xr-x   5 root     root         512 Mar 17 16:59 ../
-rw-r--r--   1 root     root           0 Mar 17 16:59 badfile

/foo/bar:
total 8
drwxr-xr-x   2 root     root         512 Mar 17 16:59 ./
drwxr-xr-x   5 root     root         512 Mar 17 16:59 ../
-rw-r--r--   1 root     root          29 Mar 17 16:59 bfile1
-rw-rw-rw-   1 root     root          25 Mar 17 16:52 bfile2
-rw-r--r--   1 root     root           0 Mar 17 16:59 bfile4

/foo/logs:
total 8
drwxr-xr-x   2 root     root         512 Mar 17 16:59 ./
drwxr-xr-x   5 root     root         512 Mar 17 16:59 ../
-rw-r--r--   1 root     root           0 Mar 17 16:59 badlog
-rw-r--r--   1 root     root          28 Mar 17 16:59 log1
-rw-rw-rw-   1 root     root          19 Mar 17 16:42 log2

CLIENT:/var/radmind/client# cat /foo/file1
This is BAD /foo/file1.
CLIENT:/var/radmind/client# cat /foo/bar/bfile1
This is BAD /foo/bar/bfile1.
CLIENT:/var/radmind/client# cat /foo/logs/log1
This is BAD /foo/logs/log1.
```

Figure 31 shows the output of the Radmind tools given the revised transcripts and directory setup. In this example, the outcome is more in line with what might be expected. The `logs` and `tmp` directories are in the negative transcript so their contents are ignored. The new directory `/foo/bad` has been deleted, since `/foo` itself is now in the positive transcript and its contents can be checked.

**Figure 31: Output of `ktcheck`, `fsdiff`, and `lapply`**

```
CLIENT:/var/radmind/client# ktcheck -w 2 -c sha1 -h \
? SERVER.giacenterprises.com
/var/radmind/client/command.K: updated

CLIENT:/var/radmind/client# fsdiff -A -c sha1 /
- d /foo/bad          0755 0 0
- f /foo/bad/badfile 0644 0 0 1111100356 0 2jmj7l5rSw0yVb/vlWAYkK/YBwk=
foo1-base.T:
+ f /foo/bar/bfile1  0644 0 0 1111099334 25 JuJ6+IvWXzpxt3UNR4H9ErgqS7c=
f /foo/bar/bfile2    0644 0 0 1111099920 25 tcME4l36b/jc5cAyxZugkO/KBDI=
+ f /foo/bar/bfile3  0644 0 0 1111099927 25 2ACxCDkfnxemmHmyWJlOgV2Y8qA=
- f /foo/bar/bfile4  0644 0 0 1111100356  0 2jmj7l5rSw0yVb/vlWAYkK/YBwk=
+ f /foo/file1       0644 0 0 1111099334 20 1CTGgMeIxXN9MyN7+y4AdO3I1tA=
f /foo/file2         0644 0 0 1111099334 20 E6GYRjt3+DVdVc7F38Jx68AR8gg=
+ f /foo/file3       0644 0 0 1111099334 20 NzG/C1ZYFwMqL9ZE77wmmvAh7Do=
- f /foo/file4       0644 0 0 1111100356  0 2jmj7l5rSw0yVb/vlWAYkK/YBwk=
foo1-neg.T:
d /foo/tmp           0755 0 0

CLIENT:/var/radmind/client# fsdiff -A -c sha1 / | lapply -w 2 \
? -h SERVER.giacenterprises.com
/foo/bad/badfile: deleted
/foo/bad: deleted
/foo/bar/bfile1: created updating mode time
/foo/bar/bfile2: updating mode
/foo/bar/bfile3: created updating mode time
/foo/bar/bfile4: deleted
/foo/file1: created updating mode time
/foo/file2: updating mode
/foo/file3: created updating mode time
/foo/file4: deleted
/foo/tmp: created updating
```

Finally, just for reference purposes, the contents of the `/foo` directory after the Radmind are shown in Figure 32. This output can be compared to the original directory contents in Figure 29.

**Figure 32: Contents of `/foo` Directory After Restoration**

```
CLIENT:/var/radmind/client# ls -al /foo/*
-rw-r--r--   1 root     root           20 Mar 17 16:42 /foo/file1
-rw-r--r--   1 root     root           20 Mar 17 16:42 /foo/file2
-rw-r--r--   1 root     root           20 Mar 17 16:42 /foo/file3

/foo/bar:
total 10
drwxr-xr-x   2 root     root          512 Mar 17 17:01 ./
drwxr-xr-x   5 root     root          512 Mar 17 17:01 ../
-rw-r--r--   1 root     root           25 Mar 17 16:42 bfile1
-rw-r--r--   1 root     root           25 Mar 17 16:52 bfile2
-rw-r--r--   1 root     root           25 Mar 17 16:52 bfile3

/foo/logs:
total 8
drwxr-xr-x   2 root     root          512 Mar 17 16:59 ./
drwxr-xr-x   5 root     root          512 Mar 17 17:01 ../
-rw-r--r--   1 root     root            0 Mar 17 16:59 badlog
-rw-r--r--   1 root     root           28 Mar 17 16:59 log1
-rw-rw-rw-   1 root     root           19 Mar 17 16:42 log2

/foo/tmp:
total 4
drwxr-xr-x   2 root     root          512 Mar 17 17:01 ./
drwxr-xr-x   5 root     root          512 Mar 17 17:01 ../

CLIENT:/var/radmind/client# cat /foo/file1
This is /foo/file1.

CLIENT:/var/radmind/client# cat /foo/bar/bfile1
This is /foo/bar/bfile1.

CLIENT:/var/radmind/client# cat /foo/logs/log1
This is BAD /foo/logs/log1.
```

### 4.4.3 Overloads

The question arises as to how to deal with intended and approved changes and updates for a client being protected by Radmind. One way to handle these is with an overload. An overload is copied to the server just as the original base load was and it works with the base load to maintain the client in the newly defined state. Multiple overloads can be set up. If they start to get unwieldy, however, there is a tool, lmerge, to merge them and create a new, unified, base load, without starting over from scratch.

When new software is added to the client, it is a good idea first to ensure that the system is clean. The Radmind tools run for this sample session show that no updates are necessary. Then a subdirectory, `/new`, and file, `newfile`, are added to the `/foo` directory, representing the installed software. The tools `fsdiff` and `lcreate` are used to make a new loadset, `new.T`, and to propagate it to the server

as an overload.

**Figure 33: Adding New Software to the Client**

```
CLIENT:/foo/bar# fsdiff -A -c sha1 /
CLIENT:/foo/bar# ktcheck -w 2 -c sha1 -h SERVER.giacenterprises.com
No updates needed

CLIENT:/foo/bar# fsdiff -A -c sha1 / | lapply -w 2 \
? -h SERVER.giacenterprises.com

CLIENT:/foo/bar# cd ..
CLIENT:/foo# mkdir new
CLIENT:/foo# cd new
CLIENT:/foo/new# echo "This is a new file" > newfile
CLIENT:/foo/new# fsdiff -C -c sha1 -o /var/radmind/client/new.T /

CLIENT:/foo/new# more /var/radmind/client/new.T
d /foo/new            0755 0 0
f /foo/new/newfile    0644 0 0 1111263000 19
A48Q+eMSAkUbCTFj6B4G+6wMbzo=

CLIENT:/foo/new# lcreate -w 2 -h SERVER.giacenterprises.com \
? /var/radmind/client/new.T
/var/radmind/client/new.T: stored
/foo/new/newfile: stored
```

On the server, the process is very much the same as it was for the original base load. The security administrator must remember to update the command file for this client. That change is shown at the bottom of Figure 34. The transcript for the overload, `new.T`, is inserted into the command file with the transcripts that already existed as part of the base load.

**Figure 34: Changes on the Server to Accommodate the Overload**

```
SERVER:/var/radmind/tmp# ls -al *
file:
total 6
drwxr-x---   3 root     root         512 Mar 19 14:11 ./
drwxr-x---   4 root     root         512 Mar 12 12:34 ../
drwxr-xr-x   3 root     root         512 Mar 19 14:11 new.T/

transcript:
total 6
drwxr-x---   2 root     root         512 Mar 19 14:11 ./
drwxr-x---   4 root     root         512 Mar 12 12:34 ../
-rw-r--r--   1 root     root         162 Mar 19 14:11 new.T
SERVER:/var/radmind/tmp/transcript# lcksum -c sha1 new.T
new.T: verified
SERVER:/var/radmind/tmp/transcript# mv /var/radmind/tmp/transcript/* \
? /var/radmind/transcript/
SERVER:/var/radmind/tmp/transcript# mv /var/radmind/tmp/file/* \
? /var/radmind/file/

SERVER:/var/radmind/command# cat client-foo1.K
# Type          File
p               foo1-base.T
p               foo1-pos.T
p               new.T
n               foo1-neg.T
```

Figure 35 shows a session on the client where the new directory is inadvertently removed. The `fsdiff` command shows the changes and `lapply` restores the new subdirectory and file. Once the overload has been created on the server, there is no difference on the client to the way checking and restoration was done prior to the overload.

**Figure 35: Checking and Restoring the Client with the Overload**

```
CLIENT:/foo/new# ktcheck -w 2 -c sha1 -h SERVER.giacenterprises.com
/var/radmind/client/command.K: updated

CLIENT:/foo/new# cd ..
CLIENT:/foo# rm -r new
CLIENT:/foo# fsdiff -A -c sha1 /
new.T:
d /foo/new            0755 0 0
+ f /foo/new/newfile  0644 0 0 1111263000 19
A48Q+eMSAkUbCTFj6B4G+6wMbzo=

CLIENT:/foo# fsdiff -A -c sha1 / | lapply -w 2 -h \
? SERVER.giacenterprises.com
/foo/new: created updating
/foo/new/newfile: created updating mode time

CLIENT:/foo# ls -al *
-rw-r--r--   1 root     root          20 Mar 17 16:42 file1
-rw-r--r--   1 root     root          20 Mar 17 16:42 file2
-rw-r--r--   1 root     root          20 Mar 17 16:42 file3

bar:
total 10
drwxr-xr-x   2 root     root         512 Mar 19 14:07 ./
drwxr-xr-x   6 root     root         512 Mar 19 14:16 ../
-rw-r--r--   1 root     root          25 Mar 17 16:42 bfile1
-rw-r--r--   1 root     root          25 Mar 17 16:52 bfile2
-rw-r--r--   1 root     root          25 Mar 17 16:52 bfile3

logs:
total 8
drwxr-xr-x   2 root     root         512 Mar 17 16:59 ./
drwxr-xr-x   6 root     root         512 Mar 19 14:16 ../
-rw-r--r--   1 root     root          28 Mar 17 16:59 log1
-rw-rw-rw-   1 root     root          19 Mar 17 16:42 log2

new:
total 6
drwxr-xr-x   2 root     root         512 Mar 19 14:16 ./
drwxr-xr-x   6 root     root         512 Mar 19 14:16 ../
-rw-r--r--   1 root     root          19 Mar 19 14:10 newfile
```

### 4.5 Using Cron to Automate Filesystem Checking

Figure 36 shows a quick script that could be called from a crontab entry to automate the process of checking and restoring the client's filesystem. The script also produces a time-stamped log file which could be picked up and used for alerts or auditing. The script is stored at `/usr/local/bin/radmind.sh`.

**Figure 36: Shell Script for Use with Cron**

```
#!/bin/sh
#
#  Make a directory for Radmind logs
mkdir -p /var/radmind/logs
#  Get the current date and time and create a time stamped log file
logfile=/var/radmind/logs/radmind_`/usr/bin/date '+%y%m%d-%H%M%S'`.log
#  Run ktcheck
echo "ktcheck: " > $logfile
/usr/local/bin/ktcheck -w 2 -c sha1 -h SERVER.giacenterprises.com >>
$logfile
#  Run fsdiff in output-only mode
echo "fsdiff: " >> $logfile
/usr/local/bin/fsdiff -A -c sha1 -K
#  Run fsdiff again, piping the results to lapply
/var/radmind/client/command.K / >> $logfile
echo "fsdiff | lapply:" >> $logfile
/usr/local/bin/fsdiff -A -c sha1 -K /var/radmind/client/command.K / |
   /usr/local/bin/lapply -w 2 -h SERVER.giacenterprises.com >> $logfile
```

### 4.6 Pitfalls to Avoid

Radmind makes a copy of the client filesystem on the server. By default, this copy is stored within the `/var` partition, although the Radmind directory path can be configured to be something else. Careful planning needs to be done when setting up the server to ensure that sufficient space exists for all of the files that will be copied over.

Transcripts **must** be in alphabetical order, as per the following from the fsdiff man page.

> Transcripts are sorted alphabetically, depth first, and case sensitively. This means subdirectories have precedence over files in the same directory: lexically, "/" has highest precedence. So the file `/etc/passwd` comes before `/etc.old` even though "." normally comes before "/", and `/Library` would come before `/dev` as capitalized characters are higher in precedence than lowercase ones. Both of the previous two directories would come before `/etc.old`.[18]

Generally, Radmind will complain and give an error message that points out the first line it thinks is misplaced, which may or may not be the line that actually needs to be fixed. It can be difficult to get a long transcript into the correct order.

Lines in command (`.K`) and transcript (`.T`) files must be terminated with a newline. A "line too long" error message will be generated otherwise.

When using `fsdiff` to generate lines to be copied to a transcript file, the security administrator should pay attention to relative versus absolute paths. The second call in Figure 37 is probably not what is desired.

**Figure 37: Use of Absolute and Relative Paths in Call to `fsdiff`**

```
CLIENT:/# fsdiff -1 /foo/bar
d /foo/bar          0755    0     0
CLIENT:/# cd /foo
CLIENT:/foo# fsdiff -1 bar
d bar               0755    0     0
```

If a mistake is made and the base load is not delivered completely or correctly to `/var/radmind/tmp/` on the server, the commands on the client cannot simply be redone without first doing some cleanup on the server. The files and templates in `/var/radmind/tmp/file` and `/var/radmind/tmp/template` must be removed first. Cleaning the /tmp directory on failure is listed as a potential improvement for Radmind version 1.6.[16]

## 5 Risks Associated with Using Radmind

The primary strength of Radmind lays not so much in its filesystem integrity checking capabilities, but in its ability to automatically restore a modified filesystem to a previous good state. However, that restoration will destroy most if not all of the evidence as to how the modifications happened in the first place. If there was an attack, nothing will be learned as to how to prevent a reoccurrence or what the attacker may have done while on the system. The security administrator must be prepared to weigh the benefits of the automated system restoration against the loss of potentially valuable forensic data. A way to handle this situation might be to use `fsdiff` to detect changes, and before correcting those changes with `lapply`, using Radmind tools to create a loadset matching the compromised system. That loadset could then be applied to a test machine and inspected.

The Radmind download is protected with a SHA-1 hash value, which is published on the Radmind download site.[20] This provides good confidence that the download is the genuine version. However, a determined attacker could both replace the download with a trojaned version and simultaneously hack the web page to substitute in the new SHA-1 value. Simply publishing a second, different, hash value for each downloadable file would provide a greater level of assurance. Digital signatures of the hash values would also provide greater assurance.

SHA-1 is one of the more widely used checksums[24] and, in fact, is used in most of the examples in the Radmind documentation. For that reason it may be the most likely choice when a security administrator is getting things set up. Recently published reports claim that SHA-1 has been broken.[25] At this time, the risk of actual compromise based on the published work would seem to be very slight.[26] However, as time passes compromise will become easier. A careful security administrator should investigate the other checksum choices available with Radmind.

For any application, even a security application, installed on an information system there is a risk that the application itself will become a means of compromise. Radmind is new enough, and not yet widespread enough throughout the Unix community, that it may not yet have been thoroughly checked out for bugs and vulnerabilities. A search of bugtraq[27] revealed no published vulnerabilities. It is likely that a rise in popularity of Radmind would lead to vulnerabilities being discovered. A careful security administrator would need to pay close attention to this possibility.

A related risk is that an attacker might recognize the Radmind tools and would compromise or replace them with versions that would ignore the attacker's other changes. This could lead to an appearance that nothing is wrong when in fact there is a big problem. A security administrator should have measures in place to occasionally check the filesystem apart from the Radmind tools, or to check the validity of the tools themselves.

The firewall, if any, between the clients and the server must have port 6662 open for TCP/IP sessions initiated by the clients. This might provide an entry point to the server

if an attacker compromises one of the clients. The communications channel could be secured with SSH or a VPN tunnel to diminish the probability of this event.

Radmind will delete files that it thinks do not properly belong to the client filesystem. This action may wipe out newly installed software or other desired changes if the system administrator forgot to update the load files on the server to reflect the update. Written procedures need to be in place to ensure that a Radmind overload is always created and written to the server when any changes are made to a client machine.

Special care needs to be taken to inspect directories that are in the negative transcript. Radmind does not look into these directories and will not care if (possibly malicious) files have been added to them.

The timestamp on directories, even if they are in the positive transcript, can be changed without Radmind caring. Figure 38 shows a session where a file and a directory both have their timestamps altered without any other change to their contents. The `fsdiff utility` flags the change to the file but ignores the change to the directory. Naturally, `lapply` also restores the files but fails to restore the directory.

**Figure 38: An Undetected Timestamp Change**

```
CLIENT:/foo# touch -t 01010101 file1
CLIENT:/foo# touch -t 01010101 bar
CLIENT:/foo# ls -al
total 16
drwxr-xr-x   5 root     root         512 Mar 19 14:00 ./
drwxr-xr-x  35 root     root        1024 Mar 17 16:43 ../
drwxr-xr-x   2 root     root         512 Jan  1 01:01 bar/
-rw-r--r--   1 root     root          20 Jan  1 01:01 file1
-rw-r--r--   1 root     root          20 Mar 17 16:42 file2
-rw-r--r--   1 root     root          20 Mar 17 16:42 file3
drwxr-xr-x   2 root     root         512 Mar 17 16:59 logs/
drwxr-xr-x   2 root     root         512 Mar 17 17:01 tmp/

CLIENT:/foo# fsdiff -A -c sha1 /
f /foo/file1      0644 0 0 1111099334 20 1CTGgMeIxXN9MyN7+y4AdO3I1tA=

CLIENT:/foo# fsdiff -A -c sha1 / | lapply -w 2 -h \
? SERVER.giacenterprises.com
/foo/file1: updating time
```

An implication of the problem shown in Figure 38 is that an attacker who became aware of the Radmind update times could place files temporarily, move or delete them before the next update, and thus escape detection. Figure 39 shows a session in which a file is placed in a directory and then later removed. Another file that belongs in the directory is temporarily removed and then replaced. Even though the directory is in the positive transcript, `fsdiff` is silent and does not recognize that any changes have taken place.

**Figure 39: A File Placed and Removed Prior to using `fsdiff`**

```
CLIENT:/foo/bar# ls
bfile1   bfile2   bfile3

CLIENT:/foo/bar# mv bfile1 /tmp
CLIENT:/foo/bar# touch badfile
CLIENT:/foo/bar# ls -al
total 8
drwxr-xr-x   2 root     root         512 Mar 19 14:07 ./
drwxr-xr-x   5 root     root         512 Mar 19 14:00 ../
-rw-r--r--   1 root     root           0 Mar 19 14:07 badfile
-rw-r--r--   1 root     root          25 Mar 17 16:52 bfile2
-rw-r--r--   1 root     root          25 Mar 17 16:52 bfile3

CLIENT:/foo/bar# mv /tmp/bfile1 .
CLIENT:/foo/bar# ls
badfile   bfile1    bfile2    bfile3

CLIENT:/foo/bar# rm badfile
CLIENT:/foo/bar# ls -al
total 10
drwxr-xr-x   2 root     root         512 Mar 19 14:07 ./
drwxr-xr-x   5 root     root         512 Mar 19 14:00 ../
-rw-r--r--   1 root     root          25 Mar 17 16:42 bfile1
-rw-r--r--   1 root     root          25 Mar 17 16:52 bfile2
-rw-r--r--   1 root     root          25 Mar 17 16:52 bfile3

CLIENT:/foo/bar# fsdiff -A -c sha1 /
CLIENT:/foo/bar#
```

## 6 Comparisons to Tripwire, Aide, and Samhain

As a tripwire-style security tool, Radmind is somewhat less flexible and powerful than Tripwire, AIDE, or Samhain. All three of these tools will check not only the 'mtime,' i.e., the last time that the contents of the file were modified but will check 'ctime' and 'atime' as well. Radmind only checks 'mtime.' 'ctime' is the last time at which a change was made to an object's inode. 'atime' is the last time at which the file was accessed. The lack of these checks could cause Radmind to miss a change that might signify some illicit activity.

Tripwire and AIDE both allow multiple checksums to be computed and checked for a given file. While it is difficult - but possible - for an attacker to produce a malicious file that matches another file's checksum; to attempt to do so for two different checksums at once would generally be a very poor use of time. Therefore, the deterrent effect of the multiple checksums is quite high.

A much more significant strength for Tripwire, AIDE, and Samhain as compared to Radmind is that all three permit the security administrator much more granularity in controlling what objects to include or exclude, and in what attributes to check for those objects. With Radmind, an object is in either a negative transcript or a positive transcript. (If a file is not explicitly listed or is not in the directory tree of a directory that is explicitly listed, then it is implicitly in the positive transcript.) Objects in a negative transcript are checked in just one way; objects in a positive transcript are checked in just one (different) way.

As a particular example, all three of the alternative programs have a setting that allows log files to be watched. An alert is issued if the log file suddenly shrinks in size, which might happen if an attacker removes the lines that show his or her activities. It is difficult to see how this kind of alerting could be done with Radmind. In addition, it would not be desirable for Radmind to replace a modified log file with an older version of itself.

For those who want to explore the alternatives more thoroughly, an online manual for AIDE is found at http://www.cs.tut.fi/~rammer/aide/manual.html,[28] and for Samhain at http://la-samhna.de/samhain/manual/.[29] There is an online manual for Tripwire at http://www.cosmic-ray.org/miscfiles/idsl_1_3.pdf,[30] however, it is for version 1.3 for Linux. Rainer Wichmann has produced a comparison of several integrity scanners (not including Radmind) at http://la-samhna.de/library/scanners.html.[31]

## 7 Conclusions

If a COTS product is not practical, and where using an open-source product of non-US origin is disallowed or discouraged, Radmind should be considered as an adequate alternative to provide filesystem integrity checking. Where those conditions do not apply, it would be difficult to recommend Radmind over the other programs discussed in this report. However, Radmind is still under development, and as time passes, it may well reach the point where it can compete on a level playing field with those other products. Its main strength lies in its capability to automatically restore a modified filesystem to a known good state as soon as the modification is detected. For some information systems with high availability requirements, this automation may be very valuable and help keep the downtime of a system to a minimum.

## Appendix A: Sample TLS Configuration File

Figure 40 shows the file used to configure TSL for this report. This file was downloaded from the Radmind website[23] and was tailored (slightly) for the current test setup.

**Figure 40: The TLS Configuration File: `openssl.cnf`**

```
#
# version 0.9.1
#

# This definition stops the following lines choking if HOME isn't
# defined.
HOME                    = .
RANDFILE        = $ENV::HOME/.rnd

# Extra OBJECT IDENTIFIER info:
#oid_file               = $ENV::HOME/.oid
oid_section             = new_oids

# To use this configuration file with the "-extfile" option of the
# "openssl x509" utility, name here the section containing the
# X.509v3 extensions to use:
# extensions            =
# (Alternatively, use a configuration file that has only
# X.509v3 extensions in its main [= default] section.)

[ new_oids ]

# We can add new OIDs in here for use by 'ca' and 'req'.
# Add a simple OID like this:
# testoid1=1.2.3.4
# Or use config file substitution like this:
# testoid2=${testoid1}.5.6

####################################################################
[ ca ]
default_ca      = CA_default            # The default ca section

####################################################################
[ CA_default ]

dir             = /var/radmind/CA # Where everything is kept
certs           = $dir/certs      # Where the issued certs are kept
crl_dir         = $dir/crl        # Where the issued crl are kept
database        = $dir/index.txt # database index file.
new_certs_dir   = $dir/newcerts   # default place for new certs.

certificate     = $dir/ca.pem     # The CA certificate
serial          = $dir/serial     # The current serial number
crl             = $dir/crl.pem    # The current CRL
private_key     = $dir/private/CAkey.pem     # The private key
RANDFILE        = $dir/private/.rand   # private random number file

x509_extensions     = usr_cert  # The extentions to add to the cert

# Extensions to add to a CRL.
# Note: Netscape communicator chokes on V2 CRLs
# so this is commented out by default to leave a V1 CRL.
# crl_extensions    = crl_ext

default_days    = 365             # how long to certify for
default_crl_days= 30              # how long before next CRL
default_md      = md5             # which md to use.
preserve = no                     # keep passed DN ordering

# A few difference way of specifying how similar the request should look
# For type CA, the listed attributes must be the same, and the optional
# and supplied fields are just that :-)
policy          = policy_match
```

## Appendix B: Sample Radmind Service Manifest

The following is a sample service manifest created by the author, based on examples of other service manifests. It enables Radmind on a Solaris 10 host. It is not necessarily a complete, well-constructed manifest, but it should serve as a good starting point.

**Figure 41: A Radmind Service Manifest: `radmind.xml`**

```
<?xml version='1.0'?>
<!DOCTYPE service_bundle SYSTEM
'/usr/share/lib/xml/dtd/service_bundle.dtd.1'>
<!--
    ident "@(#)radmind.xml      1.0   05/03/05 SMI"

    Service manifest for the radmind server daemon
-->

<service_bundle type='manifest' name='radmind'>

<service
    name='network/security/radmind'
    type='service'
    version='1'>

    <create_default_instance enabled='false' />

        <single_instance/>

        <dependency
            name='network'
            grouping='require_all'
            restart_on='none'
            type='service'>
            <service_fmri value='svc:/network/initial' />
        </dependency>

        <dependent
            name='radmind_multi-user-server'
            grouping='optional_all'
            restart_on='none'>
            <service_fmri value='svc:/milestone/multi-user-server' />
        </dependent>

    <exec_method
        type='method'
        name='start'
        exec='/usr/local/sbin/radmind -w 2'
        timeout_seconds='30'>
    </exec_method>

    <exec_method
        type='method'
        name='stop'
        exec=':kill'
        timeout_seconds='60'>
    </exec_method>

    <stability value='Standard' />

    <template>
        <common_name>
            <loctext xml:lang='C'>
                Radmind, a remote administration server
            </loctext>
        </common_name>
        <documentation>
            <manpage title='radmind' section='8'
            manpath='/usr/local/man' />
        </documentation>
    </template>
</service>
```

## References


[1] United States, Director of Central Intelligence Directive 6/3. *Protecting Sensitive Compartmented Information Within Information Systems – Manual (Industry Annex)*. 1 August 2000 (Updated 20 April 2004).

[2] Cuff, Andy, Computer Network Defence LTD. *File Integrity Checkers*. 2004. http://www.networkintrusion.co.uk/integrity.htm. (19 March 2005).

[3] Fyodor. *Top 75 Network Security Tools*. May 2003. http://www.insecure.org/tools.html. (19 March 2005).

[4] Tripwire, Inc. *Change Auditing Solutions*. 2005. http://www.tripwire.com/. (19 March 2005).

[5] *Tripwire Open Source Project*. http://www.tripwire.org/index.html. (19 March 2005).

[6] Lehti, Rami. *AIDE – Advanced Intrusion Detection Environment*. http://www.cs.tut.fi/~rammer/aide.html. (19 March 2005).

[7] SANS Institute. *Track 6 – Securing Unix/Linux: Unix/Linux Security Tools*. Volume 6.2. Sans Press, 2004.

[8] Samhain Labs. *The Samhain File Integrity/Intrusion Detection System*. http://la-samhna.de/samhain/. (19 March 2005).

[9] Holmes, Winston. *Security Aspects of a Samhain Client/Server Installation to Protect a Solaris Web Server*. 19 September 2004. http://www.giac.org/certified_professionals/practicals/gcux/0261.php. (19 March 2005).

[10] Chouanard, Jean. *YASSP – Yet Another Solaris Security Package*. 19 November 2000. http://www.yassp.org/. (19 March 2005).

[11] Lehti, Rami. *Rami's Homepage*. 24 August 1998. http://www.cs.tut.fi/~rammer/. (19 March 2005).

[12] Samhain Labs. *About Us*. http://la-samhna.de/about.html. (19 March 2005).

[13] Red Hat, Inc. *Taroon-List Archives*. March 2004. https://www.redhat.com/archives/taroon-list/2004-March/msg00293.html. (19 March 2005).

[14] Triangle Linux Users Group. *Archived Weblog Entry*. 10 August 2003. http://www.trilug.org/~amr/bx/blosxom.cgi/2003/08/10. (19 March 2005).

[15] Freebsd.org. *Freebsd-stable Archives*. October 2004. http://lists.freebsd.org/pipermail/freebsd-stable/2004-October/008940.html. (19 March 2005).

[16] Research Systems Unix Group, University of Michigan. *Radmind Roadmap*. 18 March 2005. http://rsug.itd.umich.edu/software/radmind/roadmap.html. (20 March 2005).

[17] University of Utah. *Radmind Tips*. 22 November 2004. http://www.macos.utah.edu/Documentation/radmind/intro.html. (20 March 2005).

[18] Research Systems Unix Group, University of Michigan. *fsdiff man page*. http://rsug.itd.umich.edu/software/radmind/man/fsdiff.1.html. (20 March 2005).

[19] Student Computing Labs, University of Utah. *Radmind Mac OS X File System Management Case Study*. http://www.macos.utah.edu/OSX_OnCampus/UofU_Radmind_Pres.pdf. (20 March 2005).

[20] Research Systems Unix Group, University of Michigan. *Radmind*. 15 February 2005. http://rsug.itd.umich.edu/software/radmind/download.html. (20 March 2005).

[21] Löfflmann, Johann N. *Free Java-Software - Jacksum, a java checksum utility*. 5 February 2005. http://www.jonelo.de/java/jacksum/. (20 March 2005).

[22] Research Systems Unix Group, University of Michigan. *Using TLS with Radmind.* http://rsug.itd.umich.edu/software/radmind/files/radmind-tls-0.9.1.pdf. (20 March 2005).

[23] Research Systems Unix Group, University of Michigan. *Index of /software/radmind/files*. 1 March 2005. http://rsug.itd.umich.edu/software/radmind/files/. (20 March 2005).

[24] Wikipedia. *SHA Hash Functions*. 18 March 2005. http://en.wikipedia.org/wiki/SHA-1. (20 March 2005).

[25] Schneier, Bruce. *Schneier on Security*. 18 February 2005. http://www.schneier.com/blog/archives/2005/02/cryptanalysis_o.html. (20 March 2005).

[26] Zajkowski, Jim, Radmind Mailing List. *Time to start changing the examples? Maybe not.* 17 February 2005. https://mailman.rice.edu/pipermail/radmind/2005-February/009001.html. (20 March 2005).

[27] SecurityFocus. *Bugtraq Mailing List.* 19 March 2005. http://www.securityfocus.com/archive/1. (20 March 2005).

[28] Lehti, Rami. *The AIDE Manual*. http://www.cs.tut.fi/~rammer/aide/manual.html. (20 March 2005).

[29] Wichmann, Rainer. *Samhain*. 2004. http://la-samhna.de/samhain/manual/. (20 March 2005).

[30] Tripwire Security Systems, Inc. *Tripwire Intrusion Detection System 1.3 for LINUX User Manual*. 27 July 1998. http://www.cosmic-ray.org/miscfiles/idsl_1_3.pdf. (20 March 2005).

[31] Wichmann, Rainer. *A comparison of several host/file integrity checkers (scanners)*. 30 November 2004. http://la-samhna.de/library/scanners.html. (20 March 2005).