Tripwire

Reference Guide

2.4

TRIPWIRE

Tripwire, Inc.
326 SW Broadway, 3rd Floor
Portland, OR 97205

tel: 1.877.TRIPWIRE
fax: 503.223.0182
http://www.tripwire.com
tripwire@tripwire.com

TW1003-02

About This Guide

Document List

The **Tripwire Installation Guide** describes installation procedures for Tripwire Manager and Tripwire for Servers software.

The **Tripwire for Servers User Guide** describes configuration and operation of Tripwire for Servers software.

The **Tripwire Manager User Guide** describes configuration and operation of Tripwire Manager software, which is used to manage multiple installations of Tripwire for Servers software.

The **Tripwire Reference Guide** contains detailed information about the Tripwire configuration and policy files.

The **Quick Reference Cards** summarize important functionality of Tripwire for Servers software.

You can access PDF versions of the Guides from the *docs* directories on the Tripwire Manager and Tripwire for Servers CDs.

You can access **online help** from the Tripwire Manager interface.

Conventions

This Guide uses the following typographic conventions.

Bold	in regular text indicates FTP and HTTP URLs, and emphasizes important issues.
Italic	indicates file and directory names.
`Constant`	in regular text shows commands and command-line options, and policy file rule attributes, directives, and variables.
Sans Serif	in examples shows actual user input on the command line.
Sans Serif Italic	in examples shows variables which should be replaced with context-specific values.
W	denotes sections of the text that apply only to Windows installations of Tripwire software. Unless otherwise specified, all references to Windows refer to both Windows NT and Windows 2000.
U	denotes sections of the text that apply only to UNIX or Linux installations of Tripwire software. Unless otherwise specified, all references to UNIX also refer to Linux.
[options]	the command reference section shows optional command-line arguments in brackets.
{ 1 | 2 | 3 }	the command reference section shows sets of possible options in braces, separated by the | character. Choose only one of the options.

Unless otherwise specified, command-line examples assume that the Tripwire *bin* directory is the current working directory.

Support

For the latest information and support for Tripwire products, visit the Tripwire website or contact Tripwire Technical Support.

Tripwire Support Website: **http://www.tripwire.com/support**

Tripwire Technical Support:

e-mail:	support@tripwire.com
toll-free:	1.866.TWSUPPORT (6am-6pm Pacific)
phone:	503.276.7663

General information: info@tripwire.com

Tripwire Professional Services

Tripwire Professional Services provides flexible service and support to meet your specific technical and deployment needs. If you would like Tripwire software deployment and implementation assistance, or additional training in using Tripwire software products, visit **http://www.tripwire.com** or contact your Tripwire Sales Representative.

Tripwire Educational Services

Obtain expert hands-on technical training and experience from a Tripwire Certified Instructor. Courses are offered by Tripwire Authorized Training Centers, and prepare you to install, configure, and maintain Tripwire software. Visit **http://www.tripwire.com** or contact your Tripwire Sales Representative for more information.

Contents

1

Configuration Files

Overview

This chapter describes the Tripwire for Servers configuration file and Agent configuration file and explains how to customize them for your environment.

Topics include:

- introduction to configuration files
- configuration file parameters
- Agent configuration file parameters

Introduction to Configuration Files

Configuration files contain parameters that control Tripwire for Servers local operation on host machines. Tripwire for Servers relies on two configuration files for operation. These files are located in the Tripwire for servers *bin* directory.

- The configuration file *tw.cfg* controls Tripwire for Servers operation on each local machine.
- The Agent configuration file *agent.cfg* controls Tripwire for Servers communication with Tripwire Manager. The Agent configuration file is not used when you run Tripwire for Servers as a standalone application.

U Tripwire for Servers uses an installation configuration file *install.cfg* during the installation process on UNIX platforms. The *install.cfg* file specifies the initial Tripwire for Servers directory structure and sets initial values in the *tw.cfg* and *agent.cfg* files. See page 25 of the Tripwire Installation Guide for more information about the *install.cfg* file.

Configuration File Parameters

Configuration file parameters control Tripwire for Servers operation on host machines. Some parameters are required and some are optional.

- Required parameters specify paths to Tripwire for Servers data files.
- Optional parameters enable optional integrity checking and reporting features.

Paths to Data Files

The following parameters specify absolute paths to Tripwire for Servers data files. These parameters are required. Tripwire for Servers does not allow you to save a configuration file if these parameters are incomplete.

Warning: We strongly recommend that you use fully-qualified paths for all path values. Relative paths are a security risk and may cause unpredictable behavior on some locales.

U The default values for these parameters in UNIX configuration files are identical to those shown in this section, except in UNIX paths are delimited by / characters, not \ characters. All UNIX paths are case-sensitive.

W In Windows you can use Universal Naming Convention (UNC) names *(\\machine\share)* for data path parameter values.

Policy File

This parameter specifies the path to the policy file used for integrity checking.

Parameter name: POLFILE
Default value: (*Tripwire root*)\policy\(*host name*).pol

Database File

This parameter specifies the path to the database file used for integrity checking.

Parameter name: DBFILE
Default value: *(Tripwire root)*\db*(host name)*.twd

Report File

This parameter specifies the path name for report files. Tripwire for Servers writes report files to this directory.

Parameter name: REPORTFILE
Default value: (*Tripwire root*)\report\(*host name*)-(*DATE*).twr

The host name and DATE variables represent the date and time of the integrity check. Tripwire for Servers uses them in the report file name.

Site Key File

This parameter specifies the path to the site key file that signs the Tripwire configuration and policy files.

Parameter name: SITEKEYFILE
Default value: (*Tripwire root*)\key\site.key

Local Key File

This parameter specifies the path to the local key file that signs the Tripwire database file and (optionally) report files.

Parameter name: LOCALKEYFILE
Default value: (*Tripwire root*)\key\(*host name*)-local.key

Data Files Permissions

Tripwire for Servers data files permissions specify Read/Write/Execute permissions for the software's data files. The three octal digit value represents Read/Write/Execute permissions for the file's Owner, the Owner's Group, and all Others. Tripwire for Servers sets these permissions at file creation time.

All Tripwire data files permissions revert to 644 (Read-only to all users but the Owner) by default when data files permissions are not specified.

	1st Digit (Owner)	2nd Digit (Group)	3rd Digit (Others)
Read (4)	4	4	4
Write (2)	2	--	--
Execute (1)	--	--	--
sum:	**6**	**4**	**4**

U In UNIX the default value of 644 gives:

- Read and Write permission to the Owner
- Read-only permission to the Group
- Read-only permission to all Others

In UNIX you cannot change these permissions via the system umask.

W In Windows, the 2 for Owner Write permission in the first digit sum turns the file's Read-only flag on or off.

- A first digit of 7, 6, 3 or 2 turns OFF the Read-only flag because these digit sums contain a 2.
- A first digit of 5, 4 or 1 turns ON the Read-only flag because these digit sums do not contain a 2.
- Tripwire for Servers ignores the second and third digit in Windows.

Policy Rights

This parameter specifies UNIX-style Read/Write/Execute permissions for the policy file.

Parameter name: POLICYRIGHTS
Default value: 644
Valid values: (*3 octal digit*s)

Database Rights

This parameter specifies UNIX-style Read/Write/Execute permissions for the database file.

Parameter name: DBRIGHTS
Default value: 644
Valid values: (*3 octal digit*s)

Report Rights

This parameter specifies UNIX-style Read/Write/Execute permissions for report files.

Parameter name: REPORTRIGHTS
Default value: 644
Valid values: (*3 octal digit*s)

Temporary Directory

This parameter specifies a directory for storing Tripwire for Servers temporary files.

Parameter name: TEMPDIRECTORY
Default value: */tmp* in UNIX
(C: or *System Dir*)*temp* in Windows

Integrity Checking Parameters

The following parameters control integrity checking operations.

Loose Directory Checking

This parameter overrides a number of directory and registry key object properties during integrity checks. You can use this parameter to reduce duplicate violations.

Parameter name: LOOSEDIRECTORYCHECKING
Default value: true
Valid values: true or false

If set to false, Tripwire for Servers reports two violations for some changes to files and subkeys. One violation is for the change to the object and one is for the change to its parent directory or registry key.

If set to true, Tripwire for Servers ignores the following properties so that the change to a parent directory or registry key is not reported.

UNIX directories	file size, number of links, access time, change time, modification time, number of blocks allocated, growing files, and all hashes
Windows directories	last write time, and last access time
Windows registry keys	number of subkeys, maximum length of subkey name, number of values, maximum length of value name, maximum length of data for any value in the key, and last write time

Warning: When set to true, this feature can introduce a security risk because Tripwire for Servers does not report some changes to directories and keys.

Ⓦ Reset Access Time

When Tripwire for Servers accesses a file system object during an integrity check, it changes the object's access time. In the Windows operating system, this parameter causes Tripwire for Servers to restore the access times of file system objects to their previous value.

Parameter name: RESETACCESSTIME
Default value: true
Valid values: true or false

If you want to retain original access times for data forensics, set this parameter to true.

Ⓤ Traverse Mount Points

This parameter causes Tripwire for Servers to cross file system mount points during integrity checks.

Parameter name: TRAVERSEMOUNTS
Default value: false
Valid values: true or false

Warning: By default, Tripwire for Servers does not cross file system mount points during integrity checks. Setting this parameter to true may introduce security risks. If you set this parameter to true we recommend you limit recursion by adding `recurse` attributes to the policy file. See page 54 for more information about `recurse` attributes.

E-mail Parameters

These parameters control e-mail report operations.

Mail Method

This parameter specifies a protocol for sending e-mail reports.

Parameter name: MAILMETHOD
Valid values: SMTP, sendmail, or MAPI
Default value: SENDMAIL in UNIX

MAPI works only if a MAPI-enabled mail client is open on the Tripwire for Servers machine. For the best security, use SMTP or sendmail.

U Mail Program

This parameter specifies a path and arguments to a mail program for sendmail.

Parameter name: MAILPROGRAM
Dependency: Mail Method must be set to sendmail
Case-sensitive: yes
Default value: /usr/lib/sendmail -oi -t

The mail program must:

- take an RFC822-style mail header
- list recipients in the To field of the mail header
- ignore lines of a single period (the `-oi` command-line option to sendmail produces this behavior)

SMTP Host

This parameter specifies the domain name or IP address of the SMTP server when Mail Method is set to SMTP.

Parameter name:	SMTPHOST
Dependency:	Mail Method must be set to SMTP
Valid values:	IP address or domain name of SMTP server

SMTP Port

This parameter specifies the port number for SMTP when Mail Method is set to SMTP.

Parameter name:	SMTPPORT
Dependency:	Mail Method must be set to SMTP
Default value:	25
Valid values:	1 to 65535

From Address

This parameter specifies a resolveable From address for e-mail reports sent via SMTP or sendmail. This parameter does not work with MAPI.

Parameter name:	MAILFROMADDRESS
Valid values:	one resolveable SMTP e-mail address
Example:	root@domain.com
Case-sensitive:	no

Some mail servers may not deliver e-mail without a resolveable From address in the mail header. MAILFROMADDRESS causes Tripwire for Servers to place the specified resolveable From address into the mail header of e-mail reports. This decreases the possibility that a mail server may refuse to deliver Tripwire e-mail reports.

Mail No Violations

This parameter causes Tripwire for Servers to send notification that no violations were found when integrity checks detect no violations.

Parameter name: MAILNOVIOLATIONS
Default value: true
Valid values: true or false

If set to false, Tripwire for Servers does not send e-mail notification when it detects no violations.

If set to true, Tripwire for Servers sends an e-mail to notify you that no violations were found. This allows you to distinguish between integrity checks that detect no violations and scheduled integrity checks that fail to run.

Note: For the highest security, we recommend that you set this parameter to true.

E-mail Report Level

This parameter specifies a level of detail for e-mail reports.

Parameter name: EMAILREPORTLEVEL
Default value: 3
Valid values: 0 to 4

0	single line summary report; total adds, removes, and changes
1	parsable list of all violated objects
2	summary report; lists violations by section and rule name
3	compares expected and observed properties for each violated object; more concise than a level 4 report
4	full report; maximum level of detail

See Appendix B for report level samples.

Global E-mail Address

This parameter specifies e-mail addresses to receive an e-mail report of all violations after each integrity check. This is in addition to e-mail addresses specified with `emailto` attributes in the policy file. If Mail No Violations is set to false, a global e-mail address does not receive reports when integrity checks detect no violations.

Parameter name: GLOBALEMAIL
Default value: none
Valid values: any valid e-mail address or addresses

Delimit strings of multiple e-mail addresses with semicolons or commas.

```
GLOBALEMAIL=user@domain.com,root@domain.com
# or
GLOBALEMAIL=user@domain.com;root@domain.com
```

If Mail Method is set to MAPI, you can use MAPI addresses.

```
GLOBALEMAIL=Joe Admin,Root
# or
GLOBALEMAIL=Joe Admin;Root
```

See page 56 for more information about `emailto` attributes.

Character (Mail) Encoding

This parameter specifies a character set for Tripwire SMTP e-mail reports. This parameter does not work with MAPI.

Parameter name: MAILENCODING
Default value: auto
Valid values: auto (detects the OS character set)
none (no specific character set)
ISO-2022-JP

Logging Parameters

These parameters control logging operations.

Syslog Reporting

This parameter causes Tripwire for Servers to log a record of database initializations, integrity checks, database updates, and policy file updates to a system log file.

Parameter name: SYSLOGREPORTING
Default value: false
Valid values: true or false

U In UNIX, Tripwire for Servers makes log entries to the syslog from the user facility at the notice level.

W In the Windows operating system, Tripwire for Servers makes log entries to the application event log.

Syslog Report Level

This parameter specifies a level of detail for syslog entries made for integrity checks.

Parameter name: SYSLOGREPORTLEVEL
Dependency: Syslog Reporting must be set to true
Default value: 0
Valid values: 0 to 2

0	single line summary syslog entry; total adds, removes, and changes
1	separate syslog entry for each violation; entry shows only that a violation occurred
2	separate syslog entry for each violation; entry shows that a violation occurred, and which properties were violated

Ⓦ Syslog Host

This parameter causes Tripwire for Servers to log the syslog entries to a remote host machine or number of host machines.

Parameter name: SYSLOGHOST
Valid values: \\remote_host

You can specify multiple remote hosts like this. Precede each host name with two \ characters.

Ⓦ

```
SYSLOGHOST=\\host1 \\host2 \\host3 ...
```

Audit Log

On Windows and Solaris machines, this parameter allows integration of Tripwire for Servers with CyberSafe Centrax software.

Parameter name: AUDITLOG
Dependency: Syslog Reporting must be set to true
Default value: false
Valid values: true or false

SNMP Parameters

Tripwire for Servers can send Simple Network Management Protocol (SNMP) messages to an enterprise management host after each integrity check. These parameters control this SNMP feature.

A Management Information Base (MIB) file containing information for Tripwire for Servers SNMP V1 traps is located on the Tripwire for Servers CD in the *SNMP* directory.

SNMP Host

This parameter causes Tripwire for Servers to send an SNMP message trap to the specified host. The information in the SNMP trap is identical to a level 0 e-mail report (a one-line summary of total violations).

Parameter name: SNMPHOST
Valid values: IP address or domain name of the SNMP host

SNMP Port

This parameter specifies the port on the SNMP host that Tripwire for Servers should use for SNMP traffic.

Parameter name: SNMPPORT
Default value: 162
Valid values: 1 to 65535

SNMP Community

This parameter sets the community name in the SNMP trap messages from Tripwire for Servers.

Parameter name: SNMPCOMMUNITY
Default value: public
Valid values: any text string
Case-sensitve: no

Other Operations Parameters

These parameters control Tripwire for Servers command-line operations. These parameters have meaning only for command-line administration of Tripwire for Servers.

Late Prompting

This parameter causes Tripwire for Servers to delay the prompt for passphrases on the command line until the last moment. This minimizes the amount of time a passphrase stays in memory.

Parameter name: LATEPROMPTING
Default value: true
Valid values: true or false

For the highest security, set Late Prompting to true.

Report Level

This parameter specifies a default level of display detail for Tripwire report files printed from the command line.

Parameter name: REPORTLEVEL
Default value: 3
Valid values: 0 to 4

0	single line summary report; total adds, removes and changes
1	parsable list of all violated objects
2	summary report; lists violations by section and rule name
3	compares expected and observed properties for each violated object; more concise than a level 4 report
4	full report; maximum level of detail

See Appendix B for report level samples.

Editor

This parameter sets an absolute path to a text editor for interactive integrity checks. Interactive integrity checks allow an interactive update of the database file directly after an integrity check. See page 41 of the Tripwire for Servers User Guide for more information about interactive integrity checks.

Parameter name: EDITOR
Default value: */bin/vi* in UNIX
system default text editor in Windows

To be a valid text editor, a text editor must:

- accept a file on the command line
- support multi-byte characters
- exit with 0 status on success and non-0 status on error.

U Both vi and emacs satisfy the text editor requirements in UNIX.

If the configuration file does not specify an editor and no editor is specified on the command line, Tripwire for Servers looks at the $VISUAL or $EDITOR environment variables. If these do not specify an editor, Tripwire for Servers displays an error message.

W Both Notepad and Wordpad satisfy the text editor requirements in Windows.

Agent Configuration File

The Tripwire Agent manages communication between Tripwire for Servers and Tripwire Manager. The Agent configuration file parameters control Tripwire Agent operations.

U In UNIX the Tripwire Agent is a daemon.

W In a Windows operating system the Tripwire Agent is a service.

You must restart the Tripwire Agent daemon or service to enable any changes you make to the Agent configuration file.

The installation process generates the default values in the Agent configuration file. We recommend you use the default values unless you require special Tripwire Agent configuration.

Tripwire Agent Parameters

This section describes the Agent configuration parameters.

Warning: We strongly recommend that you use fully-qualified paths for all path values. Relative paths are a security risk and may cause unpredictable behavior on some locales.

PORTNUMBER

This parameter specifies the port used for communication with Tripwire Manager.

Default value: 1169 (registered Tripwire port)
Valid values: 1 to 65535

Ports below 1024 are restricted to system access only. If you do not use port 1169, choose only a known available port.

IPADDRESS

This parameter specifies an IP address for Tripwire Agent communication with Tripwire Manager.

Valid values: any IP Address

If a Tripwire for Servers machine has more than one network interface card (NIC), use this parameter to specify the NIC you want Tripwire Agent to listen on. If you do not specify an IP address, Tripwire Agent uses the Tripwire for Servers machine's NIC IP address by default.

TWCFGFILE

This parameter's value is the path to the configuration file. The Tripwire Agent reads the configuration file for the location of the Tripwire data files.

Default value: (*Tripwire root*)\bin\tw.cfg

SITEKEYFILE

This parameter's value is the path to the site key that cryptographically signs the *agent.cfg* file.

Default value: (*Tripwire root*)\key\site.key

This may be the same site key file used to sign Tripwire data files, or a different site key file.

TRIPWIRE

This parameter specifies the path to the `tripwire` executable file.

Default value: (*Tripwire root*)\bin\tripwire.exe

TWADMIN

This parameter specifies the path to the twadmin executable file.

Default value: (*Tripwire root*)\bin\twadmin.exe

TWPRINT

This parameter specifies the path to the twprint executable file.

Default value: (*Tripwire root*)\bin\twprint.exe

AUTHKEYFILE

This parameter specifies the path to the authentication key file. The authentication key file stores the keys Tripwire Agent uses to authenticate connections with Tripwire Manager.

Default value: (*Tripwire root*)\key\authentication.dat

AUTHKEYFILERIGHTS

This parameter specifies UNIX-style Read/Write/Execute permissions for the authentication key file.

Default value: 644

See page 6 for more information about data file permissions.

SCHEDULEFILE

This parameter specifies the path to the schedule file. The schedule file stores scheduling information for integrity checks.

Default value: (*Tripwire root*)\db\schedule.dat

SCHEDULEFILERIGHTS

This parameter specifies UNIX-style Read/Write/Execute permissions for the schedule file.

Default value: 644

See page 6 for more information about data file permissions.

TASKFILE

This parameter specifies the path to the task file. The task file stores information about completed tasks.

Default value: (*Tripwire root*)\db\tasks.dat

TASKFILERIGHTS

This parameter specifies UNIX-style Read/Write/Execute permissions for the task file.

Default value: 644

See page 6 for more information about data file permissions.

LOGFILE

This parameter specifies the path to the log file.

Default value (*Tripwire root*)\report\agentlog.txt

U In UNIX, Tripwire for Servers makes log entries to the syslog from the user facility at the notice level.

W In the Windows operating system, Tripwire for Servers makes log entries to the application event log.

LOGFILERIGHTS

This parameter specifies UNIX-style Read/Write/Execute permissions to the log file.

Default value: 644

See page 6 for more information about data file permissions.

2

The Policy File

Overview

This chapter describes the Tripwire for Servers policy file and explains how to customize it for your environment.

Topics include:

- introduction to the policy file
- policy file sections
- rules
- variables
- rule attributes
- directives

Introduction to the Policy File

The Tripwire for Servers policy file contains policies or rules for specific objects (such as files, directories, and registry keys) on a computer system. By writing policy file rules, you specify which system objects Tripwire for Servers scans during integrity checks. By modifying policy file rules, you change how Tripwire for Servers scans objects during integrity checks.

The policy file performs two functions. Initially, it acts as a blueprint for the Tripwire database file. When you initialize a database file, Tripwire for Servers reads the policy file to determine which objects and properties to include in the database file's baseline data.

Later, Tripwire for Servers reads the policy file each time it performs an integrity check. It then scans the system according to the policy file's rules and compares the scan against the baseline data in the database file. Inconsistencies between the two sets of data are reported as violations or errors in the integrity check's report file.

Default Policy Files

Tripwire for Servers installs a minimal default policy file *(twpol.txt,* located in the Tripwire *policy* directory) for your operating system (OS). This default policy file monitors basic components common to all versions of your OS. It does not monitor version-specific OS components or the applications or files specific to your system.

Because the policy file specifies which objects Tripwire for Servers monitors, it is very important to customize a policy file for your specific system configuration. A customized policy file allows Tripwire for Servers to provide the best integrity assurance for your system.

Policy File Resources

The following resources are available to help you construct a customized policy file for your system.

- This chapter describes the policy file language. You can construct your own rules easily after learning a few basic syntax principles. Syntax for rule construction begins on page 29.
- A policy file syntax guide (*policyguide.txt,* located in the Tripwire *policy* directory) provides syntax examples in context for additional help in learning the policy file language.
- The Tripwire for Servers CD and Tripwire Manager CD *policyfiles* directories contain OS version-specific policy files. These policy files provide more extensive coverage than the minimal default policy file created by the installer. We recommend that you replace your initial default policy file with one of these. See page 21 of the Tripwire for Servers User Guide for more information.
- The Tripwire Policy Resource Center (**http://policy.tripwire.com**) provides custom-made policy file syntax you can copy and paste to build a policy file for your specific system. See the website or page 21 of the Tripwire for Servers User Guide for more information about this resource.

Introduction to the Policy File Language

The policy file language resembles some scripting languages. The policy file language is made up of these components.

Comments exclude (comment out) text from functional parsing. See page 30 for detailed information.

Rules specify object such as files, directories, and registry keys to scan during integrity checks. See page 35 for detailed information.

Rule attributes assign names or severity levels to rules, specify e-mail addresses for e-mailed reports, and specify recursion levels for scanned directories and registry keys. See page 50 for detailed information.

Variables can substitute for other items in the policy file. Tripwire for Servers provides some predefined variables and allows you to define your own. See page 60 for detailed information.

Stop points exclude files, directories and registry objects from an integrity check. See page 69 for detailed information.

Directives organize rules into major sections and allow conditional logic. See page 33 and page 70 for detailed information.

See page 30 for a brief description of these components. The rest of this chapter discusses these components in detail.

```
# Examplesof policy file language                          # comment
c:\winnt ->   &access;                                     # rule
c:\winnt\file  ->   &sha (emailto=user@domain.com);        # rule with attribute
TEST = c:\winnt\test;                                      # variable definition
$(TEST) ->   &size;                                        # variable substitution
!c:\winnt\documents;                                       # stop point
@@ifhost ruby                                              # directives
     c:\winnt\project -> &haval;
@@else
     c:\winnt\project -> &sha;
@@endif
```

Comments

Comments allow you to include explanations, instructions, and other non-syntax text within the policy file. Comments can help to explain or clarify a policy file's contents.

Tripwire for Servers ignores commented text in the policy file. It parses or reads only un-commented text. Comments begin with a # character and extend to the end of a line. If you want to continue a comment on the next line, place another # character at the beginning of the next line.

```
# This is a comment.
object name -> properties;  # A comment can go here, too.  To continue
# a comment on the next line, use another # character.
```

Rules

Rules specify system objects for Tripwire for Servers to scan during integrity checks. A basic rule includes an object (such as a file, directory or registry key) and the properties of that object to check or ignore.

U

```
/etc/home -> pinugs;        # This rule tells Tripwire for Servers to scan /etc/home
                            # for the specified properties.
```

W

```
c:\winnt -> &size &sha;     # This rule tells Tripwire for Servers to scan c:\winnt
                            # for the specified properties.
```

See page 35 for detailed information.

Rule Attributes

You can add optional rule attributes to rules to extend their functionality beyond basic integrity checking. Rule attributes modify how Tripwire for Servers executes a rule or reports the violation of a rule.

U

```
/etc/home -> p (recurse=2);    # This rule attribute tells Tripwire for Servers to
                               # recurse only two levels into /etc/home.
```

W

```
c:\winnt -> &size (recurse=2); # This rule attribute tells Tripwire for Servers to
                               # recurse only two levels into c:\winnt.
```

Rule attributes specify rule names, severity levels, levels of recursion into directories and registry keys, and e-mail addresses for notification of rule violations. See page 50 for detailed information.

Variables

Variables substitute for other items in the policy file. You can define your own variables, then substitute them for certain parts of policy file language, including rule objects, properties, and rule attribute values.

U

```
ROOT=/etc;        # Define a variable ROOT
$(ROOT) -> pin;   # This rule tells Tripwire for Servers to scan ROOT (/etc).
```

W

```
ROOT=c:\winnt;   # Define a variable ROOT
$(ROOT) -> &sha;# This rule tells Tripwire for Servers to scan ROOT (c:\winnt).
```

Tripwire for Servers provides some predefined variables for common property combinations. See page 60 for detailed information.

Stop Points

Stop points specify objects to exclude from integrity checks. Tripwire for Servers does not scan any object preceded by the `!` character.

U

```
!/etc;          # This stop point tells Tripwire for Servers to skip /etc.
```

W

```
!c:\winnt       # This stop point tells Tripwire for Servers to skip c:\winnt.
```

See page 69 for detailed information.

Directives

Directives allow sectioning and conditional logic in the policy file. Directives begin with `@@` followed by a directive name. The directive name (`section`, `end`, `ifhost`, `else`, `endif`, `error`, or `print`) determines which function the directive performs.

U

```
@@section FS      # This directive declares a UNIX File System section.
```

W

```
@@section NTFS  # This directive declares a Windows File System section.
```

Detailed information about the `@@section` directive and policy file sections begins on page 33. Detailed information about other directives begins on page 70.

Policy File Sections

Policy file sections allow Tripwire for Servers to differentiate between different types of rule objects. By sectioning a policy file, you can:

- define variables to use throughout the policy file
- use a single policy file to check objects on different platforms
- use a single policy file to check both file system objects and registry objects on Windows

If you have rules for more than one type of system object in a policy file, you must section that policy file. This is so that Tripwire for Servers can interpret the rules correctly.

You can use the following section types in a policy file.

GLOBAL	defines global variables (both UNIX and Windows) that you can use throughout subsequent sections of a policy file
FS	UNIX file system rules
NTFS	Windows file system rules
NTREG	Windows registry rules

Policy files can contain multiple `GLOBAL`, `FS`, `NTFS`, and `NTREG` sections.

Note: Because you must define variables before you use them in syntax, it is best to put global variables sections at the top of a policy file.

How to Section a Policy File

The `@@section` directive declares the beginning of a new section. The `GLOBAL`, `FS`, `NTFS`, and `NTREG` directive arguments specify a section type. All syntax before the first `@@section` directive is interpreted as FS syntax on a UNIX machine and as NTFS syntax on a Windows machine. All syntax following each `@@section` directive is interpreted as one section until the next `@@section` directive.

A sectioned policy file looks like this.

U

```
# global variables section
@@section GLOBAL
variable= value;
variable= value;
...
# file system section — rules for file system objects
@@section FS
rule;
rule;
...
```

W

```
# global variables section
@@section GLOBAL
variable=value;
variable=value;
...
# file system section — rules for file system objects
@@section NTFS
rule;
rule;
...
# registry section — rules for registry objects
@@section NTREG
rule;
rule;
...
```

Rules

Policy file rules specify the objects that you want to include in integrity checks. This section describes the elements of policy file rules.

Each rule specifies:

- an object on your system
- which properties of the object to check or ignore
- optional rule attributes

---

object name -> properties [attributes];

The *object name* specifies a fully qualified path to a file system or registry object and *properties* specify the properties of the object to check or ignore.

Optional rule *attributes* specify a rule name, a severity level, a level of recursion into objects, or e-mail addresses for e-mailed notification of rule violations. See page 50 for more information about rule attributes.

Constructing Rules

When constructing rules, follow these syntax conventions.

- Use a -> token (- and > characters) to delimit the rule's object and properties.
- Terminate each rule with the ; character.
- Each system object may appear in one rule only. If you want to check multiple properties for a single object, list them in a single rule.

```
# valid
object name1   -> property 1   property 2 ... ;

# invalid: a policy file cannot contain two rules with the same object name
object name1   -> property 1;
object name1   -> property 2;
```

Windows registry objects require different conventions (see page 38).

Object Names

An object name specifies a system object to scan during integrity checks. Object names must be case-sensitive, absolute paths to files, directories, or registry keys and values. For security reasons, you cannot use environment variables in object names.

U

```
/etc              # valid
$HOME             # invalid: cannot use environment variable
```

W

```
c:\winnt          # valid
%PATH%\temp       # invalid: cannot use environment variable
```

UNIX File System Object Names

UNIX file system object names must be absolute paths to system objects. UNIX object names are case-sensitive.

U

```
/etc/local      # valid
/etc/Local      # invalid if you want to scan /etc/local
```

Windows File System Object Names

Windows file system object names must be absolute paths to directories and files. Paths must begin with a single-character drive specifier (x:) or you can use Universal Naming Convention (UNC) names. Windows file system object names are not case-sensitive.

W

```
c:\programs
# or
C:\Programs
```

If you use UNC for Windows file system object names, you can use either \ or / as the path delimiter. UNC object names must begin with // or \\. The smallest acceptable UNC name is *machine**share*.

W

```
\\ruby\diamond                  # valid
\\ruby                          # invalid: no share
```

You must quote administrative shares expressed with their UNC path.

W

```
\\spock\"C$"
```

Windows Registry Object Names

Windows registry object names must be registry keys and values.

```
key                 # for a key
key\subkey          # for a subkey
key|value           # for a value
```

The *key* is a fully specified key or subkey and *value* is the value for that key or subkey. When constructing rules for registry objects, follow these syntax conventions.

- Use the \ character as the path delimiter for subkeys. If you quote a subkey path, remember that you must precede delimiting \ characters within quoted strings by another \ character.
- Use the | character to specify Windows registry values.
- Tripwire for Servers interprets unquoted registry key and registry value strings literally.
- Handle restricted characters the same as for file system objects (see page 40). You can use the \ character inside a quoted string as an escape sequence.
- Do not quote the | separating a registry key name and registry value.

```
# valid
HKEY_LOCAL_MACHINE\HARDWARE
HKEY_CLASSES_ROOT\"Media Type"

# valid
"HKEY_CURRENT_USER\\Remote Access"|InternetProfile

# invalid
"HKEY_LOCAL_MACHINE\SOME KEY"# invalid: single \ delimiter within quotes
"HKEY_CURRENT_USER|Some Value"# invalid: do not quote the | delimiter
```

Special Characters in Object Names

Tripwire for Servers requires special handling of some characters in object names. This section describes these special characters.

U Restricted Characters in UNIX Object Names

Tripwire for Servers does not allow the following characters in unquoted UNIX object names. These characters are reserved for functional meaning within the Tripwire for Servers product.

!	exclamation point	;	semicolon
[ ] or { }	braces	=	equal sign
>	greater-than sign	$	dollar sign
( )	parentheses	#	hash
,	commas	|	pipe
	white spaces	\	backslash (see Note)
+	plus sign	’	single quote

If an object name contains a restricted character, quote the entire path or the section that contains the character. Quoting allows Tripwire for Servers to read restricted characters literally without interpreting their functional meaning. There is an exception for the \ character (see Note).

U

```
"/etc/local/accounts/$receipts" # quote an entire path   OR
/etc/local/accounts/"$receipts" # quote a section containing restricted characters
```

Note: A single \ character within a quoted string introduces the Tripwire for Servers escape sequence. If you quote a UNIX object name that contains a \ character, Tripwire for Servers interprets the \ character as an escape sequence.

U

```
"/tmp\myfile"     # invalid: the \ character within quotes is an escape sequence
```

Restricted Characters in Windows Object Names

Tripwire for Servers does not allow the following characters in unquoted Windows object names. These characters are reserved for functional meaning within the Tripwire for Servers product.

!	exclamation point	;	semicolon
{ }	curly braces	=	equal sign
[ ]	brackets	$	dollar sign
( )	parentheses	#	hash
,	commas	|	pipe
	white spaces	`	single quote
+	plus sign	>	greater-than sign

If an object name contains a restricted character, quote the entire path or the section that contains the character. Quoting allows Tripwire for Servers to read restricted characters literally without interpreting their functional meaning. However, quoting a path that is delimited by the \ character requires special handling (see Note).

```
c:\"my documents"     # valid: quote path sections containing white space
"c:\my documents"     # invalid: see Note
```

Note: A single \ character within a quoted string introduces the Tripwire for Servers escape sequence. When you quote a path delimited by \ characters, Tripwire for Servers interprets each delimiter as an escape sequence.

To prevent this, precede each delimiter in a quoted path with another \ character. This escapes the original \ character so that Tripwire for Servers does not interpret it as an escape sequence.

```
"c:\\winnt\\my docs"  # valid: double all \ delimiters within a quoted path
```

U Nonprintable Characters in UNIX Object Names

If UNIX object names contain the following nonprintable character sequences, you must escape the nonprintable characters. The Tripwire for Servers escape sequence is a \ character inside a quoted string.

If you do not escape these nonprintable character sequences, Tripwire for Servers interprets their functional meaning.

Note: If you escape character sequences other than those listed, Tripwire for Servers treats them as if they are not escaped.

\t	tab	\a	bell ring
\v	vertical tab	\\	literal backslash
\b	backspace	\?	literal question mark
\r	carriage return	\'	literal single quote
\f	form feed	\"	literal double quote

For example, Tripwire for Servers expands `\t` in a quoted object name to a literal tab. Therefore, if you want to state `\t` literally (not as a tab) within a quoted object name, escape it with a preceding \ character.

U

```
/tmp/xx\tx          # interpreted by Tripwire for Servers literally as /tmp/xx\tx

# quoted
"/tmp/xx\tx"        # interpreted by Tripwire for Servers as /tmp/xx "tab" x

# escaped
"/tmp/xx\\tx"       # interpreted by Tripwire for Servers as /tmp/xx\tx,
                    # the same as the unquoted file name
```

Note: Tripwire for Servers interprets escape sequences in the same way the C++ programming language does.

Hexadecimal, Octal, or Unicode Characters

You must escape and quote any nonprintable single-byte hexadecimal, Unicode, or octal characters as follows.

Single-byte Hex	"\xXX"	X is a case-insensitive hexadecimal digit 0123456789ABCDEF Maximum length of an escaped hex sequence is two hex digits, with the sequence terminated by the first non-hex character or end quote.
Octal	"\OOO"	O is one of the octal digits 01234567 Octal character escape sequences may be one, two, or three digits.
Unicode	"\uXXXX"	X is a case-insensitive hexadecimal digit 0123456789ABCDEF Unicode escape sequences must be exactly four digits. **U** In UNIX, Tripwire for Servers accepts escaped unicode characters in the range \u0000 to \u00FF only.

```
/test              # reads "/test"
"/te\x73t2"        # reads "/test2" (hex)
"/te\163t3"        # reads "/test3" (octal)
"/te\u0073t4"      # reads "/test4" (unicode)
```

Double-byte Characters

Quote any object name that contains kanji, kana, or any other double-byte characters. Quote the entire path or the section containing the characters.

U

```
"/mysubdirectory/kanji characters"
# or
/mysubdirectory/"kanji text"
```

W

```
"c:\\winnt\\kanji characters"
# or
c:\winnt\"kanji text"
```

Wildcards

Tripwire for Servers interprets wildcard characters literally. Do not use wildcards in object names; instead, state the full object name.

U

```
/etc/p*         # interpreted literally—will not expand to passwd
/etc/passwd     # valid
```

W

```
c:\win*         # interpreted literally—will not expand to winnt
c:\winnt        # valid
```

White Space

Tripwire for Servers recognizes white space within object names only when you quote the path or section of the path containing the white space. Use quotes when you want Tripwire for Servers to recognize white space.

U

```
/usr/local/"my docs"->pinugs;
```

Remember to escape any \ delimiters within quotes by preceding them with another \ character.

W

```
c:"\\my documents"->&access;
```

Properties

The properties portion of a rule specifies the properties of the object to check or ignore during an integrity check. Each rule must positively specify at least one property. When you positively specify (include) properties, Tripwire for Servers checks them. When you negatively specify (exclude) properties, Tripwire for Servers ignores them.

---

object -> properties;

- Properties are case-sensitive.
- To include properties, place a + character before a property or series of properties. Tripwire for Servers assumes a + character when properties are not preceded by a + or - character.
- A + character before a series of properties causes Tripwire for Servers to include the entire series.
- To exclude properties, place a preceding - character before a property or series of properties, or simply omit them from the properties list.
- A - character before a series of properties causes Tripwire for Servers to ignore the entire series.

U

```
/mnt    ->  +p+n;          # check permissions and number of links
/mnt    ->  +pn;           # same; + applies to series
/mnt    ->  pn;            # same; + is assumed
/mnt    ->  PN;            # invalid: properties are case-sensitive
```

W

```
c:\winnt ->  +&access +&size;   # check access timestamp and file size
c:\winnt ->  +&access &size;    # same; + applies to series
c:\winnt ->  &access &size;     # same; + is assumed
c:\winnt ->  &Access &Size;     # invalid: properties are case-sensitive
```

Each rule must positively specify at least one property—a property list cannot consist of plus or minus characters only.

U

```
/etc   -> -+;              # invalid: must specify at least one property
/etc   ->  p;              # valid
```

W

```
c:\winnt ->   -+;          # invalid: must specify at least one property
c:\winnt ->   &size;       # valid
```

You can turn properties on and off in the same property list. Tripwire for Servers uses the last instance of a property if it occurs more than once in a rule.

U

```
/etc   -> +pns -s;                  # property turned on, then off
```

W

```
c:\winnt ->+&size &access -&size;  # property turned on, then off
```

However, if you turn a property off before you turn it on, Tripwire for Servers does not recognize the property. If the rule contains no other properties, this causes a syntax error because a rule must positively specify at least one property.

U

```
/etc   -> -s;                       # invalid: the stated property is turned off
```

W

```
c:\winnt ->  -&size;                # invalid: the stated property is turned off
```

Property Issues

When specifying properties, be aware of the following general issues.

Access timestamp (`a` in UNIX, `&access` in Windows)

The access timestamp property is incompatible with all hash properties. Hashes always cause a violation to the access timestamp because the access timestamp of an object changes when Tripwire for Servers accesses the object to calculate its hash.

Specifying the access timestamp property for a directory always causes a violation, because the access timestamp changes when Tripwire for Servers accesses a directory to check it.

To avoid these conflicts, you can set `recurse` to false in the rule's attributes, or set LOOSEDIRECTORYCHECKING and RESETACCESSTIME to true (RESETACCESSTIME is for Windows only) in the configuration file, or turn off the access timestamp property in a rule (add `-a` in UNIX or `-&access` in Windows) to the rule's properties).

Changes in file size (UNIX property `l`)

This property indicates you expect a file to grow and never shrink. An object size smaller than its last recorded size violates this property.

If a file grows from size A to size B where B > A, no violation is reported. The file's size information recorded in the database file remains size A.

If the file shrinks in size from B to C, where B > C > A, no violation is reported because C is still larger than A. Unless you explicitly update the database file with the new size C information, this change is not reported as a violation.

Properties for UNIX File System Objects

p	File permissions
i	Inode number
n	Number of links (inode reference count)
u	User ID of owner
g	Group ID of owner
t	File type
s	File size
d	Device number of the disk where the inode for the file is stored
r	For device object only; number of the device to which the inode points
b	Number of blocks allocated
m	Modification timestamp
c	Inode creation/modification timestamp
l	Changes in file size; file is expected to be larger than its last recorded size; see page 46
a	Access timestamp; incompatible with hashes (+CMSH); see page 46
C	CRC-32, for relatively high performance but relatively low security; see Glossary
M	MD5, for high security; see Glossary
S	SHA, part of the SHS/SHA algorithm; for high security; see Glossary
H	HAVAL, for high security; see Glossary

Properties for Windows File System Objects

&archive	Archive flag
&readonly	Read-only flag
&hidden	Hidden flag
&offline	Offline flag
&temp	Temporary flag
&system	System flag
&directory	Directory flag
&access	Last access time; incompatible with hashes; see page 46
&write	Last write time
&create	Create time
&size	File size
&msdosname	MS-DOS 8.3 name
&compressed	NTFS Compressed flag
&owner	NTFS Owner SID; see Appendix A
&group	NTFS Group SID; see Appendix A
&dacl	NTFS DACL; see Appendix A
&sacl	NTFS SACL; see Appendix A
&sdc	Security descriptor control; see Appendix A
&sdsize	Size of security descriptor for object; see Appendix A
&crc32	CRC-32, for relatively high performance but relatively low security; see Glossary
&md5	MD5, for high security; see Glossary
&sha	SHA, part of the SHS/SHA algorithm; for high security; see Glossary
&haval	HAVAL, for high security; see Glossary
&strm_count	Number of NTFS streams
&strm_crc32	CRC-32 hash of all non-default data streams; see Glossary
&strm_md5	MD5 hash of all non-default data streams; see Glossary
&strm_sha	SHA hash of all non-default data streams; see Glossary
&strm_haval	HAVAL hash of all non-default data streams; see Glossary

Properties for Windows Registry Key Objects

&write	Last write time
&owner	Owner SID; see Appendix A
&group	Group SID; see Appendix A
&dacl	DACL; see Appendix A
&sacl	SACL; see Appendix A
&sdc	Security descriptor control; see Appendix A
&sdsize	Size of security descriptor for the key; see Appendix A
&classname	Name of class
&nsubkeys	Number of subkeys
&maxsubkeyname	Maximum length of subkey name
&maxclassname	Maximum length of classname
&nvalues	Number of values
&maxvaluename	Maximum length for value name
&maxdatalen	Maximum length of data for any value in the key

Properties for Windows Registry Value Objects

&datatype	Type of value data
&datalen	Length of value data
&crc32	CRC-32 hash of value data; see Glossary
&md5	MD5 hash of value data; see Glossary
&sha	SHA hash of value data; see Glossary
&haval	HAVAL hash of value data; see Glossary

Rule Attributes

Rule attributes provide options for customizing your policy file. You can extend rule functionality with different combinations of the following rule attributes.

rulename	assigns meaningful names to rules
severity	assigns severity levels to rules
recurse	controls recursion into directories and registry keys
emailto	specifies recipients for e-mail reports

You can assign rule attributes to rules only, not to stop points or any other components of the policy file.

---

rule (attribute=value);

- Place rule attributes after a rule's properties and before the ending ; character.
- Rule attributes must be enclosed in ( ) characters.
- List multiple attributes delimited with , characters.
- Rule attributes and rule attribute values are case-sensitive.

---

single attribute
object -> properties (attribute=value);

# multiple attributes
object -> properties (*attribute1=value*, *attribute2=value*, *attribute3=value*);

You can assign attributes to single rules or groups of rules called rule blocks. When assigned to a rule block, rule attributes apply to every rule in the block. See page 56 for more information about rule attributes for rule blocks.

Specifying Rule Names

A `rulename` attribute assigns a name to a rule. Quote a `rulename` value if it contains white space.

---

object -> *properties* (rulename="Critical files");

You can use `rulename` attributes to help you interpret integrity check reports. Level 3 and 4 reports list violated object details by rulename (see Appendix B for sample reports).

You can run integrity checks by `rulename`. When you do, the integrity check scans objects with that rulename only.

If a rule has no `rulename` attribute, Tripwire for Servers uses the object name as the default `rulename`.

U

```
# this rule's rulename is "password"
/etc/passwd -> pinugs (rulename=password);

# but this rule's default rulename is "/etc/passwd"
/etc/passwd -> pinugs;
```

W

```
# this rule's rulename is "Web files"
c:\winnt\web  ->  &access &dacl (rulename="Web files");

# but this rule's default rulename is "c:\winnt\web"
c:\winnt\web  ->  &access &dacl;
```

Specifying Severity Levels

A `severity` attribute assigns a level of severity to a rule. For a specific severity value you can use any number from 0 to 1,000,000.

---

object -> *properties* (severity=75);

Severity levels can help you to identify important violations in a report. In Tripwire Manager, severity levels result in color-coded rule violations. See page 20 of the Tripwire Manager User Guide for more information about severity in Tripwire Manager.

You can run an integrity check by severity level. When you do, the integrity check scans only objects with the specified severity level or higher.

U

```
# A severity level 80 check scans the first object but not the second object.

/etc/local      -> m (severity=90);
/etc/passwd  -> m (severity=75);
```

W

```
# A severity level 80 check scans the first object but not the second object.

c:\winnt     -> &write (severity=90);
c:\projects -> &write (severity=75);
```

Default Severity

If you do not assign `severity` attributes to rules, there is no default `severity` level for rules in that policy file.

When you assign the first `severity` attribute to any rule in a policy file, Tripwire for Servers assumes a default severity of 0 for all other rules in that policy file. Be sure to assign `severity` levels to other rules when you assign severity to one rule. If you do not, important objects may remain classified at a 0 severity level.

Severity in Tripwire Manager

Tripwire Manager displays rule violations color-coded by severity level. The following severity levels show as blue, yellow, or red violations in Tripwire Manager. You cannot redefine these ranges.

Classification	Severity Range	Tripwire Manager displays
Low severity	0 to 32	blue report icons blue pie chart
Medium severity	33 to 65	yellow report icons yellow pie chart
High severity	66+	red report icons red pie chart

Specifying Recursion

The `recurse` attribute limits how many levels Tripwire for Servers recurses into objects during integrity checks. For a recurse value you can use true, false, or any number from 0 to 1,000,000.

object -> *properties* (recurse=75);

When a rule has no `recurse` attribute, Tripwire for Servers fully recurses the object by default. You can also set `recurse=true` or `recurse=-1` to specify full recursion.

U

```
# recurse the entire object by default
/etc -> s;

# specify full recursion
/etc -> s (recurse = true);
# or
/usr/local -> p (recurse=-1);
```

W

```
# recurse the entire object by default
c:\temp -> &size;

# or specify full recursion
c:\temp -> &size (recurse = true);
# or
c:\winnt -> &size (recurse=-1);
```

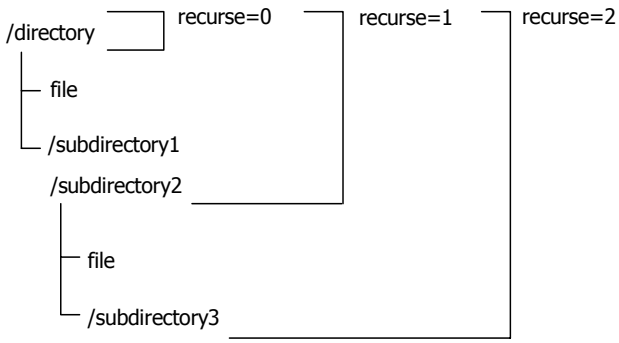
Turning Recursion Off

You can turn recursion off by setting `recurse=false` or `recurse=0`. When you do, Tripwire for Servers scans the object itself but does not recurse into any of its contents.

Numerical Recursion Levels

You can specify a numerical recursion value. A positive `recurse` value *`n`* scans up to *`n`* levels below the rule's start point.

- Set `recurse` to 0 to scan the properties of a directory but none of its contents.
- Set `recurse` to 1 to scan a directory, its files, and the properties of its first level of subdirectories.
- Set `recurse` to 2 to scan a directory, its files, the properties and contents of its first level of subdirectories, and the properties of its second level of subdirectories.



Note: You can use stop points to stop Tripwire for Servers from scanning certain objects while it recurses into directories and registry keys. See page 69 for more information about stop points.

```
HKEY_USERS ->  &owner &group (recurse=3);
!.DEFAULT;         # skips this object
```

Sending E-mail Reports

The `emailto` attribute specifies recipients for e-mail reports. When Tripwire for Servers detects the violation of any rule with an `emailto` attribute, it sends notice of that violation to the specified addresses.

Note: You must enable all required e-mail parameters in the configuration file so that Tripwire for Servers can deliver the e-mail to specified recipients. See chapter 1 for more information about e-mail parameters.

Specifying E-mail Addresses

You can use SMTP or MAPI addresses for `emailto` attribute values. When a single e-mail address contains white space, quote the address so that Tripwire for Servers recognizes the white space.

---

object -> properties (emailto=user@domain.com);
object -> properties (emailto="Joe Admin");

To specify multiple e-mail addresses, list them delimited by ; characters and quote the entire string. Tripwire for Servers ignores any leading and trailing white spaces between multiple addresses.

---

object -> properties (emailto="user@domain.com; Joe Admin");

Global E-mail

If you want Tripwire for Servers to send an e-mail report of all violations after each integrity check, use the Global E-mail parameter in the configuration file (see page 13). Global e-mail recipients receive a full report after each integrity check. This is in addition to e-mail addresses specified with `emailto` attributes in the policy file.

Using Rule Attributes to Construct Rule Blocks

You can group rules together and apply one set of attributes to the group. This construction is called a rule block. Rule blocks save time because they allow you to specify a set of attributes for multiple rules.

To construct a rule block, list the attributes first. Second, list any number of rules below the attributes. Enclose the list of rules within a pair of { } characters. Place the opening { character after the attributes and before the first rule. Place the closing } character after the last rule.

(*attribute1=value, attribute2=value, attribute3=value*)
{
rule;
rule;
...
}

The { } characters following the attributes define the scope of the attributes. How you arrange the attributes and rules makes no functional difference as long as you follow the construction principles. Therefore, you could also arrange the attributes and rules this way

---

(
attribute=value,
attribute=value
)
{
rule;
rule;
}

---

or this way

(*attribute=value, attribute=value*)
{ *rule;*
rule; }

Individual Rule Attributes in Rule Blocks

You can specify additional rule attributes for individual rules within rule blocks. When you do, it is important to understand that the `rulename`, `severity`, and `recurse` attributes are not additive. This means that when you assign these attributes to an individual rule in a rule block, the individual attribute value overrides the inherited rule block value.

However, `emailto` attribute **is** additive. When you specify an `emailto` attribute for an individual rule in a rule block, Tripwire for Servers sends e-mail to all recipients.

In the example below, if you specify a minimum `severity` of 90 for an integrity check, Tripwire for Servers uses only rule 4. This is because its `severity` of 100 overrides the inherited rule block `severity` of 80.

However, both Sys Admin and Root receive e-mail when Tripwire for Servers detects the violation of rule 5. This is because the `emailto` attribute is additive.

```
(rulename="wonderland", emailto="Sys Admin", severity=80)
{
rule1 -> $(ReadOnly);
rule2 -> $(IgnoreAll);
rule3 -> $(ReadOnly);
rule4 -> $(ReadOnly)(severity=100);
rule5 -> $(ReadOnly)(emailto=Root);
}
```

Nesting Rule Blocks

You can nest rule blocks within rule blocks. When you do, it is important to remember that `rulename`, `severity`, and `recurse` attributes are not additive. This means all nested `rulename`, `severity`, or `recurse` values override the `rulename`, `severity`, or `recurse` values inherited from a parent rule block.

If you specify the First Block `rulename` for an integrity check, Tripwire for Servers ignores rules in Second Block. This is because Second Block overrides First Block as the `rulename` for the nested rules.

However, both Sysadmin 1 and Sysadmin 2 receive e-mail when Tripwire for Servers detects a violation of rule 3. This is because the `emailto` attribute is additive.

```
(rulename="First Block", emailto="Sysadmin 1")
{
rule1;
rule2;
        (rulename="Second Block", emailto="Sysadmin 2")
        {
        rule3;
        }
}
```

You can nest rule blocks within rule blocks up to sixteen levels.

```
(rulename="First Block")
{
rule;
rule;
            ...
                        (rulename= "Sixteenth Block")
                        {
                        rule;
                        rule;
                        }
}
```

Variables

In constructing a rule, you can use variables to replace object names, properties, and directive arguments. Types of variables include:

predefined	static variable definitions built into Tripwire for Servers for your convenience—you can use these variables anywhere that variables are permitted
user-defined	your own variables defined as needed
global	variables defined in the global variable section—these have scope throughout the policy file
local	variables defined locally in any section other than a global variable section—these have scope only within their particular section

You can use predefined and user-defined variables both globally and locally. If a local variable has the same name as a global variable, the local definition overrides the global definition for that local section only.

Predefined Variables

Predefined variable definitions are built into Tripwire for Servers for your convenience. These predefined variables represent some common property combinations. You cannot change their definitions but you can use them to define your own variables.

Predefined variables are case-sensitive.

U

```
/usr       ->  $(dynamic);  # invalid: predefined variable is case-sensitive
/usr/local ->  $(Dynamic);  # valid
```

W

```
c:\temp     ->  $(dynamic);  # invalid: predefined variable is case-sensitive
c:\programs ->  $(Dynamic);  # valid
```

You cannot use a predefined variable name for another variable.

U

```
ReadOnly=pinugs;              # invalid: ReadOnly is predefined
```

W

```
ReadOnly = &write &size;      # invalid: ReadOnly is predefined
```

You can use the + or - characters to turn properties within a predefined variable on or off in a single instance.

U

```
/etc/dir -> $(Dynamic) +ra;           # you can add properties to a
                                      # predefined variable

MyVariable = $(Dynamic) -ug;          # use a predefined variable to define
/etc/usr ->$(MyVariable);             # your own variable
```

W

```
c:\winnt\files -> $(Dynamic) +&haval;  # you can add properties to a
                                       # predefined variable

MyVariable = $(Dynamic) -&readonly;    # use a predefined variable to define
c:\winnt\profiles -> $(MyVariable);    # your own variable
```

Predefined Variables for UNIX File System

ReadOnly	Use for files that are widely available but which should not be changed. Expands to: `+pinugsmtdbCM -raclSH`
Dynamic	Use for user directories and other files that change frequently. Expands to: `+pinugtd -rsacmblCMSH`
Growing	Use for files expected to grow but that should not shrink. Expands to: `+pinugtdl -rsacmbCMSH`
IgnoreAll	Tracks a file's presence or absence, but does not check any other properties. Expands to: `-pinusgamctdrblCMSH`
IgnoreNone	Turns on all properties. This variable is a good starting point for defining your own variables. Expands to: `+pinusgamctdrbCMSH -l` Always specify `-ar` to avoid erroneous violations when using `IgnoreNone`. This is because the `a` property (access timestamp) conflicts with cryptographic hashes and the `r` property (device number) applies for device objects only. *object name* -> $(IgnoreNone) -ar;
Device	Use for devices or other files that Tripwire software should not attempt to open. Expands to: `+pugsdr -intlbamcCMSH` Because some device numbers may dynamically change, you may need to specify `-r` in some cases when using `Device`. *object name* -> $(Device) -r;

Predefined Variables for Windows File System

ReadOnly	Use for files that are widely available but which should not be changed.
	
	Expands to:
	
	`+ &archive &readonly &offline &hidden &system &directory &write &create &size &owner &group &dacl &sacl &sdc &sdsize &crc32 &md5 &strm_count &strm_crc32 &strm_md5`
	
	`- &temp &msdosname &compressed &access &sha &haval &strm_sha &strm_haval`

Dynamic	Use for user directories and other files that change frequently, but whose properties should remain constant.
	
	Expands to:
	
	`+ &archive &readonly &offline &temp &hidden &system &directory &create &owner &group &dacl &sacl &sdc &sdsize`
	
	`- &size &msdosname &compressed &access &write &sha &haval &md5 &crc32 &strm_count &strm_sha &strm_haval &strm_md5 &strm_crc32`

IgnoreAll	Tracks an object's presence or absence but no other properties. Expands to: `- &archive &readonly &offline &temp &hidden &system &directory &access &write &create &size &msdosname &compressed &owner &group &dacl &sacl &sdc &sdsize &sha &haval &md5 &crc32 &strm_count &strm_sha &strm_haval &strm_md5 &strm_crc32`
IgnoreNone	Use for critical objects. Turns on all properties. This variable is a good starting point for defining your own variables. Expands to: `+ &archive &readonly &offline &temp &hidden &system &directory &access &write &create &size &msdosname &compressed &owner &group &dacl &sacl &sdc &sdsize &sha &haval &md5 &crc32 &strm_count &strm_sha &strm_haval &strm_md5 &strm_crc32` Always specify `-&access` to avoid erroneous violations when using `IgnoreNone`. This is because the `&access` (last access time) property conflicts with cryptographic hashes. *object name* -> (IgnoreNone) -&access;

Predefined Variables for Windows Registry

ReadOnly	Use for registry objects that are widely available but which should not be changed.
	
	Expands to:
	
	`+ &owner &group &dacl &sacl &sdc &classname &nsubkeys &maxsubkeyname &maxclassname &nvalues &maxvaluename &maxdatalen &sdsize &datatype &datalen &crc32 &md5`
	
	`- &write &sha &haval`

Dynamic	Use for registry objects that change frequently, but whose properties should remain constant.
	
	Expands to:
	
	`+ &owner &group &dacl &sacl &classname &datatype &sdc &sdsize`
	
	`- &nsubkeys &maxsubkeyname &maxclassname &nvalues &maxvaluename &maxdatalen &write &datalen &crc32 &md5 &sha &haval`

IgnoreAll	Tracks a registry object's presence or absence, but does not check any other properties. Expands to: `- &owner &group &dacl &sacl &sdc &classname &nsubkeys &maxsubkeyname &maxclassname &nvalues &maxvaluename &maxdatalen &sdsize &write &datatype &datalen &crc32 &md5 &sha &haval`
IgnoreNone	Use for critical objects. Turns on all properties. This variable is a good starting point for defining your own variables. Expands to: `+ &owner &group &dacl &sacl &sdc &classname &nsubkeys &maxsubkeyname &maxclassname &nvalues &maxvaluename &maxdatalen &sdsize &write &datatype &datalen &crc32 &md5 &sha &haval`

Predefined Variables for Windows Registry Values

Variable	Value
HKCR	HKEY_CLASSES_ROOT
HKCU	HKEY_CURRENT_USER
HKLM	HKEY_LOCAL_MACHINE
HKU	HKEY_USERS
HKCC	HKEY_CURRENT_CONFIG
HKPD	HKEY_PERFORMANCE_DATA

User-Defined Variables

Define your own global and local variables as follows:

---

variable = *value;*

The *variable* is the case-sensitive variable name and the *value* is its assigned value. Terminate each variable definition with a ; character. Quote a value when it contains white space or escaped characters.

---

variable = "my value";

Remember that the \ character within quotes introduces the Tripwire for Servers escape sequence. Use another \ character to escape \ delimiters or other restricted characters in a quoted variable value.

```
PROGRAMFILES = c:\"program files";    # valid
SYSTEMDIR = "c:\\winnt\\system32";    # valid
SYSTEMROOT = "c:\winnt";              # invalid; \ delimiter within quotes not
                                      # escaped with preceding \ character
```

You can redefine a user-defined variable at any time.

U

```
# define and use a variable "mask"
mask = pinugs;
/etc -> $(mask);

mask = pinl;
/etc  -> $(mask);
```

W

```
# define and use a variable "mask"
mask = &sdc;
c:\winnt\system32 -> $(mask);

mask = &sdc &haval;
c:\winnt\system32 -> $(mask);
```

You cannot use variables that contain functional tokens in their definition.

```
salesdept = emerald || pearl;
@@ifhost $(salesdept)          # invalid: variable definition contains the ||
                               # functional token
```

Variable Substitution

After defining a variable you can substitute it wherever a string could appear. Begin a variable substitution with the $ character, followed by the variable name within ( ) characters.

U

```
# variable substitution on the left, right, and both sides of a rule

mask1 = pinugs;                          # define variable "mask1"
dir1 = /mnt;                             # define variable "dir1"

$(dir1)/system -> pin;                   # left-hand substitution
/temp -> $(mask1);                       # right-hand substitution
$(dir1) -> $(mask1);                     # double substitution
```

W

```
# variable substitution on the left, right, and both sides of a rule

mask1 = &owner &dacl;                    # define variable "mask1"
dir1 = c:\winnt;                         # define variable "dir1"
key1 = HKEY_LOCAL_MACHINE\SYSTEM;        # define variable "key1"

$(dir1)\system -> &access;               # left-hand substitution
c:\winnt\tmp -> $(mask1);                # right-hand substitution
$(key1) -> $(mask1);                     # double substitution
$(dir1) -> $(mask1);                     # double substitution
```

If a variable represents properties, you can modify the variable definition by turning off properties or adding properties when you substitute it.

U

```
mask1 = pinug;              # define a variable "mask1"
/file1 ->  $(mask1)-g;      # use "mask1", but turn off property "g"
```

W

```
mask1 = &size &access;          # define a variable "mask1"
c:\myfile -> $(mask1)-&size;    # use 'mask1', but turn off property "&size"
```

You can substitute variables for directive arguments.

```
server1 = emerald;
server2 = sapphire;
@@ifhost $(server1)        # apply these rules if server is emerald
     rule1;
     rule2;
     @@ifhost $(server2)   # apply these rules if server is sapphire
          rule1;
     @@endif
@@endif
```

You cannot use variables to represent a literal token or a directive name.

U

```
arrow = ->;
/temp $(arrow) pin;        # invalid: variable cannot represent a token
ifhostnameis = @@ifhost;
$(ifhostnameis) pearl      # invalid: variable cannot represent a directive name
```

W

```
arrow = ->;
c:\temp $(arrow) &size;    # invalid: variable cannot represent a token
ifhostnameis = @@ifhost;
$(ifhostnameis) pearl      # invalid: variable cannot represent a directive name
```

Stop Points

Stop points exclude objects from an integrity check. To specify a stop point, place a ! character before the object name, then terminate the stop point with a ; character.

!*object name;*

- Because Tripwire for Servers skips the object, a stop point does not require properties.
- When you assign a stop point to a directory, Tripwire for Servers skips the entire directory including its contents.
- A stop point object name cannot duplicate another rule's object name.

U

```
/etc/passwd -> p;
!/etc/passwd;          # invalid: stop point object duplicates rule object
```

W

```
c:\winnt -> &size;
!c:\winnt;             # invalid: stop point object duplicates rule object
```

Directives

Directives allow you to specify sections, use conditional logic, and perform some debugging and diagnostic operations in the policy file.

@@*directive name* [*arguments*]

The *directive name* determines the function of the directive and the *arguments* assign a value for that function. When using directives, follow these syntax conventions.

- Directive names and arguments are case-sensitive.
- White space may precede or follow the @@ construct, but no other characters may appear before or between the two @ characters.
- You cannot substitute a variable for a directive name, but you can use variables as arguments to a directive.

```
machine=diamond;
@@ifhost $(machine)       # valid; variable used as a directive argument

IFHOST=ifhost;
@@ $(IFHOST) diamond    # invalid; cannot use variables for directive names
```

These are the directive names you can use in a policy file.

@@section	declares a policy file section
@@ifhost	applies rules and logic to the specified host
@@else	applies rules and logic to any other host
@@endif	ends conditional logic
@@print	prints its text string argument to *stdout*
@@error	prints text string to *stdout* and exits with a 1 status
@@end	logical end of the policy file

Directive Issues

With the exception of `@@print` and `@@error`, you cannot use directives within a rule block. The `@@section`, `@@end`, `@@ifhost`, `@@else`, and `@@endif` directives within a rule block cause a syntax error.

Declaring Sections

The `@@section` directive declares major sections in the policy file. See page 33 for more information about policy file sections.

Conditional Logic

The `@@ifhost`, `@@else`, and `@@endif` directives allow conditional logic. You can use these directives to apply different rules to specific machines using one policy file.

@@ifhost *host1* || *host2* || ...
rule;
...
@@else
rule;
...
@@endif

The `host1, host2, ...` must be case-sensitive unqualified hostnames, and the `||` notation is interpreted as the logical OR operation. There is no `@@elseif` directive.

- All rules and logic between the `@@ifhost` and `@@endif` apply to any hostname matching the `@@ifhost` arguments.
- If you place an `@@else` between `@@ifhost` and `@@endif`, then all rules and logic between `@@else` and `@@endif` apply to every other hostname.

U

```
@@ifhost pearl                          # If pearl,
    /etc -> abcdgimnpstu;               # apply this rule.
@@else                                  # Or else, if any other host,
    /mnt -> abcdgimnpstu;               # apply this rule.
@@endif                                 # (end)
```

W

```
@@ifhost pearl                          # If pearl,
    c:\mydocs -> &size;                 # apply this rule.
@@else                                  # Or else, if any other host,
    c:\mydocs -> &sha;                  # apply this rule.
@@endif                                 # (end)
```

You can use a logical OR operation (| | notation) to apply conditions to multiple hosts.

U

```
@@ifhost ruby || diamond                # If ruby OR diamond,
    /etc/special -> $(IgnoreAll);       # apply this rule.
@@endif                                 # (end)
@@ifhost emerald || sapphire            # If emerald OR sapphire,
    /etc/projects -> $(IgnoreNone) -ar; # apply this rule
@@endif                                 # (end)
```

W

```
@@ifhost ruby || diamond                # If ruby OR diamond,
    c:\special -> &size &write &haval;  # apply this rule.
@@endif                                 # (end)
@@ifhost emerald || sapphire            # If emerald OR sapphire,
    c:\projects -> &sdc &haval;         # apply this rule.
@@endif                                 # (end)
```

Nested Conditional Logic

You can perform specific conditional logic by nesting directives. In this example, Tripwire for Servers first checks for hostname `pearl`. If the hostname is `pearl`, it applies the first rule. If the hostname is not `pearl`, it checks for the hostname `emerald`. If the hostname is `emerald`, Tripwire for Servers ignores all properties for */etc/passwd*. For all other hosts, it fully examines */etc/services*.

U

```
@@ifhost pearl                          # If pearl,
    /etc/passwd -> $(Growing);          # apply this rule.
@@else                                  # Or else,
    @@ifhost emerald                    # if emerald,
        /etc/passwd -> $(IgnoreAll);    # apply this rule.
    @@endif                             # (end)
    /etc/services -> $(IgnoreNone) -ar; # All others, apply this rule.
@@endif                                 # (end)
```

In this example, Tripwire for Servers first checks for hostname `pearl`. If the hostname is `pearl`, it applies the first rule. If the hostname is not `pearl`, it checks for the hostname `emerald`. If the hostname is `emerald`, Tripwire for Servers ignores all properties for *c:\winnt*. For all other hosts, it checks the Dynamic variable's properties for *c:\temp*.

W

```
@@ifhost pearl                          # If pearl,
    c:\program files -> $(Dynamic);     # apply this rule.
@@else                                  # Or else,
    @@ifhost emerald                    # if emerald,
        c:\winnt -> $(IgnoreAll);       # apply this rule.
    @@endif                             # (end)
    c:\temp -> $(Dynamic);              # All others, apply this rule.
@@endif                                 # (end)
```

Debugging and Diagnostics

The `@@print` and `@@error` directives allow debugging and some remote diagnostics. Each argument to these directives must be one string only. You must quote any string containing white spaces.

```
@@print string
@@error string
@@print "two strings"
@@print two strings          # invalid: must quote strings that contain white space
```

When Tripwire for Servers reads the policy file, the `@@print` arguments print to *stdout*. The `@@error` arguments print to *stdout* with the calling program exiting with a status of 1.

U

```
@@ifhost sapphire
    /etc  -> $(Dynamic);
@@else
    @@ifhost amethyst
        @@print "Scanning projects on amethyst"
        /etc/projects  -> pinug;
    @@else
        @@error "This policy file not written for this machine"
    @@endif
@@endif
```

W

```
@@ifhost sapphire
    c:\winnt -> &write &access &haval;
@@else
    @@ifhost amethyst
        @@print "Scanning projects on amethyst"
        D:\projects -> &access &haval;
    @@else
        @@error "This policy file not written for this machine"
    @@endif
@@endif
```

Logical End of the Policy File

The `@@end` directive marks the logical end of a policy file. Tripwire for Servers does not parse past an `@@end` directive.

You can use space below an `@@end` directive to write instructions, comments or any other non-syntax text. You do not need to precede comments below an `@@end` directive with the # character.

Appendices

Appendix A: Windows Security Attributes

Tripwire software can monitor the following security attributes of the Windows operating system. For more information about these attributes, see the Microsoft Software Developers Network (MSDN) website at **http://www.msdn.microsoft.com.**

Discretionary access control list (DACL)—A list that specifies levels of access that users or groups may have to an object. The object's owner controls its DACL. The `&dacl` property monitors the full DACL value for an object. If larger than 2KB, Tripwire software stores an MD5 hash of the DACL value.

System access control list (SACL)—A list that controls the generation of audit log entries for attempts to access a securable object. Typically, only system administrators may set an object's SACL. The `&sacl` property monitors the full SACL value for an object. If larger than 2KB, Tripwire software stores an MD5 hash of the SACL value.

Security identifier (SID)—Unique security identifier for a user or group. The `&owner` and `&group` properties monitor the full values for Owner and Group SIDs.

Security descriptor—A structure that contains security information for a securable object. The security descriptor identifies the object's owner and primary group. It may also contain a DACL that controls access to the object, and a SACL that controls the logging of attempts to access the object. The `&sdsize` property monitors the size of an object's security descriptor.

Security descriptor control (SDC)—A value that represents how components of the security descriptor (SACL, DACL, etc.) were created, and controls inheritance of security information (especially in Windows 2000). The `&sdc` property monitors the full SDC value of an object, a hexadecimal value which is the bitwise-OR of the following components:

```
OWNER_DEFAULTED          (0x0001)
GROUP_DEFAULTED          (0x0002)
DACL_PRESENT             (0x0004)
DACL_DEFAULTED           (0x0008)
SACL_PRESENT             (0x0010)
SACL_DEFAULTED           (0x0020)
DACL_AUTO_INHERIT_REQ    (0x0100)
SACL_AUTO_INHERIT_REQ    (0x0200)
DACL_AUTO_INHERITED      (0x0400)
SACL_AUTO_INHERITED      (0x0800)
DACL_PROTECTED           (0x1000)
SACL_PROTECTED           (0x2000)
SELF_RELATIVE            (0x8000)
```

For more information about these components of an SDC value, see the Microsoft Software Developers Network (MSDN) website at

http://msdn.microsoft.com/library/psdk/winbase/acctrlow_7u5u.htm
http://msdn.microsoft.com/library/psdk/winbase/accctrl_6fqk.htm

Appendix B: Sample Reports

Tripwire for Servers provides five levels of report detail. Each level shows a different subset of the details contained in Tripwire report files.

Use the REPORTLEVEL and EMAILREPORTLEVEL parameters in the configuration file to specify a default report and e-mail report level.

Level 0: Single Line Report

The level 0 report summarizes integrity check results in a single line. This information also serves as the subject line of every Tripwire e-mail report.

```
Tripwire Report SAPPHIRE 20010321134026 V:9 S:100 A:2 R:1 C:6  L:2 M:6 H:1
```

This single line report contains the following information.

- Tripwire Report
- Hostname (SAPPHIRE)
- Date and time of report generation
- V—Total number of violations
- S—Severity of the highest severity violation found
- A, R, C—Number of added, removed, and changed objects
- L, M, H—Number of low, medium, and high severity violations

Level 1: Parsable List of Violations

The level 1 report is a single parsable list of all violated objects. You could use this report format to direct a backup program to restore tampered files automatically, or to automate some other response.

U

```
Note: Report is not encrypted.
Added:   "/home/myname/file1"
Added:   "/home/myname/file2"
Removed: "/home/myname/file3"
Modified:"/home/myname/file4"

Total violations found: 4
```

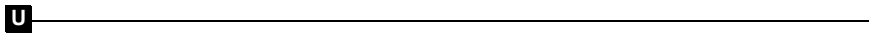
W

```
Note: Report is not encrypted.
Added:  "C:\\WINNT\\System32\\WinNTDL\\pin.txt"
Removed: "C:\\WINNT\\System32\\WinNTDL\\sysfd.txt"
Modified:  "C:\\WINNT\\System32\\WinNTDL"
Modified:  "C:\\WINNT\\System32\\WinNTDL\\fdjk.txt"
Added:  "C:\\PROGRAM FILES\\TRIPWIRE\\bin\\twprint.exe"
Added:  "C:\\PROGRAM FILES\\TRIPWIRE\\bin\\siggen.exe"
Removed:   "C:\\PROGRAM FILES\\TRIPWIRE\\bin\\twadmin.exe"
Modified:  "C:\\PROGRAM FILES\\TRIPWIRE\\bin"
Modified:  "C:\\PROGRAM FILES\\TRIPWIRE\\bin\\twcfg.txt"
Added:  "HKEY_CURRENT_USER\\Software\\Tripwire\\Tripwire"|"TW"
Removed:   "HKEY_CURRENT_USER\\Software\\Tripwire\\Tripwire"|"Test"
Modified:  "HKEY_CURRENT_USER\\Software\\Tripwire\\Tripwire"|"Version"
Added:  "HKEY_LOCAL_MACHINE\\SOFTWARE\\Microsoft\\Rpc"|"Sys"
Removed:   "HKEY_LOCAL_MACHINE\\SOFTWARE\\Microsoft\\Rpc"|"TWRPT"

Total violations found: 13
```

Level 2: Summary Report

The level 2 report lists all violations by policy file section and rule name. The Object Summary section lists added, removed, and modified objects.



```
Note: Report is not encrypted.
Tripwire(R) 2.4.0 Integrity Check Report

Report generated by:          Admin
Report created on:            Mon Mar 12 13:34:05 2001
Database last updated on:     Mon Mar 12 10:00:05 2001

================================================================================
Report Summary:
================================================================================
Host name:                    EMERALD
Host IP address:              198.6.100.43
Host ID:                      450a0d0a
Policy file used:             /usr/local/tripwire/policy/tw.pol
Configuration file used:      /usr/local/tripwire/bin/tw.cfg
Database file used:           /usr/local/tripwire/db/test.twd
Command line used:            ./tripwire -m c

================================================================================
Rule Summary:
================================================================================

--------------------------------------------------------------------------------
  Section: Unix File System
--------------------------------------------------------------------------------

   Rule Name                  Severity Level    Added    Removed    Modified
   ----------                 --------------    -----    -------    --------
* My Home                    100               2        1          1

Total objects scanned:  215
Total violations found:  4

================================================================================
Object Summary:
================================================================================

--------------------------------------------------------------------------------
 Section: Unix File System
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------
Rule Name: My Home (/home/myname)
Severity Level: 100
--------------------------------------------------------------------------------

Added:
"/home/myname/file3"
"/home/myname/file4"
```

```
Removed:
"/home/myname/file2"

Modified:
"/home/myname/file1"

================================================================================
Error Report:
================================================================================

No Errors

--------------------------------------------------------------------------------
*** End of report ***

Report generated by:
Tripwire(R) for Servers v (2.4.0.104)

Tripwire is a registered trademark of Tripwire, Inc.
All rights reserved.
```

```
Note: Report is not encrypted.
Tripwire(R) 2.4.0 Integrity Check Report

Report generated by:       Admin
Report created on:         Wednesday, March 14, 2001 4:50:05 PM
Database last updated on:  Wednesday, March 14, 2001 4:50:05 PM

================================================================================
Report Summary:
================================================================================
Host name:                 DIAMOND
Host IP address:           10.100.1.35
Policy file used:          C:\Program Files\Tripwire\policy\test.pol
Configuration file used:   C:\Program Files\Tripwire\bin\tw.cfg
Database file used:        C:\Program Files\Tripwire\db\test.twd
Command line used:         tripwire -m c -v

================================================================================
Rule Summary:
================================================================================

--------------------------------------------------------------------------------
  Section: Windows File System
--------------------------------------------------------------------------------

 Rule Name                 Severity Level  Added    Removed   Modified
 ---------                 --------------  ------   -------   --------
* Critical Startup files   100             1        1         1
* Tripwire                 60              1        1         1
(C:\PROGRAM FILES\TRIPWIRE\bin)

Total objects scanned:  94
Total violations found:  6

--------------------------------------------------------------------------------
  Section: Windows Registry
--------------------------------------------------------------------------------
 Rule Name                 Severity Level  Added    Removed   Modified
 ---------                 --------------  ------   -------   --------
* Tripwire Reg. Keys       100             1        1         1
  (HKEY_CURRENT_USER\Software\Tripwire)

Total objects scanned:  109
Total violations found:  3

================================================================================
Object Summary:
================================================================================

--------------------------------------------------------------------------------
# Section: Windows File System
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Rule Name: Critical Startup files (C:\WINNT\System32\WinNTDL)
Severity Level: 100
--------------------------------------------------------------------------------

Added:
"C:\\WINNT\\System32\\WinNTDL\\pin.txt"
```

```
Removed:
"C:\\WINNT\\System32\\WinNTDL\\sysfd.txt"

Modified:
"C:\\WINNT\\System32\\WinNTDL"

-------------------------------------------------------------------------------
Rule Name: Tripwire (C:\PROGRAM FILES\TRIPWIRE\bin)
Severity Level: 60
-------------------------------------------------------------------------------

Added:
"C:\\PROGRAM FILES\\TRIPWIRE\\bin\\twprint.exe"

Removed:
"C:\\PROGRAM FILES\\TRIPWIRE\\bin\\twadmin.exe"

Modified:
"C:\\PROGRAM FILES\\TRIPWIRE\\bin"

-------------------------------------------------------------------------------
# Section: Windows Registry
-------------------------------------------------------------------------------

-------------------------------------------------------------------------------
Rule Name: Tripwire Reg. keys (HKEY_CURRENT_USER\Software\Tripwire)
Severity Level: 100
-------------------------------------------------------------------------------

Added:
"HKEY_CURRENT_USER\\Software\\Tripwire\\Tripwire"|"TW"

Removed:
"HKEY_CURRENT_USER\\Software\\Tripwire\\Tripwire"|"Test"

Modified:
"HKEY_CURRENT_USER\\Software\\Tripwire\\Tripwire"|"Version"

===============================================================================
Error Report:
===============================================================================

-------------------------------------------------------------------------------
  Section: Windows File System
-------------------------------------------------------------------------------

1.   Win32 API failure.
     GetFileAttributesEx failed for C:\Testdir\twcrit.dll:
     The system cannot find the file specified.

-------------------------------------------------------------------------------
*** End of report ***

Report generated by:
Tripwire(R) for Servers v (2.4.0.104)

Tripwire is a registered trademark of Tripwire, Inc.
All rights reserved.
```

Level 3: Concise Report

The level 3 report contains the same information as a level 2 Summary Report, with the addition of compared expected and observed values for modified objects only. There are no additional details shown for added or removed objects. For full details of added, removed, and modified objects see the Level 4 report (page 91).

In the Object Detail section, the * character indicates a changed property.

U

```
Note: Report is not encrypted.
Tripwire(R) 2.4.0 Integrity Check Report

Report generated by:       Admin
Report created on:         Fri Mar 9 11:58:41 2001
Database last updated on:  Never

================================================================================
Report Summary:
================================================================================

Host name:                 EMERALD
Host IP address:           100.66.22.68
Host ID:                   430a0d9a
Policy file used:          /usr/local/tripwire/tfs/policy/tw.pol
Configuration file used:   /usr/local/tripwire/tfs//bin/tw.cfg
Database file used:        /usr/local/tripwire/tfs/db/emerald.twd
Command line used:         ./tripwire -m c

================================================================================
Rule Summary:
================================================================================

--------------------------------------------------------------------------------
  Section: Unix File System
--------------------------------------------------------------------------------

  Rule Name                Severity Level    Added    Removed  Modified
  ---------                --------------    -----    -------  --------
* tmp                      0                 1        1        6
  (/tmp)

Total objects scanned:     83
Total violations found:    8

================================================================================
Object Detail:
================================================================================

--------------------------------------------------------------------------------
  Section: Unix File System
--------------------------------------------------------------------------------
```

```
--------------------------------------------------------------------------------
Rule Name: tmp (/tmp)
Severity Level: 0
--------------------------------------------------------------------------------
  ---------------------------------------
  Added Objects: 1
  ---------------------------------------
Added object name:  /tmp/install.gol

  ---------------------------------------
  Removed Objects: 1
  ---------------------------------------
Removed object name:  /tmp/install.log

  ---------------------------------------
  Modified Objects: 6
  ---------------------------------------
Modified object name: /tmp
    Access Time Expected   Fri Mar  9 11:57:24 2001
*               Observed   Fri Mar 9 11:58:27 2001

    Modify Time Expected   Fri Mar  9 11:58:12 2001
*               Observed   Fri Mar  9 11:58:42 2001

    Change Time Expected   Fri Mar  9 11:58:12 2001
*               Observed   Fri Mar  9 11:58:42 2001

Modified object name: /tmp/.ICE-unix
    Access Time Expected   Fri Mar  9 11:57:24 2001
*               Observed   Fri Mar  9 11:58:12 2001

Modified object name: /tmp/.X0-lock
    Access Time Expected   Fri Mar  9 11:57:24 2001
*               Observed   Fri Mar  9 11:58:12 2001

Modified object name: /tmp/.X11-unix
    Access Time Expected   Fri Mar 9 11:57:24 2001
*               Observed   Fri Mar 9 11:58:12 2001

Modified object name: /tmp/.Xauth07TLxu
    Access Time Expected   Fri Mar  9 11:57:24 2001
*               Observed   Fri Mar  9 11:58:12 2001

Modified object name: /tmp/.Xauth5ZPP6I
    Access Time Expected   Fri Mar  9 11:57:24 2001
*               Observed   Fri Mar  9 11:58:12 2001

================================================================================
Error Report:
================================================================================

No Errors

--------------------------------------------------------------------------------
*** End of report ***

Report generated by:
Tripwire for Servers version 2.4.0.146 for Linux Operating Systems

Tripwire is a registered trademark of Tripwire, Inc.
All rights reserved.
```

```
Note: Report is not encrypted.
Tripwire(R) 2.4.0 Integrity Check Report

Report generated by:       Admin
Report created on:         Friday, March 09, 2001 11:28:50 AM
Database last updated on:  Never

================================================================================
Report Summary:
================================================================================

Host name:                 DIAMOND
Host IP address:           10.168.2.22
Host ID:                   410a0d7a
Policy file used:          C:\Program Files\tripwire\tfs\policy\tw.pol
Configuration file used:   C:\Program Files\tripwire\tfs\bin\tw.cfg
Database file used:        C:\Program Files\tripwire\tfs\db\DIAMOND.twd
Command line used:         tripwire -m c

================================================================================
Rule Summary:
================================================================================

--------------------------------------------------------------------------------
  Section: Windows File System
--------------------------------------------------------------------------------

  Rule Name                Severity Level    Added    Removed  Modified
  ---------                --------------    -----    -------  --------
* test                     0                 1        0        1
  (C:\test)

Total objects scanned:     8
Total violations found:    2

--------------------------------------------------------------------------------
  Section: Windows Registry
--------------------------------------------------------------------------------

 Rule Name                 Severity Level    Added    Removed  Modified
  ---------                --------------    -----    -------  --------
* Tripwire                 0                 1        1        1
  (HKEY_LOCAL_MACHINE\SOFTWARE\Tripwire)

Total objects scanned:     3
Total violations found :   3

================================================================================
Object Detail:
================================================================================

--------------------------------------------------------------------------------
  Section: Windows File System
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------
Rule Name: test (C:\test)
Severity Level: 0
--------------------------------------------------------------------------------
```

```
  ----------------------------------------
  Added Objects: 1
  ----------------------------------------

Added object name:  C:\test\Cmmssetup.exe

  ----------------------------------------
  Modified Objects: 1
  ----------------------------------------

Modified object name: C:\test
    Access Time Expected   Friday, March 09, 2001 11:14:25 AM
*               Observed   Friday, March 09, 2001 11:28:30 AM

     Write Time Expected   Tuesday, February 06, 2001 12:03:11 PM
*               Observed   Friday, March 09, 2001 11:27:54 AM


--------------------------------------------------------------------------------
  Section: Windows Registry
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------
Rule Name: Tripwire (HKEY_LOCAL_MACHINE\SOFTWARE\Tripwire)
Severity Level: 0
--------------------------------------------------------------------------------

  ----------------------------------------
  Added Objects: 1
  ----------------------------------------
Added object name:  HKEY_LOCAL_MACHINE\SOFTWARE\Tripwire 2.4.0

  ----------------------------------------
  Removed Objects: 1
  ----------------------------------------
Removed object name:  HKEY_LOCAL_MACHINE\SOFTWARE\Tripwire Security

  ----------------------------------------
  Modified Objects: 1
  ----------------------------------------
Modified object name:  HKEY_LOCAL_MACHINE\SOFTWARE\Tripwire
         Number of Subkeys
                Expected   0
*               Observed   1

================================================================================
Error Report:
================================================================================

No Errors

--------------------------------------------------------------------------------
*** End of report ***

Report generated by:
Tripwire(R) for Servers version 2.4.0.146 for
Windows NT(R)/2000 Operating Systems

Tripwire is a registered trademark of Tripwire, Inc. All rights reserved.
```

Level 4: Full Report

The level 4 report provides the most detail of all report levels. This report level shows all observed property values for modified, added, and removed objects. In the Object Detail section, the * character indicates a changed property.

In Windows, this Object Detail section of this report level shows security descriptor control (SDC) detail. This SDC information is not available in any other report level.

U

```
Note: Report is not encrypted.
Tripwire(R) 2.4.0 Integrity Check Report

Report generated by:       Admin
Report created on:         Fri Mar 9 11:58:41 2001
Database last updated on:  Never

================================================================================
Report Summary:
================================================================================

Host name:                 SAPPHIRE
Host IP address:           10.196.8.62
Host ID:                   420a120a
Policy file used:          /usr/local/tripwire/tfs/policy/tw.pol
Configuration file used:   /usr/local/tripwire/tfs/bin/tw.cfg
Database file used:        /usr/local/tripwire/tfs/db/sapphire.twd
Command line used:         ./tripwire -m c

================================================================================
Rule Summary:
================================================================================

--------------------------------------------------------------------------------
  Section: Unix File System
--------------------------------------------------------------------------------

  Rule Name                Severity Level    Added    Removed  Modified
  ---------                --------------    -----    -------  --------
* tmp                      0                 1        0         1
  (/tmp)

Total objects scanned:     83
Total violations found:    2

================================================================================
Object Summary:
================================================================================

--------------------------------------------------------------------------------
# Section: Unix File System
--------------------------------------------------------------------------------
```

```
--------------------------------------------------------------------------------
Rule Name: tmp (/tmp)
Severity Level: 0
--------------------------------------------------------------------------------
Added:
"/tmp/install.gol"

Modified:
"/tmp"

================================================================================
Object Detail:
================================================================================
--------------------------------------------------------------------------------
  Section: Unix File System
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Rule Name: tmp (/tmp)
Severity Level: 0
--------------------------------------------------------------------------------
  ----------------------------------------
  Added Objects: 1
  ----------------------------------------

Added object name:  /tmp/install.gol

    Object Type Expected   ---
*               Observed   Regular File
  Device Number Expected   ---
*               Observed   8454
      File Device Number
                Expected   ---
*               Observed   0
   Inode Number Expected   ---
*               Observed   180226
          Mode Expected   ---
*               Observed   -rw-r--r--
Number of Links Expected   ---
*               Observed   1
            UID Expected   ---
*               Observed   Admin (0)
            GID Expected   ---
*               Observed   Admin (0)
    Access Time Expected   ---
*               Observed   Fri Mar  9 11:58:12 2001
    Modify Time Expected   ---
*               Observed   Wed Feb 21 10:40:53 2001
    Change Time Expected   ---
*               Observed   Fri Mar  9 11:58:37 2001
         Blocks Expected   ---
*               Observed   40
          CRC32 Expected   ---
*               Observed   Cr6drQ
            MD5 Expected   ---
*               Observed   CAi7kDIgvhYrlD/cj9avaq
            SHA Expected   ---
*               Observed   H9pjE7Rdkd6YlFbImYS2xIsvugi
          HAVAL Expected   ---
*               Observed   CWGeshX/AtUM/Smo+N98jh
```

```
  ---------------------------------------
  Modified Objects: 1
  ---------------------------------------

Modified object name:  /tmp

    Object Type Expected   Directory
                Observed   Directory
  Device Number Expected   8454
                Observed   8454
      File Device Number
                Expected   0
                Observed   0
   Inode Number Expected   180225
                Observed   180225
           Mode Expected   drwxrwxrwt
                Observed   drwxrwxrwt
Number of Links Expected   14
                Observed   14
            UID Expected   Admin (0)
                Observed   Admin (0)
            GID Expected   Admin (0)
                Observed   Admin (0)
           Size Expected   4096
                Observed   4096
    Access Time Expected   Fri Mar 9 11:57:24 2001
*               Observed   Fri Mar 9 11:58:27 2001
    Modify Time Expected   Fri Mar 9 11:58:12 2001
*               Observed   Fri Mar 9 11:58:42 2001
    Change Time Expected   Fri Mar 9 11:58:12 2001
*               Observed   Fri Mar 9 11:58:42 2001
         Blocks Expected   8
                Observed   8


================================================================================
Error Report:
================================================================================

No Errors

--------------------------------------------------------------------------------
*** End of report ***

Report generated by:
Tripwire for Servers version 2.4.0.146 for Linux Operating Systems

Tripwire is a registered trademark of Tripwire, Inc. All rights reserved.
```

```
Note: Report is not encrypted.
Tripwire(R) 2.4.0 Integrity Check Report

Report generated by:       Admin
Report created on:         Friday, March 09, 2001 11:28:50 AM
Database last updated on:  Never
================================================================================
Report Summary:
================================================================================

Host name:                 AMETHYST
Host IP address:           10.186.2.06
Host ID:                   410a0d4a
Policy file used:          C:\Program Files\tripwire\tfs\tw.pol
Configuration file used:   C:\Program Files\tripwire\tfs\tw.cfg
Database file used:        C:\Program Files\tripwire\tfs\db\AMETHYST.twd
Command line used:         tripwire -m c

================================================================================
Rule Summary:
================================================================================

--------------------------------------------------------------------------------
  Section: Windows File System
--------------------------------------------------------------------------------
  Rule Name                Severity Level    Added    Removed  Modified
  ---------                --------------    -----    -------  --------
* test                     0                 0        1        1
  (C:\test)

Total objects scanned:     8
Total violations found:    2

--------------------------------------------------------------------------------
  Section: Windows Registry
--------------------------------------------------------------------------------
  Rule Name                Severity Level    Added    Removed  Modified
  ---------                --------------    -----    -------  --------
* Tripwire                 0                 1        1        0
  (HKEY_LOCAL_MACHINE\SOFTWARE\Tripwire)

Total objects scanned:     4
Total violations found:    2

================================================================================
Object Summary:
================================================================================

--------------------------------------------------------------------------------
# Section: Windows File System
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------
Rule Name: test (C:\test)
Severity Level: 0
--------------------------------------------------------------------------------

Removed:
"C:\\test\\Copy of mmssetup.exe"
```

```
Modified:
"C:\\test"

--------------------------------------------------------------------------------
# Section: Windows Registry
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------
Rule Name: Tripwire (HKEY_LOCAL_MACHINE\SOFTWARE\Tripwire)
Severity Level: 0
--------------------------------------------------------------------------------
Added:
"HKEY_LOCAL_MACHINE\\SOFTWARE\\Tripwire\\2.4.0"

Modified:
"HKEY_LOCAL_MACHINE\\SOFTWARE\\Tripwire"

================================================================================
Object Detail:
================================================================================

--------------------------------------------------------------------------------
  Section: Windows File System
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------
Rule Name: test (C:\test)
Severity Level: 0
--------------------------------------------------------------------------------

  ----------------------------------------
  Removed Objects: 1
  ----------------------------------------

Removed object name:  C:\test\Copy of mmssetup.exe

    Object Type Expected   File
*               Observed   ---

 Directory Flag Expected   0
*               Observed   ---

 Read Only Flag Expected   0
*               Observed   ---

    Hidden Flag Expected   0
*               Observed   ---

    System Flag Expected   0
*               Observed   ---

   Archive Flag Expected   1
*               Observed   ---

Compressed Flag Expected   0
*               Observed   ---

   Offline Flag Expected   0
*               Observed   ---

 Temporary Flag Expected   0
*               Observed   ---
```

```
           Size Expected   770,320
*               Observed   ---

    MS-DOS Name Expected   COPYOF~1.EXE
*               Observed   ---

        SD Size Expected   92
*               Observed   ---

     SD Control Expected   Value: 0x8404
                             ( -Owner Default -Group Default +Self Relative
                           DACL:           +Present     -Auto Inhrt Request
                            -Protected     -Defaulted    +Auto Inherited
                            SACL:           -Present       -Auto Inhrt Request
                              -Protected     -Defaulted    -Auto Inherited   )
*    SD Control Observed   ---

            SHA Expected   O1RSyIdN1yExdlXLhRsO8XPG1Dm
*               Observed   ---

          HAVAL Expected   A4WJOjJyI4NniycNCGZIJe
*               Observed   ---

            MD5 Expected   B6z+LWEOUK3Q6iPyRnTIWq
*               Observed   ---

          CRC32 Expected   BJl5e9
*               Observed   ---

    Access Time Expected   Tuesday, February 06, 2001 12:05:26 PM
*               Observed   ---

     Write Time Expected   Wednesday, December 06, 2000 8:06:23 AM
*               Observed   ---

    Create Time Expected   Wednesday, December 06, 2000 3:40:31 PM
*               Observed   ---

          Owner Expected   BUILTIN\Administrators
                           (S-2-5-32-522)
*         Owner Observed   ---

          Group Expected   Domain Users
                           (S-1-3-21-1706602236-120039981-200888474211-491)
*         Group Observed   ---

           DACL Expected   Revision 2, Size: 28, Number of ACEs: 1
                           Allow: Everyone
                             Mask Value: 0x001F01FF
                             ( +Delete         +Read Control    +Write DACL
                               +Write Owner    +Synchronize     -Access SACL
                               -Max Allowed    -Generic Read    -Generic Write
                               -Generic Exec   -Generic All                 )
                             Flag Value: 0x0010
                             ( -Obj. Inherit   -Cont. Inherit   -No Propagate
                               -Inherit Only   +Inherited       -Failed Access
                               -Successful Access                           )
*          DACL Observed   ---
```

```
  ----------------------------------------
  Modified Objects: 1
  ----------------------------------------

Modified object name: C:\test
    Object Type Expected   Directory
                Observed   Directory

 Directory Flag Expected   1
                Observed   1

 Read Only Flag Expected   0
                Observed   0

    Hidden Flag Expected   0
                Observed   0

    System Flag Expected   0
                Observed   0

   Archive Flag Expected   1
                Observed   1

Compressed Flag Expected   0
                Observed   0

   Offline Flag Expected   0
                Observed   0

 Temporary Flag Expected   0
                Observed   0

    MS-DOS Name Expected   test
                Observed   test

        SD Size Expected   92
                Observed   92

     SD Control Expected   Value: 0x8404
                            (-Owner Default -Group Default +Self Relative
                           DACL:            +Present      -Auto Inhrt Request
                            -Protected   -Defaulted     +Auto Inherited
                           SACL:            -Present      -Auto Inhrt Request
                              -Protected    -Defaulted     -Auto Inherited   )
     SD Control Observed   Value: 0x8404
                            ( -Owner Default -Group Default +Self Relative
                           DACL:            +Present      -Auto Inhrt Request
                            -Protected   -Defaulted     +Auto Inherited
                           SACL:            -Present      -Auto Inhrt Request
                             -Protected    -Defaulted     -Auto Inherited   )

   Number of Alt Streams
                Expected   4
                Observed   4

     Stream SHA Expected   ID2O8STHXgkpoN8MTrO9DE2a2o5
                Observed   ID2O8STHXgkpoN8MTrO9DE2a2o5

           Stream HAVAL    Expected A0K4pN3kqVxBOiIoJ5Vuwg
                Observed   A0K4pN3kqVxBOiIoJ5Vuwg

             Stream MD5    Expected Dsil3EuoJrvEZ7DIZO4lh6
                Observed   Dsil3EuoJrvEZ7DIZO4lh6
```

```
            Stream CRC32   Expected  B3Sx08
                Observed   B3Sx08

    Access Time Expected   Friday, March 09, 2001 11:14:25 AM
*               Observed   Friday, March 09, 2001 11:28:30 AM

     Write Time Expected   Tuesday, February 06, 2001 12:03:11 PM
*               Observed   Friday, March 09, 2001 11:27:54 AM

           DACL Expected    Revision 2, Size: 28, Number of ACEs: 1
                            Allow: Everyone
                             Mask Value: 0x001F01FF
                             ( +Delete        +Read Control    +Write DACL
                               +Write Owner  +Synchronize     -Access SACL
                              -Max Allowed  -Generic Read    -Generic Write
                              -Generic Exec -Generic All                 )
                            Flag Value: 0x0013
                           ( +Obj. Inherit  +Cont. Inherit  -No Propagate
                            -Inherit Only   +Inherited      -Failed Access
                                 -Successful Access                       )
              DACL Observed  Revision 2, Size: 28, Number of ACEs: 1
                            Allow: Everyone
                             Mask Value: 0x001F01FF
                             ( +Delete       +Read Control   +Write DACL
                               +Write Owner  +Synchronize    -Access SACL
                               -Max Allowed  -Generic Read   -Generic Write
                               -Generic Exec -Generic All                )
                              Flag Value: 0x0013
                               ( +Obj. Inherit  +Cont. Inherit  -No Propagate
                                 -Inherit Only  +Inherited      -Failed Access
                                 -Successful Access                        )


--------------------------------------------------------------------------------
  Section: Windows Registry
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------
Rule Name: Tripwire (HKEY_LOCAL_MACHINE\SOFTWARE\Tripwire)
Severity Level: 0
--------------------------------------------------------------------------------
  ----------------------------------------
  Added Objects: 1
  ----------------------------------------

Added object name:  HKEY_LOCAL_MACHINE\SOFTWARE\Tripwire

    Object Type Expected   ---
*               Observed   Key
          Class Expected   ---
*               Observed   ""

       Number of Subkeys
                Expected   ---
*               Observed   0

  Max Subkey Name Length
                Expected   ---
*               Observed   0

   Max Class Name Length
                Expected   ---
*               Observed   0
```

```
          Number of Values
                Expected   ---
*               Observed   0

   Max Value Name Length
                Expected   ---
*               Observed   0

        SD Size Expected   ---
*               Observed   276

     SD Control Expected   ---
*               Observed   Value: 0x8404
                           (-Owner Default-Group Default +Self Relative
                           DACL:          +Present       -Auto Inhrt Request
                             -Protected -Defaulted     +Auto Inherited
                           SACL:          -Present       -Auto Inhrt Request
                              -Protected -Defaulted     -Auto Inherited   )

          Owner Expected   ---
*         Owner Observed   BUILTIN\Administrators
                           (S-1-2-32-577)

          Group Expected   ---
*         Group Observed   Domain Users
                           (S-1-3-21-1706602236-120039981-200888474211-491)

     Write Time Expected   ---
*               Observed   Friday, March 09, 2001 11:28:47 AM

           DACL Expected   ---
*          DACL Observed   Revision 2, Size: 212, Number of ACEs: 9
                           Allow: BUILTIN\Users
                               Mask Value: 0x00020019
                              ( -Delete         +Read Control   -Write DACL
                                -Write Owner    -Synchronize    -Access SACL
                                -Max Allowed    -Generic Read   -Generic Write
                                -Generic Exec   -Generic All               )
                               Flag Value: 0x0010
                             ( -Obj. Inherit   -Cont. Inherit  -No Propagate
                               -Inherit Only   +Inherited      -Failed Access
                                -Successful Access                         )
                            Allow: BUILTIN\Users
                               Mask Value: 0x80000000
                              ( -Delete         -Read Control   -Write DACL
                                -Write Owner    -Synchronize    -Access SACL
                                -Max Allowed    +Generic Read   -Generic Write
                                -Generic Exec   -Generic All               )
                               Flag Value: 0x001A
                             ( -Obj. Inherit   +Cont. Inherit  -No Propagate
                               +Inherit Only   +Inherited      -Failed Access
                                -Successful Access                         )
                            Allow: BUILTIN\Power Users
                               Mask Value: 0x0003001F
                              ( +Delete         +Read Control   -Write DACL
                                -Write Owner    -Synchronize    -Access SACL
                                -Max Allowed    -Generic Read   -Generic Write
                                -Generic Exec   -Generic All               )
                               Flag Value: 0x0010
                             ( -Obj. Inherit   -Cont. Inherit  -No Propagate
                               -Inherit Only   +Inherited      -Failed Access
                                -Successful Access                         )
                            Allow: BUILTIN\Power Users
```

```
    Mask Value: 0xC0010000
   ( +Delete          -Read Control   -Write DACL
     -Write Owner     -Synchronize    -Access SACL
     -Max Allowed     +Generic Read   +Generic Write
     -Generic Exec    -Generic All                )
    Flag Value: 0x001A
   ( -Obj. Inherit    +Cont. Inherit  -No Propagate
     +Inherit Only    +Inherited      -Failed Access
     -Successful Access                           )
Allow: BUILTIN\Administrators
    Mask Value: 0x000F003F
   ( +Delete          +Read Control   +Write DACL
     +Write Owner     -Synchronize    -Access SACL
     -Max Allowed     -Generic Read   -Generic Write
     -Generic Exec    -Generic All                )
    Flag Value: 0x0010
   ( -Obj. Inherit    -Cont. Inherit  -No Propagate
     -Inherit Only    +Inherited      -Failed Access
     -Successful Access                           )
Allow: BUILTIN\Administrators
    Mask Value: 0x10000000
   ( -Delete          -Read Control   -Write DACL
     -Write Owner     -Synchronize    -Access SACL
     -Max Allowed     -Generic Read   -Generic Write
     -Generic Exec    +Generic All                )
    Flag Value: 0x001A
   ( -Obj. Inherit    +Cont. Inherit  -No Propagate
     +Inherit Only    +Inherited      -Failed Access
     -Successful Access                           )
Allow: NT AUTHORITY\SYSTEM
    Mask Value: 0x000F003F
   ( +Delete          +Read Control   +Write DACL
     +Write Owner     -Synchronize    -Access SACL
     -Max Allowed     -Generic Read   -Generic Write
     -Generic Exec    -Generic All                )
    Flag Value: 0x0010
   ( -Obj. Inherit    -Cont. Inherit  -No Propagate
     -Inherit Only    +Inherited      -Failed Access
     -Successful Access                           )
Allow: NT AUTHORITY\SYSTEM
    Mask Value: 0x10000000
   ( -Delete          -Read Control   -Write DACL
     -Write Owner     -Synchronize    -Access SACL
     -Max Allowed     -Generic Read   -Generic Write
     -Generic Exec    +Generic All                )
    Flag Value: 0x001A
   ( -Obj. Inherit    +Cont. Inherit  -No Propagate
     +Inherit Only    +Inherited      -Failed Access
     -Successful Access                           )
Allow: CREATOR OWNER
    Mask Value: 0x10000000
   ( -Delete          -Read Control   -Write DACL
     -Write Owner     -Synchronize    -Access SACL
     -Max Allowed     -Generic Read   -Generic Write
     -Generic Exec    +Generic All                )
    Flag Value: 0x001A
   ( -Obj. Inherit    +Cont. Inherit  -No Propagate
     +Inherit Only    +Inherited      -Failed Access
     -Successful Access                           )
```

```
  ----------------------------------------
  Removed Objects: 1
  ----------------------------------------

Removed object name:  HKEY_LOCAL_MACHINE\SOFTWARE\Tripwire Security
    Object Type Expected   Key
*               Observed   ---

          Class Expected   ""
*               Observed   ---

       Number of Subkeys
                Expected   0
*               Observed   ---

   Max Value Data Length
                Expected   0
*               Observed   ---

        SD Size Expected   276
*               Observed   ---

     SD Control Expected   Value: 0x8404
                          (-Owner Default  -Group Default +Self Relative
                           DACL:             +Present       -Auto Inhrt Request
                              -Protected  -Defaulted    +Auto Inherited
                           SACL:             -Present       -Auto Inhrt Request
                               -Protected   -Defaulted     -Auto Inherited   )
*               Observed   ---

          Owner Expected   BUILTIN\Administrators
                           (S-1-5-35-522)
*         Owner Observed   ---

          Group Expected   Domain Users
                           (S-1-3-21-1706602236-120039981-200888474211-491)
*         Group Observed   ---

     Write Time Expected   Wednesday, December 06, 2000 4:07:05 PM
*               Observed   ---


================================================================================
Error Report:
================================================================================

No Errors

--------------------------------------------------------------------------------
*** End of report ***

Report generated by:
Tripwire(R) for Servers version 2.4.0.146
for Windows NT(R)/2000 Operating Systems

Tripwire is a registered trademark of Tripwire, Inc. All rights reserved.
```

Appendix C: Viewing Exit Codes

To view exit codes, enter

U

```
echo $?
```

or

W

```
echo %ERRORLEVEL%
```

on the command line immediately after running a Tripwire command.

Integrity Checking Mode

The return value for integrity check exit codes shows whether the integrity check was successful, and what types of violations the integrity check detected.

Exit Code	Meaning
0-7	Success Any of the following values may be combined to produce a return result for integrity checking: 1 - files added 2 - files removed 4 - files changed
Greater than 7	Failure 8 - an error prevented the report file from being written **Note:** A return value less than 8 does not always mean errors did not occur, or that integrity checking actually occured. The report file itself could contain error messages (if objects specified in the policy file could not be accessed, or did not exist on the system, etc.).

Glossary

asymmetric cryptography

A type of cryptographic system that uses public and private keys, for encryption and decryption of information.

attribute

In the policy file, attributes modify the behavior of policy file rules. Attributes allow you to associate a name or numberic severity level to a rule, or to send e-mail if the rule is violated.

checksum

A value computed, via some parity or hashing algorithm, for information that requires protection against error or manipulation. Checksums are stored or transmitted with data and are intended to detect data integrity problems.

configuration file

A Tripwire file that stores information and settings, including the paths to files, and default settings for integrity checks and other operations. The configuration file is encoded and signed with the site key file, and you must specify the site passphrase to change this file.

CRC-32 algorithm

A Cyclic Redundancy Check algorithm. This is a fast, robust algorithm that detects data transmission errors reliably. CRC-32 is well understood and consequently is a fast, but insecure, alternative to the slower message-digest algorithms. CRC-32 generates a 32-bit signature.

create configuration file mode

A `twadmin` command that signs a plain text file and saves it as the Tripwire configuration file.

create policy file mode

A `twadmin` command that signs a plain text file and saves it as the Tripwire policy file.

damage assessment and recovery

The process of determining the extent and severity of damage after an intrusion. Tripwire integrity systems allow you to quickly see what has changed, and sort the changes based on importance or functional characteristics. This saves time and recovery resources.

database file

A Tripwire file representing a snapshot of a system that serves as the baseline for integrity checks. The database file is used for most Tripwire operations, and should be created from a system in a known secure state. The database file is encoded and signed with the local key file, and you must specify the local key file to update it.

database initialization mode

A `tripwire` command that uses the rules in the current policy file to generate the Tripwire database file.

database update mode

A `tripwire` command that updates the objects in the Tripwire database file with the data from a report file.

directive

In the policy file, a language element that begins with `@@` and defines a section (`@@section`), applies policy rules conditionally (`@@ifhost`, `@@ifelse`, and `@@endif`), or marks the logical end of the file (`@@end`).

encryption mode

A `twadmin` command that signs Tripwire files using the site or local key.

escape sequence

A character sequence that introduces a special-case interpretation of functional characters or sequences. Escape sequences can also be used to represent nonprintable characters.

examine encryption mode

A `twadmin` command that examines Tripwire files and displays the filename, file type, whether the file is signed, and what key, if any, was used to sign it.

generate keys mode

A `twadmin` command that creates site or local keys for Tripwire files.

global variable

A variable you define in the `@@GLOBAL` section of the policy file and use in any section that follows. If a local variable and a global variable have the same name, the local section uses the local variable definition.

hash

The value that a hash algorithm calculates. A simple hash is sometimes called a checksum, and a one-way hash is sometimes called a message digest.

HAVAL algorithm

A one-way hash algorithm for high security. It was written by Yuliang Zheng at the University of Wollongong and is described in the following document:

> Zheng, Y., Pieprzyk, J. and Seberry, J. (1993), "HAVAL: a one-way hashing algorithm with variable length of output" in Advances in Cryptology: AUSCRPT'92, Lecture Notes in Computer Science, Springer-Verlag.

As shipped with Tripwire for Servers, HAVAL is configured with a 128-bit signature using four passes to ensure pseudo-random output.

host-based intrusion detection

Strategy of collecting information about changes to machines to detect intrusions or policy violations.

integrity check

A Tripwire for Servers operation that compares the last known properties of a system object to the current properties to see if there are changes.

integrity check mode

A `tripwire` command that compares the last known properties of an object to the current properties to see if there are any violations.

key files

Files that hold the public and private keys that Tripwire for Servers uses to sign files and verify signatures. Tripwire software uses two key files, the site key file and the local key file. If either of the key files are overwritten or otherwise destroyed, any files signed with those keys will be unusable. See the Appendix for more information.

local key file

A file containing the keys that Tripwire for Servers uses to sign and verify the database file and (optionally) report files. You must specify the local passphrase to write to a file protected with the local key file.

local variable

In the policy file, a variable you define in the file system or registry sections, whose scope is limited to that section. If a local variable and a global variable have the same name, the local section uses the local variable.

MD5 algorithm

A one-way hash algorithm created by RSA Data Security Inc. and a proposed data authentication standard for high security. The Internet draft submission, Internet working draft RFC 1321, is available from **http://www.merit.edu/internet/documents**. The MD5 algorithm generates a 128-bit signature that uses four passes to ensure pseudo-random output.

message-digest algorithm

A type of algorithm used to render files tamper-evident. A small change to an input data file will cause a large change to the message digest value for that file.

network-based intrusion detection

A class of intrusion detection tools that detect intrusions by looking for anomalous patterns of network traffic.

object name

In a policy file, the name of an object that Tripwire software monitors. The object name is the first element of a rule.

passphrases

Long passwords which Tripwire for Servers uses to generate site and local keys. It then uses the keys to sign files. Once a file is signed, you must know the appropriate passphrase to update it.

policy compliance

Using Tripwire software to detect changes to the configuration of a system that violate corporate IT policy.

policy file

A file containing rules for checking system objects on a computer. Each rule in the policy file specifies a system object to be monitored, and describes which changes to the object should be reported, and which ones can safely be ignored. The policy file is encoded and signed with the site key file, and you must specify the site passphrase to change it.

predefined variable

A named set of properties that you can declare and use as a variable in a policy file rule.

print configuration file mode

A `twadmin` command that prints the current contents of the configuration file in a readable text format.

print policy file mode

A `twadmin` command that prints the current contents of the policy file in a readable text format.

private key

A component of Tripwire site and local key files that signs files.

property

A characteristic (e.g. file size, last access time, user permissions) of a system object that Tripwire software can monitor.

public key

A component of Tripwire site and local key files that verifies files that are signed.

recursion level

An optional level of subdirectory scanning for a policy file rule. You specify the level with the `recurse` attribute, choosing to scan only the starting directory or registry key, scan from the starting point through all subdirectories, or scan down to a particular level.

remove encryption mode

A `twadmin` command that removes cryptographic signatures from configuration, policy, database, and report files.

report file

A Tripwire file that presents the results of an integrity check violation.

rule

A policy file statement that specifies which system objects to scan and which object properties to include or exclude during integrity checks. A rule often specifies optional attributes as well. There is only one rule for each object and each rule ends with a semicolon.

rule attribute

An optional part of a policy file rule that specifies the rule's name (`rulename`), the object's recursion level (`recurse`), the rule's severity level (`severity`), or an e-mail address for violation notices (`emailto`).

rule block

A set of policy file rules that share common rule attributes.

rule name

An optional name for a policy file rule or block of rules. You specify the name with the `rulename` attribute.

section

A part of a policy file defined with an `@@section` directive. A policy file for a UNIX system has an optional global section (`@@section GLOBAL`) and a file system section (`@@section FS`). A policy file for a Windows system has an optional global section (`@@section GLOBAL`), a file system section (`@@section NTFS`) and a registry section (`@@section NTREG`).

severity level

A numeric value (from 0 to 1000000, with 0 as the lowest) for the importance of a policy file rule. You specify the level with the `severity` attribute. If no severity level is specified, it defaults to 0.

SHA/SHS algorithm

An algorithm for high security. SHS is the NIST Digital Signature Standard, called the Secure Hash Standard, and is described in NIST FIPS 180. It is referred to here as the SHA, or Secure Hash Algorithm, because Tripwire for Servers uses a non-certified implementation and cannot claim standards conformance. SHS generates a 160-bit hash.

signed file

A Tripwire policy file, configuration file, database file, or optionally, report file, that Tripwire for Servers signs using appropriate site and local keys. You must specify the site or local passphrase to write to a signed file.

site key file

A file containing the keys that Tripwire for Servers uses to sign and verify the configuration and policy files. You must specify the site passphrase to write to a file protected with the local key file.

stop point

In a policy file, a rule that specifies objects to ignore during an integrity check. A ! symbol marks a stop point.

system object

A file, directory, or Windows registry key or value that Tripwire software monitors. Tripwire for Servers monitors system objects according to rules in the policy file.

violation

An addition, deletion, or modification to a system object that violates a rule in the Tripwire policy file.

Index

Symbols

A

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W