The RIACS Intelligent Auditing and Categorizing System

Matt Bishop

August, 1988

Research Institute for Advanced Computer Science
NASA Ames Research Center

RIACS Memorandum 88.2

NASA Cooperative Agreement Number NCC 2-398

RIACS

Research Institute for Advanced Computer Science

The RIACS Intelligent Auditing and Categorizing System

Matt Bishop

Research Institute for Advanced Computer Science
NASA Ames Research Center
Moffett Field, CA 94035
and
Department of Mathematics and Computer Science
Dartmouth College
Hanover, NH 03755

ABSTRACT

This guide describes the organization of the RIACS auditing package, how to install it, and how to interpret the output.

1. Introduction

This document contains installation and configuration instructions for the RIACS Intelligent Auditing and Categorizing System. It describes how to set up both local and remote file system auditing. Logging is done on a time-driven basis, and auditing in a passive mode.

The audit package generates a listing of the file system being audited compares it with a previously-generated listing; the differences reveal what has changed and these changes are reported.

In what follows, file names in **boldface** are real file names; file names in *italics* should be replaced by the relevant file names on your system. Sometimes shell variables are relevant; these are indicated by **boldface**. Variables defined in the relevant makefile use the syntax of a makefile variable reference; for example, ${*makefile_variable*}. Finally, specific host names are in **boldface** and a name that is to be replaced by a host name will be in *italics*.

2. Configuring and Compiling the Package

This package can be compiled on either Berkeley UNIX† or System V UNIX machines with no changes. Other versions of UNIX may require some changes.

This work was supported by grant NCC2-398 from the National Aeronautics and Space Administration (NASA) to the Research Institute for Advanced Computer Science (RIACS); portions of the work were done at the author's current institution, Dartmouth College.

† UNIX is a trademark of Bell Laboratories.

1. Determine whether your system is closer to System V or Berkeley UNIX, and copy the appropriate file (**Make.bsd4** or **Make.sysv**) to **Makefile**.
2. Edit **Makefile**. The parameters which may have to be reset are described in the next section.
3. Switch to the superuser and compile and install the sources:

   make install

4. For each host which is to be audited remotely, run the program *rinstall*. This takes as arguments a remote host name, a type (either "bsd4" or "sysv", depending on which the system most closely resembles), and the name of a directory on the remote host where the executable files will be put. (The full path name of this directory will be placed in the appropriate **Environ** file; see the next two steps and Section 4.) This copies several files to the named remote host and installs them. Unless this is done, the remote host cannot be audited from the local one.
5. Create the directory for the data files:

   mkdir ${*LISTDIR*}

6. For each host (including the local one) that will be audited, create a directory for the data files specific to that host:

   mkdir ${*LISTDIR*}*/hostname*

7. Configure the data files. Section 4 describes how to do this.
8. Set up a root list for each host; this list names the roots of the file trees to be audited. Section 4 describes how to do this.
9. Generate and inspect master files for each host. This is important, because if no master files are present and you instruct the audit programs to update the master file whenever it runs, you will **not** see the first update. So, you must check that the master files reflect the desired state of the system. You ought to go to dinner while this part is being done; on any decent-sized system, it takes at least half an hour.
10. Set up the auditing commands to run every so often. Section 5 describes the best way to do this.
11. Go home and get some sleep.

3. Makefile

The **Makefile** contains several variables that can be changed to provide the type of auditing desired. This section summarizes those variables.

3.1. ROOTDIR

This is the root of the audit file tree. All directories and programs related to the audit (that is, executables and data files) should go under this directory.

3.2. AUDITOR

This variable should be set to a list of addresses of people who are to receive the audit report and any error messages from the audit. If more than one address is to be specified, separate the addresses by commas. The addresses may be on the local machine or remote. The setting of this option may be overridden when the audit is run.

3.3. DESTBIN

This is the directory where the executable files are to be placed on the local machine. Both shell scripts and executable object files go there. Be aware that the two programs **auditls** and **lstat** are assumed to reside in this directory on the remote host, unless another path is specified. (See the **Environ** file, below.)

3.4. DESTMAN

This is the root of the directory tree where manual pages are put. It assumes there are subdirectories **man1** and **man8**.

3.5. DESTSRC

If the auditing system is to audit more than one machine, the machine on which the auditing is done is called the *server* and the machines being audited are the *clients* (yes, one machine can be the server and a client at the same time!) Client machines must have two programs installed. This is the directory containing the sources that are sent to the client and the makefiles used to compile them.

3.6. FILESYS

This is the root of the file tree that will be audited if no other file tree is specified. There are numerous ways to specify the root of the file tree to be audited at run time, so pick something that is quick.

3.7. FIND

This is the *find*(1) command and is usually in either **/bin** or **/usr/bin**. Set this to the path name of the executable for your system.

3.8. LIMIT

This is the number of days for which **Reject** files are valid.

3.9. LISTDIR

This variable specifies the root of a set of directories which hold the data files.

3.10. MAILER

This is the program that is used to mail error messages and the results of an audit to the appropriate people. It is invoked by giving the list of addresses, separated by blanks, as command-line arguments, and the letter as standard input. If possible, a "Subject: " field may be specified using an "–s" option; see **SUBJECT**, below.

3.11. REMCOPY

This is the program used to copy files to a client. It is called with two arguments: the first is the name of the file to be copied, and the second the name of the remote host followed by a colon followed by the name the file is to be copied to.

3.12. REMEXEC

This is the program that is used to execute a command remotely. It is called with several arguments: the first is a host name, the second a "–n" flag to indicate that there is no input, and the third and successive arguments are a standard UNIX command with its arguments. This program invokes a file system logging program on a remote system when a remote audit is being done, and to perform remote compilation when two programs are being installed on clients.

3.13. SUBJECT

The legal settings of this field are "yes" and "no". If the mailer allows specification of a "Subject: " header field by giving a command-line option of the form "–s "*subject*"", set this to "yes". Then the first and second command-line arguments will be set to "–s "*subject*"", and the addresses will be the third (and successive) arguments.

3.14. SHELL

On some versions of UNIX, the specific shell *make*(1) uses to run its commands is determined by the setting of this variable. It should be set to the Bourne shell *sh*(1).

3.15. TEST

The shell command *test*(1) is either a built-in command or a binary, usually in **/bin**. Define this as **test** if it is a built-in shell command, or to the path name of the executable if it is not.

3.16. THRESH

Under certain circumstances (notably if network software is unreliable), the auditing program will scan only part of a file system before the scan ends. The result is that a large number of files in that file system will be reported as deleted. Since the master file may be changed, this also results in a bogus message the next time the scan does succeed, since all those files will show as having been added. Use this variable to control this problem; when ${*THRESH*} is set to a positive value, the master file will never be replaced unless the new master file would be greater than ${*THRESH*} percent of the old. For example, if the old master file showed 100 files in **/bin**, and the scan of **/bin** showed 12, the results of the scan would be saved, an error message describing the problem and what to do would be generated, and the master file would remain in place. A reasonable value to set this to is 10 (for 10%.)

3.17. TIMEOUT

When the command to log information about the file system is run, it is possible that the command may hang. (This is especially true during a remote file system audit; problems with the communications protocols or medium may cause this to happen.) Every ${*TIMEOUT*} seconds, the file system listing program will be checked to determine if the process has hung. If so, it will terminate the process. If you do not wish this to be done, set the value of this variable to 0; this disables the timeout feature.

3.18. TMPDIR

Temporary files generated during the audit go in this directory. The directory may be protected; the audit package must be able to create and delete directories in it, but that is all. It is *strongly* recommended that neither **/tmp** nor **/usr/tmp** be used, because that enables people to delete the temporary files during processing and can corrupt the results of the audit.

3.19. EQN, PIC, ROFF, and TBL

These are the commands used to print the paper "Security Monitoring" and the installation guide. ${*EQN*} should be *eqn*(1) or its equivalent, ${*PIC*} should be *pic*(1) or its equivalent, ${*TBL*} should be *tbl*(1) or its equivalent, and ${*TROFF*} should be *troff*(1) or its equivalent.

3.20. Defaults

The following table summarizes the default settings:

Makefile variables		
variable	*default*	*what it means*
ROOTDIR	/usr/local/adm/audit	root for audit system
AUDITOR	root	where to send audit results
DESTBIN	${*ROOTDIR*}/=bin	where to put executables
DESTMAN	(*4BSD*) /usr/local/man (*SYSV*) /usr/local/man/a_man	where to put manual pages
DESTSRC	${*ROOTDIR*}/=src	where to put sources for clients
FILESYS	/bin	file system to be audited
FIND	(*4BSD*) /usr/bin/find (*SYSV*) /bin/find	program to scan file tree
LIMIT	1	days to leave unmodified reject files
LISTDIR	${*ROOTDIR*}	where data files are kept
MAILER	(*4BSD*) Mail (*SYSV*) mailx	program to mail results
REMCOPY	rcp	remote copy command
REMEXEC	(*4BSD*) rsh (*SYSV*) remsh	remote execution command
SHELL	/bin/sh	shell used by *make*(1)
SUBJECT	yes	does MAILER know ''–s''
TEST	(*4BSD*) /bin/test (*SYSV*) test	*test*(1) program
THRESH	10	audit suspicion threshhold
TMPDIR	${*ROOTDIR*}/=tmp	where temporary files go

See the individual entries above for more details

4. Files

There are several type of files relevant to the audit system. This section describes each one, especially those that the user must set up.

4.1. ${*LISTDIR*}

This directory contains the subdirectories that the audit package will scan during the course of its audit. The directories may be named anything convenient; a good convention is to name them after the host to be audited. For example, if the hosts **icarus**, **hydra**, and **miranda** would be audited by this system, ${*LISTDIR*} would contain three subdirectories:

* ${*LISTDIR*}/**icarus** contains the data files for **icarus**;
* ${*LISTDIR*}/**hydra** contains the data files for **hydra**; and
* ${*LISTDIR*}/**miranda** contains the data files for **miranda**.

4.2. Equiv

This file is used to map various host names to directory names. It consists of lines of two fields separated by blanks or tabs, the first of which must begin in the first column. The first field contains a version of the host name; the second, the name of the

subdirectory of ${*LISTDIR*} that contains the associated data files. Anything following the second field is treated like a comment and is ignored. Here is a sample file:

This file contains the mappings from host names to directories.

host	directory	any comments
icarus.riacs.edu	icarus	full domain names
hydra.riacs.edu	hydra	
miranda.riacs.edu	miranda	
riacs-icarus.arpa	icarus	old-style arpanet name

Requesting any of the systems **icarus.riacs.edu**, **icarus**, and **riacs-icarus.arpa** will use the files in ${*LISTDIR*}/**icarus**. However, the remote communications protocol will access the remote host by the given name, not the name in field 2; so if the last form were used, the remote connection would be made to **riacs-icarus.arpa** (which ought to be the same as **icarus**!)

No host name in the second column may contain a comma. That character indicates the data files for the host named in the first column are distributed among several directories. This format will be described at the end of this section.

Note that lines which do not begin in column 1 are ignored and so may be used for comments. Hence the first two lines of the above file affect nothing.

4.3. Environ

Each host directory should contain one of these. Lists of files on remote hosts are generated by a script on the remote host; as a result, certain programs on the remote host must be executed. This file contains the path name of the programs involved. The defaults are given below. The programs involved are:

* **auditls** is the program that generates the listing. It is a shell script located in the audit package source directory.
* **egrep** is the extended pattern-matching version of *grep*(1).
* **find** is the recursive file system searching utility *find*(1).
* **ls** is the standard program *ls*(1) that lists the files in a directory.
* **lstat** is the program that lists the files in an auditable form. The source to this program is in the audit package directory.
* **mount** is the program that lists the names and types of the volumes that file systems are mounted on; it is usually *mount*(1) or *mount*(8) (depending on the distributor).
* test is the shell's condition command *test*(1). If it is built into the shell (as for System V), do not specify a path name; otherwise, do.

The **Environ** file consists of lines of two fields separated by blanks or tabs, the first of which must begin in the first column. The first field contains the name of the command; the second, the full path name of the remote command. Anything following the second field is treated like a comment and is ignored. Here is a sample file:

```
This file contains the path names of some remote programs
command          full path name              any comments
lstat            /usr/adm/audit/bin/lstat    file stat function
auditls          /usr/adm/audit/bin/auditls  logging function
```

This file indicates that when the auditing package issues the command to scan the remote file system, it will use the command **/usr/adm/audit/bin/auditls**. When that command calls the file lister, it will use **/usr/adm/audit/bin/lstat** to do so. The other commands all use defaults; they are:

Program Defaults	
command	*path name*
auditls	${*DESTBIN*}/auditls
egrep	/usr/bin/egrep
find	(*4BSD*) /usr/bin/find (*SYSV*) /bin/find
ls	/bin/ls
lstat	${*DESTBIN*}/lstat
mount	/etc/mount
test	(*4BSD*) /bin/test (*SYSV*) test

Note that lines which do not begin in column 1 are ignored and so may be used for comments. Hence the first two lines of the above file affect nothing.

IMPORTANT: If a client is running a BSD-based UNIX and the server is running a System V-based UNIX, specify both *find* and *test* in the Environ file, or the remote audit will not work. The same holds if a client is running a System V-based UNIX and the server is running a BSD-based UNIX.

4.4. F*

These files contain the master lists. They may be generated automatically using the audit system, and this should be done, since the auditing mechanism is very sensitive to the format of these files. The files are named by deleting all "/"es from the name of the root directory of the file tree, prefixing a "F", and (on some systems) truncating the name to 14 characters (this is a result of system limits; the program will not do this.) If only setuid files are being examined, the prefix "FU", rather than "F", is used; if only setgid files are being examined, the prefix "FG", rather than "F", is used; and if both setuid and setgid files are being examined, the prefix "FB", rather than "F", is used.

4.5. I*

These files contain the ignore lists. When an audit is run, there are certain files that are expected to change, and the auditors do not want these changes to be reported. (An example is the file **/etc/utmp**, which is changed whenever a user logs in or logs out.) These files provide a mechanism to eliminate such files from the audit.

The name of each ignore file is the same as the master file associated with the root directory, except that the leading "F" is replaced by an "I". Each line of the ignore file

contains a pattern suitable for passing to *egrep*(1) that will match the files which are to be ignored. The field ends with a newline or tab. Anything on the line following that field is ignored; it is very strongly recommended that you put a comment there, explaining why that file is being ignored. Here is a sample ignore file for the root directory; it is named **I**:

```
\*      This lists files in the root directory not to be audited.
\*      All these suckers are directories; they will be audited
\*      separately. This is the file for hydra.riacs.edu.
\*
\*
bin             directory with system programs used in single user mode
dev             directory with device files
etc             directory with maintenance programs
lib             directory with library programs
lost+found      directory for wayward files on this partition
mnt             directory used to mount new file systems
pub             directory with "public" stuff; really manufacturer's idea
stand           directory with standalone programs
sys             directory with system sources
tmp             directory used for temporary files by everyone
u1              directory containing one set of users
u2              directory containing another set of users
usr             directory containing links to u1 and u2
x.*             garbage files
```

The first five lines are comments. (How many files named * are in your root directory?) The remainder are entries for root subdirectories; since the important ones are audited separately, and the unimportant ones (like **/tmp**) are expected to change, we can ignore the changes in size to the directory files. The files beginning with "x" are temporary files included to show how to apply pattern matching; no files or directories beginning with "x" will be audited.

Note that the files are listed relative to the root of the file tree being audited. Had the file **bin** been written as **/bin**, the auditing tool would not have eliminated **bin** from the audit.

4.6. Reject

There are times when a system administrator wants to suppress messages about changes to a specific file; this usually happens when a file on an audited subsystem is updated or changed. The ignore file mechanism is not suitable for this, because ignore files *permanently* eliminate the file from the listing. A clumsy solution is to have the system administrator add the file name to the appropriate ignore file and delete it after the audit has run again.

Reject files provide a cleaner mechanism to do this. A reject file contains a list of file names, one per line; each file name must be a full path name and no pattern matching is done. (If the file name ends in "/", it is taken to be a directory, and both it and all files and subdirectories in the tree rooted at that directory are ignored.) When any of these

files show up in the audit listings, they are ignored, just as though they had been entered in the appropriate ignore file. However, if the reject file has not been modified (written) in the past ${*LIMIT*} days, it is deleted instead of being used.

A sample reject file is:

```
/vmunix
/bin/su
/usr/src/bin/su/
```

The first two lines state that any change to the files **/vmunix** and **/bin/su** are to be ignored; the third line states that if the directory **/usr/src/bin/su** or any file or subdirectory below it has been changed, that is to be ignored.

4.7. List

A **List** file lists the roots of the file trees to be audited, and for each root in the list specify a set of options that will be applied to that file tree only along with the other command line options. This also allows the auditing system to scan a different set of file trees for each system.

A **List** file consists of the names of the roots of the file trees to be scanned, one per line, followed by any options to be added to those specified on the command line when that file tree is scanned. Here is a sample **List** file:

```
/ –d
/bin
/usr/bin
/usr/ucb
/usr/unsupported/bin –u
```

The last line states that when the file system **/usr/unsupported/bin** is audited, the **–u** option is to be used in addition to any command line options; all other file systems just use the given command line options.

The **List** file goes in the appropriate host file directory; so,

* the **List** file for **icarus** is ${*LISTDIR*}**/icarus/List**;
* the **List** file for **hydra** is ${*LISTDIR*}**/hydra/List**; and
* the **List** file for **miranda** is ${*LISTDIR*}**/miranda/List**.

4.8. Equiv again

At times it is convenient to have the **F***, **I***, **Reject**, **List**, and **Environ** files for one host in different directories. For example, consider an environment with 35 computers all of a single type. Some are used for production runs, some for developing new software, some for network connections, and some for administration. In this case, it may be necessary to have 4 sets of master lists, three sets of ignore lists (assuming the network connection and administration systems use the same ignore lists), 2 sets of **List** files, and one common **Environ** file.

To allow this, the **Equiv** file has a special syntax for the second column. In this format, there are four subfields separated by commas. The first contains the name of the subdirectory with the master lists, the second the name of the subdirectory with the

ignore lists and reject file, the third the name of the subdirectory with the **List** file, and the fourth the name of the subdirectory with the **Environ** file. For example:

This file contains the mappings from host names to directories.

host	directory	any comments
icarus.riacs.edu	mriacs,iicarus,Listd,Envd	full domain names
hydra.riacs.edu	mriacs,ihydra,Listd,Envd	
miranda.riacs.edu	mriacs,imiranda,imiranda,imiranda	

With this configuration, audits of **icarus.riacs.edu** and **hydra.riacs.edu** will use the same set of master lists (those in the subdirectory **mriacs**) and the same **List** and **Environ** files (**Listd/List** and **Envd/Environ**, respectively) but separate sets of ignore lists and reject files. Audits of **miranda.riacs.edu** will use the same master lists as **icarus.riacs.edu** and **hydra.riacs.edu**, and all other data files for **miranda.riacs.edu**'s audit will be found in the subdirectory **imiranda**.

If fewer than four subfields are given, the last subfield is used for the missing ones. Thus, the last two **imirandas** could have been omitted from the last line of the sample **Equiv** file above.

5. Running an Audit

The actual auditing script is **audit**. The interface to this program is horrendous, because it is designed to be able to work for very odd configurations. (Masochists may refer to *audit*(8).) The script **runaudit** provides a simple, efficient interface, and is the way to run an audit easily.

Runaudit has several options:

−a This option lists the people to whom the results of the audit are to be sent. If this option is given, no messages are sent to the address named in ${*AUDITOR*}. As an example, the option

−aauditor,audit@domain

sends the results of the audit to **auditor** and **audit@domain**.

−d Do not recurse down the file tree. If this option is given, *only* the root directory of the file tree is audited.

−g Generate a master file. Future audits will compare the information gathered from the files on the system with the data in this file, and report any differences.

−ga Audit the file tree, and then replace the master file with a new master file reflecting the state of the system at the time the audit is done. This is useful when the system changes often; the auditors see the changes once, when they occur, rather than every day thereafter.

−h This option runs an audit on a remote system. The option

−hicarus

will audit the host **icarus**. Using this option causes *runaudit* to look in a specific directory (see the section on the **Equiv** file) for the data files; if no **Equiv** file is present, or the host is not listed in it, the name of the directory is that of the given host.

-n This option lists the types of volumes on which no auditing is to be done. For example, if only locally-mounted file systems are to be audited, the option

-nnfs,rfs

will prevent any file system mounted on volumes of type *nfs* or *rfs* (both of which usually indicate that the file volumes are mounted remotely) from being audited. This is useful in an environment where there are file servers, such as NFS.

-o With this option, only filesystems mounted on volumes of the specified types will be audited. For example, if only remote-mounted file systems are to be audited, the option

-nnfs,rfs

will prevent any file system mounted on volumes of type other than *nfs* and *rfs* (both of which usually indicate that the file volumes are mounted remotely) from being audited. It is the exact complement to the option **-n**. This is useful in an environment where there are file servers, such as NFS.

-sg Check only those files with the setgid bit set. This is done regardless of the group of the file.

-su Check only those files with the setuid bit set. This is done regardless of the owner of the file.

-u When this option is given, the audit system mails a letter to every user who owns a file that has changed. The message is in the form of an audit report. If the ownership of a file has changed, both the old and new owners receive messages. This is useful in scanning file trees where users put contributed software.

A typical set of commands to audit a computer system would be:

```
runaudit -d -ga -aroot
runaudit -ga -su -sg -aroot /
```

The first line instructs the audit system to scan the directories listed in *${LISTDIR}/localhost*/List and report changes. The second command scans the entire UNIX file system, and reports changes made to setuid and setgid files.

6. Reading An Audit Report

A typical audit report looks like this:

```
From root Wed May 20 11:05:55 1987
Received: by hydra.riacs.edu (4.12/2.0N)
        id AA07912; Wed, 20 May 87 11:05:53 pdt
Message-Id: <8705201805.AA07912@hydra.riacs.edu>
Date: Wed, 20 May 87 11:05:53 pdt
From: root <root>
To: mab
Subject: Audit report for hydra.riacs.edu:/tmp
Status: R

Audit date:  Wed May 20 11:05:40 PDT 1987
Host system: hydra.riacs.edu
File system: /tmp
Options: update-master-list,nonrecursive
Copies sent to: mab
```

```
The following files have been added:

file name   type  mode  links  user   group  size            date
---------   ----  ----  -----  ----   -----  ----            ----
Aml0132       -   0644      1  mab    root      0  May 19, 1987 at 07:27:37

The following files have been deleted:

 file name    type mode  links   user    group  size            date
 ---------    ---- ----  -----   ----    -----  ----            ----
Aml0130         -  0644      1  mab     root      0  May 19, 1987 at 07:27:37
dqms007633      -  0666      1  daemon  root    118  May 20, 1987 at 11:02:38

The following files have been changed; the previous attributes
are shown on the line with (old), and the current attributes
are shown on the line with (new):

     file name       type  mode  links  user   group  size            date
     ---------       ----  ----  -----  ----   -----  ----            ----
(old) .                 d  0777      5  root   root   3072  May 20, 1987 at 11:02:45
(new) .                 d  0777      5  root   root   3072  May 20, 1987 at 11:05:24
(old) newsa007443       -  0600      1  mab    root   6936  May 20, 1987 at 10:59:31
(new) newsa007443       -  0600      1  mab    root   6936  May 20, 1987 at 11:04:53
```

The first part is the header of the letter. The next part gives some information about the audit: it was made at 11:05 AM on May 20, 1987, for the directory **/tmp** on **hydra.riacs.edu**, and the master file for **/tmp** was updated (''update-master-list''). No subdirectories were audited (''nonrecursive''), and copies of the audit report were mailed to the user ''mab''. New files, old files, and changed files are reported; the audit report gives as complete information as possible.

Here is a table of options and what they mean:

Possible Options in the Field ''Options: ''	
listed option	*command line option*
nonrecursive	–d
notify-users	–u
setgid	–sg
setuid	–su
update-master-list	–ga

Occasionally an error may be encountered when the audit is running. In this case, an appropriate error message is sent to the auditors:

```
From root Wed May 20 11:03:56 1987
Received: by hydra.riacs.edu (4.12/2.0N)
        id AA07749; Wed, 20 May 87 11:03:51 pdt
Message-Id: <8705201803.AA07749@hydra.riacs.edu>
Date: Wed, 20 May 87 11:03:51 pdt
From: root <root>
To: mab
Subject: errors from runaudit command
Status: R

command: runaudit -ga -d /tmp
host: hydra.riacs.edu

/usr/mab/tasks/security/src/audit/audit: changing -ga option to -g
    ... generating /usr/mab/tasks/security/src/audit/HOSTS/hydra/Ftmp
```

The error message indicates the command being run and the host on which the command was run (which may or may not be the host being audited.) With this message, there was no master file, so the option ''–ga'' was not applicable. The audit program assumed the option should have been ''–g''.

7. Error Messages and What To Do

This section lists possible error messages from the audit system and how to handle them. System errors may come from a number of other programs, and these are not included.

audit: expecting list of people to mail audit to

The option ''–a'' was given without any list of addresses.

audit: expecting name of root of tree to be audited

This means that no root directory was specified. If **runaudit** is used, this error message should not occur; if **audit** is used, it means there was no file name after the ''–f'' option.

audit: expecting file of pathnames to ignore

This means that no ignore file was specified. If **runaudit** is used, this error message should not occur; if **audit** is used, it means there was no file name after the ''–i'' option.

audit: expecting file to compare audit against

This means that no master file was specified. If **runaudit** is used, this error message should not occur; if **audit** is used, it means there was no file name after the ''–m'' option.

audit: expecting name of host to audit

The option ''–h'' was given without any list of addresses.

audit: master file *filename* is unreadable

The master file exists and cannot be read.

audit: changing –ga option to –g
... generating *filename*

The option ''–ga'' was specified and no master file exists. The audit simply generates the master file.

audit: master file *filename* does not exist

No master file exists for the file system to be audited.

********** WARNING **********
There is a potential problem with the file system *filesystem*
on *host* -- the audit showed that file system has *pct%*
of the files it had when the master file was made. Either
the audit failed or most files on that file system have
been deleted. Check to be sure it is not the latter.

and if the master file must be regenerated, delete the
current one and replace it with *file*.
****

The difference in what should be in *filesystem* on *icarus* and what the audit program picked up as being there seems too much to be reasonable. Check to be sure *filesystem* was not obliterated. The results of the questionable file system scan were saved in *file*.

********** **WARNING** **********
There is a potential problem with the file system *filesystem*
on *host* -- the audit showed that file system has *pct%*
of the files it had when the master file was made. Either
the audit failed or most files on that file system have
been deleted. Check to be sure it is not the latter,
and if the master file must be regenerated, delete the
current one and replace it with *file*.
+++ NOTE: THE MASTER FILES HAVE NOT BEEN UPDATED +++
****

This is the same as the previous error message, except that the auditing system was supposed to update the master file. The update was not done. If it should have been done (that is, if the scan was correct), replace the current master file with *file*.

gettype: need a file system name

The program **gettype** was called with zero, two, or more arguments. This should never happen, since **gettype** is called from within **audit**.

Usage: auditfmt file1 ...

The program **auditfmt** was called with zero, two, or more arguments. This should never happen, since **auditfmt** is called from within **audit**.

auditgrep: bad option *option*
auditgrep: bad field number *n*
auditgrep: too few args

The program **auditgrep** was called incorrectly. This should never happen, since **auditgrep** is called from within **audit**.

Usage: lstat -acdfgimnpstu <file>
lstat: too many options

The program **lstat** was called incorrectly. This should never happen, since **lstat** is called from within **audit**.

find: ...
ls: ...

Messages from *find*(1) or *ls*(1) mean that a problem was encountered during the scanning of the file tree. There are usually two reasons for this: either the auditing program did not have sufficient privileges to scan a part of the subtree (the "cannot open ..." and "cannot

chdir to ...'' messages from *find*(1) indicate this) or the file being scanned was temporary and was deleted after the name had been found but before any other information could be found.

8. Troubleshooting

Runaudit has two debugging options: –v and –x. These act just like the options –v and –x to the Bourne shell **sh**(1). The –x option prints the commands and their arguments as they are executed; it is very useful when strange error messages come from remote audits, because it is propagated to the file listing program (even if that is on another host.) The –v option is useful when the scripts are changed; its utility is more limited.

Both options generate lots of output, and unless you know the Bourne shell quite well, you may find the output confusing. So use these options judiciously.