Purpose and Scope

This user's guide is for the UNIX administrator responsible for installing and running the Tripwire product. It is specifically designed for less-experienced users who are familiar with UNIX systems.

Conventions

The following conventions apply:

Bold type helps identify security issues.

<*Italic*> or *`Italic`* type identifies user-defined or context-specific material: `tripwire -m p --site-keyfile` *`../key/key-file.key myfile.txt`*

Italic type identifies vocabulary specific to Tripwire software or its field, *integrity assessment.*

Brackets [ ] encase an optional argument.

{Braces} encase multiple options, of which one must be chosen. The token | delimits each option: {1 | 2 }.

./ is specified as the path for all Tripwire commands. This convention is recommended to ensure that Tripwire commands execute from the current directory, and to protect against Trojan Horse attacks.

`Courier` type identifies Tripwire commands, system filenames, directory names, examples, and syntax:

```
command mode-selector [option1][argument1]...[file]
```

Related Documents

In addition to this guide are man pages, a `policyguide.txt` file, and a Quick Reference Card.

Latest Information and Support

You can get the latest information through discussion groups at:

`http://www.tripwiresecurity.com/support/dgroups.html`

The Tripwire Security Systems technical support staff can be reached at:

`http://www.tripwiresecurity.com/supintro.html`

If necessary, you can send email to “support@tripwiresecurity.com” or call 1-877-TRIPWIRe.

Contents

About This Guide iv

Contents vii

Introduction to Tripwire Software 15

1.0 Overview 16

1.1 What Tripwire Software Is 16

1.2 Deploying Tripwire Software 17

1.3 Tripwire Components 18

1.3.1 Tripwire Asymmetric Cryptography 19

1.4 How Tripwire Software Works 20

1.5 Other Tripwire Applications 22

1.6 New Features in this Release 22

1.6.1 Changes from the Tripwire 2.2 Release 22

1.6.2 Changes from the Academic Source Release 23

Installing and Configuring Tripwire 25

2.0 Overview 26

2.1 System Requirements 26

2.1.1 Quick Start 27

2.2 Planning Passphrases 28

2.3 Installation Procedure 29

2.3.1 More About Installation Files 30

2.3.2 Tripwire Configuration File Settings 38

2.3.3 Configuration File Variables Reference 41

2.4 Overview of the Policy File 41

2.4.1 How Tripwire Monitors Itself 43

2.4.2 First-Time Edit of the Policy File 44

2.5 Initializing the Tripwire Database 45

Operations and Command Reference 47

3.0 Overview 48

3.1 Getting Command Help 48

3.2 The tripwire Command Modes 49

3.2.1 tripwire Database Initialization Mode 50

3.2.2 tripwire Integrity Check Mode 51

How to Send Integrity Check Results in Email 52

How to Screen Violations by Severity or Rule 52

How to Examine Specific Properties During Integrity Check 53

3.2.3 tripwire Database Update Mode 55

3.2.4 tripwire Policy Update Mode 57

Resolving Policy Update Violations 58

Policy Update vs. Create Policy mode 59

3.2.5 Test Email 60

3.3 twprint Command Overview 61

3.3.1 twprint Print Report Mode 61

3.3.2 twprint Print Database Mode 62

3.4 twadmin Command Overview 63

3.4.1 Replacing a Configuration File 64

3.4.2 Printing a Configuration File 65

3.4.3 Replacing a Policy File 66

3.4.4 Printing a Policy File 68

3.4.5 Removing Signatures from a File 68

3.4.6 Signing a File 69

3.4.7 Getting the Encryption Status of a File 70

3.4.8 Generating Keys 71

3.5 Siggen Command Overview 72

3.5.1 Hash Throughput Performance 74

Policy Reference 75

4.0 Overview 76

4.1 Rules 76

4.1.1 Normal Rules 77

Object names 78

Managing Recursion Across Mount Points with Rules 79

Property masks 80

4.1.2 Stop Points 83

4.2 Rule Attributes 84

4.2.1 rulename 87

4.2.2 emailto 88

4.2.3 severity 89

4.2.4 recurse 90

4.3 Directives 90

4.3.1 Conditional Interpretation 92

4.3.2 Message Reporting 93

4.3.3 Indicating End-of-File 94

4.4 Variables 94

Glossary 97

Appendix 107

Appendix A: Exit Codes 108

Appendix B: Sample Tripwire Reports 109

Index 123

Tables

Table 1: Differences between Commercial and ASR Releases 23

Table 2: Installer Command Line Arguments 31

Table 3: Installer Target Directories and Default Settings 32

Table 4: Directory and File Permissions Set at Installation 36

Table 5: TSS/man Table 37

Table 6: Help Arguments 48

Table 7: tripwire Integrity Check Mode Arguments 53

Table 8: tripwire Database Update Mode Arguments 55

Table 9: twprint Print Report Mode Arguments 60

Table 10: tripwire Test Mode Arguments 61

Table 11: twprint Print Database Mode Arguments 63

Table 12: twadmin Create Configuration File Mode Arguments 65

Table 13: Print Configuration File Arguments 66

Table 14: Create Policy File Arguments 67

Table 15: Print Policy File Arguments 68

Table 16: twadmin Removing Encryption Arguments 69

Table 17: twadmin Encryption Mode Arguments 70

Table 18: twadmin Examine Encryption Mode Arguments 71

Table 19: twadmin Generate Key Mode Arguments 72

Table 20: siggen Display Signature Arguments 73

Table 21: Property Mask Characters 81

Table 22: Tripwire Attributes 84

Table 23: Tripwire Directives 91

Table 24: Predefined Variables 96

Table 25: Exit Codes for tripwire Integrity Checking Mode 108

Table 26: Exit Codes for all other command modes: 108

Figures

Figure 1: Tripwire Software in the Corporate Network 17

Figure 2: Tripwire Operation Process 21

Figure 3: Installation Directory Tree 35

Figure 4: Default Configuration File 38

Figure 5: Defining Variables 42

Figure 6: Monitoring Tripwire Binary Files 43

Figure 7: Hash Throughput Approximations 74

Figure 8: Examples of Normal Rules 77

Figure 9: Express Object Names with Absolute Pathnames 78

Figure 10: Equivalent Ways to Express Rules 79

Figure 11: Using Quoted Strings in Object Names 79

Figure 12: Bypassing Specific Files with Stop Points 83

Figure 13: Single and Scoped Rules Using emailto Attribute 85

Figure 14: Stop Points in Scoped Rule Blocks 85

Figure 15: Grouping Rules with Rulename Attribute 87

Figure 16: Nested emailto Attributes 88

Figure 17: Valid and Invalid Use of Directives 91

Figure 18: Applying One Policy File to Several Hosts 92

1

Introduction to Tripwire Software

1.0 Overview

This section is intended for system administrators who are new to Tripwire software or who are unfamiliar with the concepts of *integrity assessment*. Experienced Tripwire users may want to skip to Section 1.5, New Features in this Release, or to Chapter 2, Installing Tripwire Software. This section describes:

- What Tripwire software is
- Deploying Tripwire software
- How Tripwire software works
- Changes for Tripwire 2.2.1

1.1 What Tripwire Software Is

Tripwire 2.2.1 is a tool for *file integrity assessment*, a form of intrusion detection that works in conjunction with firewalls and other technologies to provide the most fundamental layer of defense within the enterprise.

Tripwire software works by first scanning a computer and creating a database of system files, a compact digital "snapshot" of the system in a known secure state. You can configure Tripwire software very precisely, specifying individual files and directories on each machine to monitor, or you can create a standard template for use on all machines in an enterprise.

Once this baseline database is created, a system administrator can run an *integrity check* at any time. By scanning the current system and comparing that information with the data stored in the database, Tripwire software detects and reports any additions, deletions, or changes to the system outside of the specified boundaries. If these changes are valid, the administrator can update the baseline database with the new information. If malicious changes are found, the system administrator will instantly know which parts of which components of the network have been affected.

1.2 Deploying Tripwire Software

Tripwire 2.2.1 is not meant to replace your firewall or other perimeter security measures. Instead, Tripwire software is host-based, deployed behind perimeter security measures on the servers and workstations that compose the network.

Tripwire software works in a number of ways to ensure the integrity of your network. Because of its ability to detect intrusions, it is often deployed to "guard the guards"– to monitor the integrity of firewalls and network security appliances that are often themselves the target of attacks. At the same time, Tripwire software monitors all of the systems inside the firewall, detecting and reporting unexpected changes whether they come through the firewall or originate within the system.

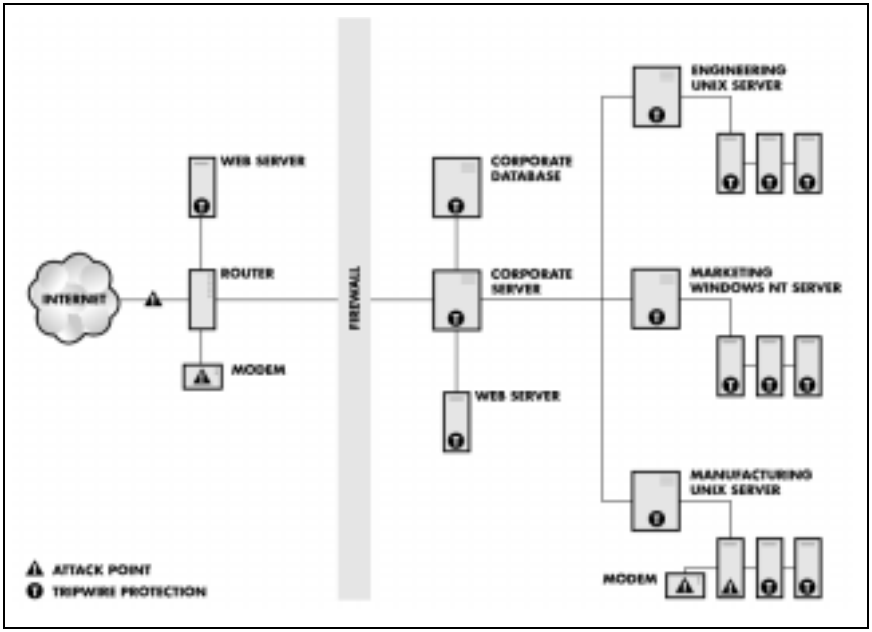

Figure 1: Tripwire Software in the Corporate Network

Because Tripwire software detects file system changes from any source, it has many advantages over other types of security software. Most other intrusion detection tools use anomaly detection, searching for attacks based on a catalog of known “attack signatures”. The problem with this strategy is that hacker exploits are changing on a daily basis, and no anomaly detection tool can keep pace with this rapid rate of change.

1.3 Tripwire Components

The *policy file* lets the administrator specify the way that Tripwire software checks a system. Each *rule* in the policy file specifies a system object that Tripwire software monitors, and describes which changes to the object should be reported, and which ones can be ignored. See Chapter 4, Policy File Reference, for a detailed description of the policy file.

The *database file* is at the center of the integrity assessment strategy. When Tripwire 2.2.1 is first installed, it uses the rules specified in the policy file to create a “snapshot” of a computer system in a known secure state. Then, during an integrity check, this “baseline” database is compared against the current state of the system to determine what, if any, changes have occurred.

Report files are produced every time an integrity check is run. The results of that check, including any changes (additions, deletions, or modifications) that violate policy file rules, will be stored in a report file. Report files summarize the results of the integrity check and can be viewed in a variety of formats, at varying levels of detail. The appendix contains sample report files for each level of detail.

The *configuration file* stores system-specific information, such as the location of Tripwire data files, and the settings used for email notification of violations. Some of this information is generated during the Tripwire installation process, but the Tripwire administrator can change these parameters at any time. See Editing the Configuration File, for more details.

1.3.1 Tripwire Asymmetric Cryptography

To protect against unauthorized modification, all important Tripwire files are stored on disk in a binary-encoded and signed form. Tripwire database, policy, configuration, and (optionally) report files are protected with El Gamal asymmetric cryptography with a 1024-bit signature.

The El Gamal signature process uses a paired set of keys, one *public key* and one *private key.* In Tripwire software's cryptographic system, two *key files* each store one pair of public and private keys. The *site key file* is used to protect policy and configuration files, which can be used across an entire site. The *local key file* is used to protect Tripwire databases and report files.

This structure makes it possible for a site administrator to author a single policy for an entire site and to sign it with a key file for which only he or she knows the *passphrase*, while delegating the task of maintaining each local database to the administrator for that system.

Unlike most cryptographic systems, Tripwire software uses cryptographic signatures to prevent unauthorized *writing* of files, rather than *reading* of files. Only the public key is required to read files, and since the public key is available to all users, anyone can view these files. However, editing or replacing these files requires the private key, which is encrypted with a secret passphrase.

You must choose a passphrase at the time that a key file is generated, and it is very important that you remember the passphrases that you choose. For security reasons, passphrases are not stored on the system, and Tripwire cannot help you recover "lost" passphrases.

Security Issue: Cryptographic techniques do not protect against all attacks, such as the deletion of all Tripwire data files. For maximum security, important files should be protected by regularly verifying their hashes using the Tripwire `siggen` utility, comparing them to known reliable backups, or storing them on read-only media.

1.4 How Tripwire Software Works

After you have installed Tripwire 2.2.1 on a system, you will need to configure the software for normal operation. These steps are described in greater detail in Chapter 2, Installing Tripwire Software.

1. Edit the default policy file, or create a custom policy file, to specify the objects that you want to monitor.
2. Using the rules in the policy file, the Tripwire software will collect data for the specified objects and initialize the Tripwire database file. For most implementations, this step only needs to be done once, when the Tripwire software is first installed.

After the database file has been generated, these steps are used for routine operation. Each of these procedures is described in greater detail in Chapter 3, Operations and Command Reference.

3. When you run an integrity check, the `tripwire` executable will compare the objects in the database file to the current system objects and write the results of the check to a report file. Email notification can also be sent to specified recipients.
4. If any changes are discovered that violate policy file rules, you can use the information in a Tripwire report file to decide if the changes are allowed.
5. If unauthorized changes are discovered, you can take appropriate measures. This may include restoring files from backup or changing security procedures to prevent further intrusions.
6. If the changes discovered are authorized, you should update the Tripwire database to reflect the changed state of the system, and to prevent these changes from being flagged as violations in the future.

After resolving all of the changes discovered by, you can run another integrity check to verify that all changes have been resolved successfully.

7. You may want to update the existing policy file to add new objects to monitor, or to remove objects from the database that are generating “noise” in Tripwire report files.

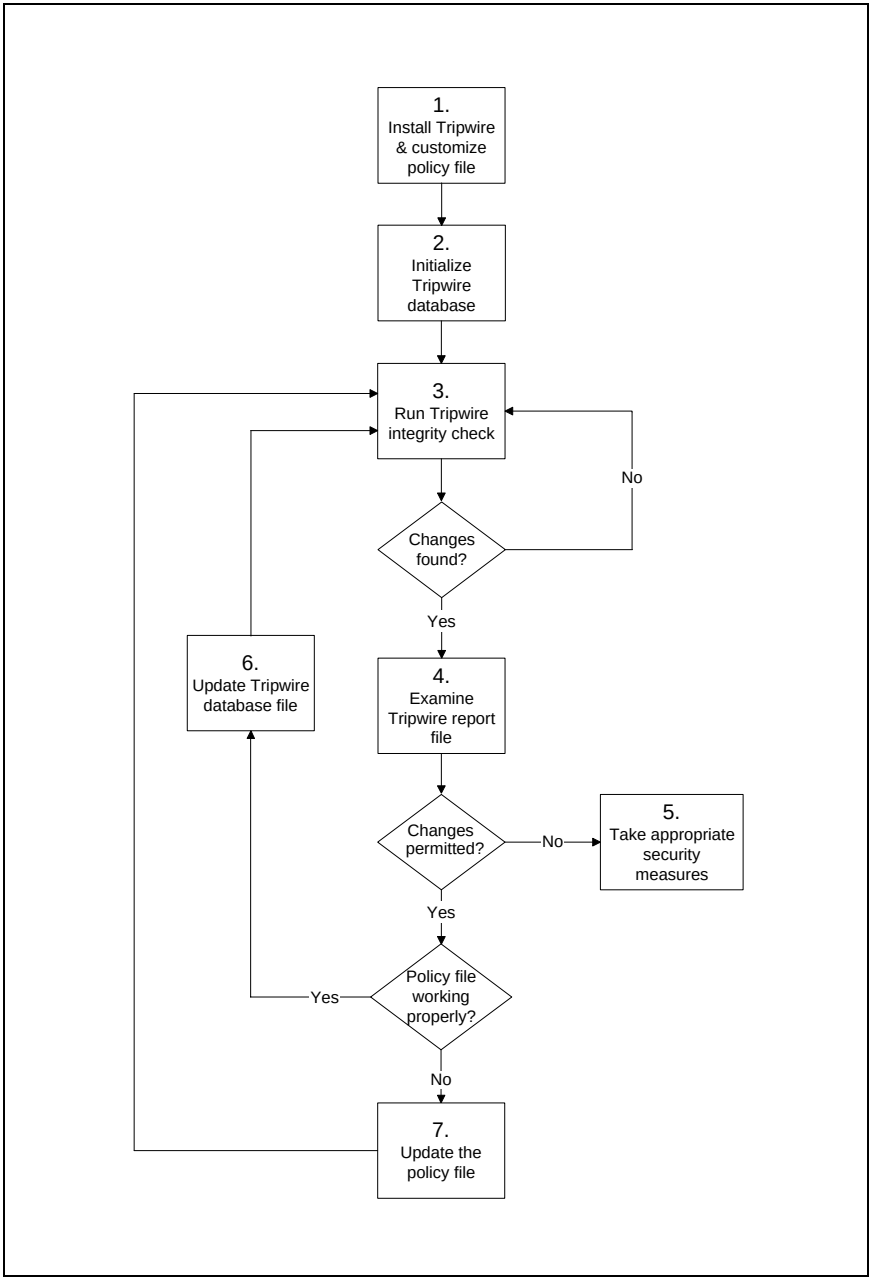

Figure 2: Tripwire Operation Process

1.5 Other Tripwire Applications

Tripwire 2.2.1 can be used in a wide range of applications outside of the field of intrusion detection. Many customers use Tripwire software in a variety of roles, from policy compliance to forensics. Some of these applications include:

System and Policy Compliance – Ensure that systems meet your IT standards by monitoring system files for any changes. By comparing your machines to a baseline database generated from an ideal system, you can easily detect potential security holes or configuration problems.

System Lockdown – Verify that no new, unauthorized software has been installed on a system. Once a machine has been "locked down", Tripwire software can monitor that system for unauthorized software or applications.

Damage Assessment and Recovery – Use Tripwire software to assist in assessing the damage in the event of a successful attack. You can use the reported violations as a list of files to repair or replace.

Forensics – Tripwire reports can be used to establish a chain of evidence necessary to prosecute offenders after an attack has occurred.

1.6 New Features in this Release

1.6.1 Changes from the Tripwire 2.2 Release

Tripwire 2.2.1 is different from Tripwire 2.2 in the following ways:

- Database update and policy update behavior now defaults to high security mode. This means that Tripwire software will exit on finding a violation without changing the policy file or the database file. Formerly the operation warned about inconsistencies but defaulted to an update mode.
- Policy files are backward-compatible to Tripwire 2.2 only.

- Due to changes to the database format, Tripwire 2.2 database and report files will not work with Tripwire 2.2.1. The installation chapter includes information about how to use pre-existing policy and configuration Tripwire files, if you have them.

1.6.2 Changes from the Academic Source Release

Tripwire's initial version, release 1.0, was originally designed in 1992. Tripwire software has been completely rewritten as of version 2.0. The rewritten code is not open source. Policies are no longer in the configuration file, and the policy language has also changed. The base 64 notation is also different. Tripwire 2.0 and later versions use four cryptographic signatures; the ASR offered eight signatures, a "no signature" option, and the ability to add a custom signature. Many of the ASR signatures are weak by current standards.

The following table list of differences between the ASR version and the commercial version.

Table 1: Differences between Commercial and ASR Releases

Tripwire 2.x Commercial Release	Tripwire 1.3.x Academic Source Release
Workstation or site license fees.	Free license per workstation (license agreement must be generated per workstation)
Commercial-level quality assurance testing and certification by Tripwire Security Systems, Inc.	Undefined bug list
• Email tech support included with license fee. • Extended support option available: four business-hour turnaround on email , same version software upgrades, bug fixes, access to knowledge base. • Telephone support option available.	Support limited to on-line knowledge base and long-term turn around email support.

Table 1: Differences between Commercial and ASR Releases

• Fast, easy, simple installation • Configuration information easily changed with Tripwire commands.	• Complex installation • User-compiled • Extensive installation configuration
• User manual • "man" page documentation	“man” page documentation only
• Optimized for performance • Ability to prioritize execution of specific policies via severity level assignments • Approximately 50% better performance than ASR version • Selectively check rules	• Resource use intensive • "All or nothing" checking
Self-certified as Y2K -compliant	Not certified as Y2K-compliant Uncertified by TSS
• Critical files in binary format and cryptographically signed • Safely stored on target machine • Facilitates automated operation	• Critical Tripwire files stored in plain-text format • Requires use of read-only media for database and configuration files • Time-consuming management of physical media.
Product upgrade path to include: • Full scale enterprise • Network deployment • Centralized management console • Trend analysis • Forensic tools • Graphical user interface	1.x Product line is static; feature set hasn't changed in four years
• Email reporting • Reports organized by user-defined rule names and severity levels. Summary sections followed by increasing detail	• No email reporting • On-host text-based reports only • "Noisy" reports

2

Installing and Configuring Tripwire

2.0 Overview

Experienced users of commercial versions of Tripwire software may want to skip directly to the Quick Start or to the Installation Procedure. This section lists system requirements, rules for passphrase protection of the cryptographic keys, and the software installation procedure, including editing the configuration and policy files for new users.

During the installation procedure, you will:

1. Specify configuration options for default operation.
2. Plan for two passphrases to be assigned for your site and local keys.
3. Run the installation script.
4. Edit the configuration and policy files.
5. Troubleshoot configuration and policy file setup, as needed.
6. Initialize the Tripwire software database.

2.1 System Requirements

- No specific hardware dependencies, other than implied by operating system versions
- HP-UX 10.20 or 11.0 with PA-RISC 1.1 or higher processor, IBM AIX 4.2 or 4.3, Solaris 2.6 or 7.0, Irix 6.5, Linux, or Tru64
- A text editor that can accept a file on the command-line, exits with 0 status on success, and exits with non-0 status on error
- Approximately 10MB available space for installation, excluding the database file
- Approximately 32 MB RAM (64 MB recommended)

2.1.1 Quick Start

If you are familiar with Tripwire software and need only a brief summary to start, this section is for you.

1. Read the README and Release Notes for the latest Tripwire software information. You'll find them in the root directory on the CD.
2. Read the rest of the Installation section of this manual.
3. If you have an existing Tripwire software installation, back it up before installing the new version. Tripwire 2.2.1 software is backward-compatible with Tripwire 2.2 software files only.
4. Copy the installation configuration file (`install.cfg`) from the distribution CD to the hard disk of the machine on which you want to install the product.
5. Modify `install.cfg` to specify installation paths for your system.
6. Run the installation command:

```
./install.sh [path]/install.cfg
```

7. Review the Configuration File Settings in this chapter and modify the twcfg.txt file as needed for your installation.
8. Review Chapter 4, Policy Reference, and modify the default policy file, or create your own. The policyguide.txt file may be helpful.
9. From the `/bin` directory, install your customized configuration file with the command:

```
./twadmin --create-cfgfile --site-keyfile ../key/site.key twcfg.txt
```

10. From the `/bin` directory, install your customized policy file with the command:

```
./twadmin --create-polfile ../policy/twpol.txt
```

11. Initialize your database file:

```
./tripwire --init
```

If you successfully completed these steps, then you are ready to begin using the Tripwire product.

2.2 Planning Passphrases

Tripwire files are signed using the site or local key. These keys are in turn protected by passphrases. The following recommendations apply:

1. Use at least eight alphanumeric and symbolic characters for each passphrase.
   The maximum length of a passphrase is 1023 characters. Any passphrase that contains wildcard characters must be quoted whenever it is entered on the command line. For this reason, quotes should not be used as passphrase characters.
2. Assign a unique passphrase for the *site* key.
   Tripwire software uses the site passphrase to protect the site key, which is used to sign Tripwire software configuration and policy files. These files are meant to be shared among many machines.
3. Assign a unique passphrase for the *local* key.
   The local key signs Tripwire database files; Tripwire report files may be signed with it as well. The same rigor in choosing the site passphrase should be used for the local passphrase.

With two passphrases, an intruder who compromises the local key on one machine does not necessarily have the ability to compromise other machines.

4. Store the passphrases in a secured location, if you wrote them down. Tripwire Security Systems cannot help you restore a lost passphrase, and there is no way to remove encryption from a signed file if you forget your passphrase. If you forget the passphrases, the files will be rendered unusable! You will have to reinitialize the database.

2.3 Installation Procedure

The installation CD contains a complete distribution of Tripwire 2.2.1.

1. If you have an existing Tripwire software installation, run a final integrity check and back it up before installing the new version. If you want to re-use your policy and configuration files, convert them to text format with the old installation's twadmin command and store them in a temporary directory until you are ready to compile them.
2. Insert the CD in the CD-ROM drive and mount it.
3. Change to the root directory of the CD.
   The root directory of the CD contains the README file and Release Notes. If you have not yet read the README, you should do so now.

   For an attended installation, continue with step 4. For an unattended installation, skip to step 5.
4. Run the installation script according to whether you are using default or custom configuration values. To use default configuration values, enter:

```
./install.sh
```

For a custom installation, copy the `install.cfg` file from the CD to the hard drive and modify it before executing it:

```
./install.sh /tmp/install.cfg
```

where `/tmp/install.cfg` is the argument specifying a custom configuration file.

5. To perform an unattended install, you would use a command similar to the following:

```
./install.sh /tmp/install.cfg -s "Darth4Vader" -l "Sky8Walker" -f -n
```

6. Replace the text configuration and policy files with the files you saved in step 1.
7. Make any additional changes you need to, and then compile them with the Tripwire 2.2.1 `twadmin` utility.

2.3.1 More About Installation Files

The file `install.cfg` is a Bourne shell script used by the installer to set configuration variables. These variables specify the target directories where the installer will copy files and what the installer should do if the installation process would overwrite existing Tripwire software files.

`install.sh` is the installation script, which you run to begin installation. You can specify that it should read the default installation configuration file, or the configuration file that you customized for your site.

`License.txt` is the license agreement for Tripwire 2.2.1. You will be required to accept the license agreement to install the product, but you should read it beforehand to understand your rights.

`/pkg` is a directory containing files that the installation script needs to install Tripwire software on your machine. These files are used exclusively by the installer and should not be modified.

`/bin` is a directory containing Tripwire software binaries, which can be used to verify Tripwire software installations on the hard disk. These can be run on an as-needed basis directly from the removable media to prevent unauthorized user access to the executables.

You can specify any installation option as a command-line argument to the install script `install.sh`, or you can set options in the configuration file.

The following table summarizes installer command line arguments:

Table 2: Installer Command Line Arguments

configfile	Use the specified file for installation values. By default, the installer uses the values in *./install.cfg* for installation options.
-n	No prompting. By default, the installer will display all the target directories that will be created and populated, and prompt the user for verification before proceeding. Using -n turns on -f implicitly. This mode requires the site and local passphrase arguments.
-s *passphrase*	Use the specified passphrase for site key.
-l *passphrase*	Use the specified passphrase for local key.
-f	Force installer to overwrite any existing files found in the target directories. This will override the `CLOBBER` setting in the *install.cfg* file.

An example of settings found in the configuration file is shown below:

```
# The root of the TSS directory tree.
TWROOT="/usr/TSS"

# Tripwire software binaries are stored in TWBIN.
TWBIN="${TWROOT}/bin"
```

The next table shows each installer target directory and its default setting:

Table 3: Installer Target Directories and Default Settings

Key	Default Setting	Description & Behavior
TWROOT	/usr/TSS	The root directory; all other Tripwire files appear underneath `/usr/TSS`.
TWBIN	${TWROOT}/bin	Contains the program executables and the configuration files.
TWPOLICY	${TWROOT}/policy	Contains the policy files.
TWMAN	${TWROOT}/man	Contains the man pages.
TWDB	${TWROOT}/db	Contains the database files created from the policy file.
TWSITEKEYDIR	${TWROOT}/key	Contains the site cryptographic key, which is used to secure configura-tion and policy files.
TWLOCALKEYDIR	${TWROOT}/key	Contains the local cryptographic key, which is used to secure database files and reports.
TWREPORT	${TWROOT}/report	Contains the results of integrity checks for archival purposes.
TWEDITOR	/bin/vi	Specifies the editor for interactive report modes.
TWLATEPROMPTING	false	Sets LATEPROMPTING value in `tw.cfg` and `twcfg.txt`.
TWLOOSEDIRCHK	false	Sets LOOSEDIRECTORY-CHECKING value in `tw.cfg` and `twcfg.txt`.

Table 3: Installer Target Directories and Default Settings

TWMAILNOVIOLATIONS	true	Sets MAILNOVIOLATIONS value in the `tw.cfg` and `twcfg.txt`. Controls behavior if an integrity check finds no violations to report in email.
TWEMAILREPORTLEVEL	3	Sets EMAILREPORTLEVEL value in the `tw.cfg` and `twcfg.txt`, which controls verbosity of email reports.
TWREPORTLEVEL	3	Sets REPORTLEVEL value in the `tw.cfg` and `twcfg.txt`, which controls verbosity of report printouts.
TWSYSLOG	true	Sets SYSLOGREPORTING value in the `tw.cfg` and `twcfg.txt` to log events to the system log.
TWMAILMETHOD	SENDMAIL	Sets MAILMETHOD value in the `tw.cfg` and `twcfg.txt`, which controls how Tripwire sends email reports.
TWMAILPROGRAM	/usr/lib/sendmail -oi -t	Sets the MAILPROGRAM value in the `tw.cfg` and `twcfg.txt`, which sets the program used to send email reports if MAILMETHOD is set to SENDMAIL.
TWSMTPHOST	mail.domain.com	Sets SMTPHOST value in the `tw.cfg` and `twcfg.txt`. Commented (inactive) by default.

Table 3: Installer Target Directories and Default Settings

TWSMTPPORT	25	Sets SMTPPORT value in the `tw.cfg` and `twcfg.txt`. Commented (inactive) by default.
CLOBBER	false	Specifies whether the installer will overwrite existing files. By default, the installer will not overwrite existing files.

The CLOBBER Setting. When files would be overwritten, Tripwire software will print a warning and skip the file copy for that file, unless the file is a configuration or policy file. To insure a self-consistent installation, the installation procedure always creates new copies, saving existing configuration or policy files with a `.bak` extension.

The default installation process will create the following directory tree:

```
/usr/TSS
            README
            License.txt
            Release_Notes
            /bin
                twcfg.txt
                tw.cfg
                siggen.exe
                tripwire.exe
                twadmin.exe
                twprint.exe
            /db
            /key
                site.key
                $(Hostname)-local.key
            /man
                /man4
                    twpolicy.4
                    twconfig.4
                /man5
                    twfiles.5
                /man8
                    siggen.8
                    tripwire.8
                    twadmin.8
                    twintro.8
                    twprint.8
            /policy
                policyguide.txt
                tw.pol
                twpol.txt
            /report
```

Figure 3: Installation Directory Tree

The permissions for the installed directories and files appear in the table on the next page.

Table 4: Directory and File Permissions Set at Installation

Directory	Permissions	Contents	Permissions	Files
TSS	drwxr-xr-x	License.txt	-r--r--r--	
		README	-r--r--r--	
		Release_Notes	-r--r--r--	
		bin	drwxr-x---	See TSS/bin
		db	drwxr-x---	See TSS/db
		key	drwxr-x---	See TSS/key
		man	drwxr-xr-x	See **TSS/man Table**
		policy	drwxr-x---	See TSS/policy
		report	drwxr-x---	See TSS/report
TSS/bin	drwxr-x---	siggen	-r-xr-x---	
		tripwire	-r-xr-x---	
		tw.cfg	-rw-r-----	
		twadmin	-r-xr-x---	
		twcfg.txt	-rw-r-----	
		twprint	-r-xr-x---	
TSS/db	drwxr-x---			
TSS/key	drwxr-x---	site.key	-rw-r-----	
		$(Hostname).key	-rw-r-----	
TSS/policy	drwxr-x---	policyguide.txt	-r--r-----	
		tw.pol	-rw-r-----	
		twpol.txt	-rw-r-----	
		twpol.txt.bak	-rw-r-----	
TSS/report	drwxr-x---			

Table 5: TSS/man Table

Directory	Permissions	Contents	Permissions	Files	Permissions
TSS/man	drwxr-xr-x	man4	drwxr-xr-x		
				twconfig.4	-r--r--r--
				twpolicy.4	-r--r--r--
		man5	drwxr-xr-x		
				twfiles.5	-r--r--r--
		man8	drwxr-xr-x		
				siggen.8	-r--r--r--
				tripwire.8	-r--r--r--
				twadmin.8	-r--r--r--
				twintro.8	-r--r--r--
				twprint.8	-r--r--r--

After you install the software, you must:

1. Edit `twcfg.txt` with a text editor.
2. Edit the policy file with a text editor.
3. Sign the edited policy and configuration files.
   For example, enter the following command to sign the configuration file:

```
./twadmin --create-cfgfile --site-keyfile ../key/site.key twcfg.txt
```

   The encoded and signed version of the configuration file should not be renamed or moved.
4. Delete the plain text copy to hide the locations of Tripwire's files and prevent anyone from creating a second configuration file.
5. Initialize the Tripwire database file.
6. Run the first integrity check.

The next sections describe editing the configuration file and the policy file in more detail.

2.3.2 Tripwire Configuration File Settings

The installation process used the shell script, `install.cfg`, to create an encoded and signed configuration file, `tw.cfg`, and a text copy, `twcfg.txt`, in the Tripwire `/bin` directory. (The file `install.cfg` refers to the ${ROOT}directory. The file `twconfig.txt` refers to the $(ROOT) directory.) The installation procedure also set the default values for `tw.cfg` and `twcfg.txt`. Once you completed the installation, you can make any additional changes needed to `twcfg.txt`.

The default configuration file will resemble this example, where $(ROOT) is the Tripwire destination directory set by the `install.cfg` file:

```
# Tripwire Default Configuration File.
DBFILE=$(ROOT)/db/$(HOSTNAME).twd
POLFILE=$(ROOT)/policy/tw.pol
REPORTFILE=$(ROOT)/report/$(HOSTNAME)-$(DATE)tripwire.twr
SITEKEYFILE=$(ROOT)/key/site.key
LOCALKEYFILE=$(ROOT)/key/$(HOSTNAME)-local.key
MAILMETHOD=SENDMAIL
MAILPROGRAM=/usr/lib/sendmail -oi -t
# SMTPHOST=<system-name>
# SMTPPORT=<port-specification>
LATEPROMPTING=false
LOOSEDIRECTORYCHECKING=false
MAILNOVIOLATIONS=true
EDITOR=/bin/vi
REPORTLEVEL=3
EMAILREPORTLEVEL=3
SYSLOGREPORTING=true
```

Figure 4: Default Configuration File

The Tripwire configuration file consists of case-sensitive keyword-value pairs and variables. It may contain comments, which follow the # symbol.

The Tripwire file variables—POLFILE, DBFILE, REPORTFILE, SITEKEYFILE, and LOCALKEYFILE—specify where the policy file, the Tripwire database file, the Tripwire report files, and the site and local key files, respectively, reside. These are the minimum requirements for a

valid configuration file. If any one of these variables is not defined, an error will occur on execution and the program will exit.

To share a single site key file across multiple systems, place the site key file on a shared volume and specify the location of that key file in the configuration file. For example:

```
SITEKEYFILE=/mnt/server/keydir/site.key
LOCALKEYFILE=/usr/local/tw/key/local.key
```

`MAILMETHOD=SENDMAIL` specifies the system's mailing protocol. The only valid values for this variable are SMTP and SENDMAIL; MAPI is not supported under UNIX. The default value is SENDMAIL.

`SMTPHOST=<system-name>` specifies the domain name or IP address of the SMTP server used for email notification. Ignored if MAILMETHOD is set to SENDMAIL.

`SMTPPORT=<port specification>` Specifies the port number used with SMTP. Ignored if MAILMETHOD is set to SENDMAIL. If you change MAILMETHOD to SMTP, change to the TCP/IP port on which your mail server listens for connections.

`MAILPROGRAM=/usr/lib/sendmail -oi -t` specifies the program that will be used for email reporting of rule violations detected by Tripwire. The program must take an RFC822 style mail header. Recipients will be listed in the "To:" field of the mail header. Mail headers and the body of the report are sent to `stdin` of MAILPROGRAM. The mail program must be able to ignore lines that consist of a single period (the `-oi` option to `sendmail` produces this behavior). This variable is ignored if `MAILMETHOD=SMTP`.

`LATEPROMPTING=true` delays the prompt for a passphrase for as long as possible, shortening the amount of time the passphrase is in memory. The recommended setting is `true`.

`LOOSEDIRECTORYCHECKING=false` specifies that Tripwire will not check directories for any properties that would change when a file is

added or deleted. If the value for this variable is `true`, then loose directory checking is turned on. Set it to `true` to reduce noise in reports, but be aware this will create a very small security hole by suppressing monitoring for some properties.

`MAILNOVIOLATIONS=true` enables email messages for integrity checks if no violations occurred. In practice, this is often set to `true`. However, if you run Tripwire automatically, you may want to receive mail assuring you that the Tripwire process executed as scheduled. The recommended setting is `false`.

`EDITOR=<selected text editor>` specifies the editor to use in any interactive mode. If `EDITOR` is not defined, and no editor is specified on the command line, Tripwire will look at the $VISUAL or $EDITOR environment variables. If there is no editor defined by any of these sources, then specifying interactive mode will cause an error. The default value is `/bin/vi`.

`REPORTLEVEL=3` specifies the default level of report for the `twprint --print-report` mode. The default value is 3, and values may be set from 0 to 4. “Appendix B: Sample Tripwire Reports” on page 109 includes examples of each report type.

`EMAILREPORTLEVEL=3` specifies the default level of report sent by the command `tripwire --check --email-report`. The default level is 3, and values may be set from 0 to 4.

`SYSLOGREPORTING=true` specifies that the syslog should receive reports about database initialization, integrity check completions, database updates, and policy updates. Tripwire syslog messages are sent from the “user” facility at the “notice” level. See syslogd(1) for more information. The default value is `true`.

If you specify email reporting, make sure the values for MAILMETHOD, SMTPHOST, SMTPPORT, MAILPROGRAM, and EMAILREPORTLEVEL are configured appropriately.

2.3.3 Configuration File Variables Reference

When you edit the configuration file after installation, you should know that:

- Variables must be defined before they are used.
- There is no logical limit to the number of variables that may be defined.
- Right-hand substitution is permitted using the syntax: $(*varname*)
- Relative pathnames are permitted, expressed in relation to the directory in which the Tripwire binaries reside.
- Variable names are case-sensitive, and may contain all alphanumeric characters; underscores; and the characters plus (`+`), minus(`-`), at-symbol (`@`), colon (`:`), and period (`.`).
- Two variables are predefined by the Tripwire package and may not be redefined: HOSTNAME, which is the unqualified hostname that Tripwire is running on, and DATE, which is a string representation of the date and time (e.g. 19990127-180833).
- Tripwire will always use the file called `tw.cfg` located in the same directory as the Tripwire executables, unless it is overridden with a `--cfgfile` command-line argument.
- If the SITEKEYFILE variable points to a different key file from that used to sign the configuration file, Tripwire will print a warning.

2.4 Overview of the Policy File

The policy file specifies what files and directories, called *system objects*, to check. A *rule* specifies how to check the object you want to monitor, and a *property* is what specifies how to check. A *property mask* specifies individual *properties* of a file to examine during integrity checks.

Attributes help refine how groups of rules work. For example, *Rulename* gives a meaningful name to one or more rules. *Severity levels* group rules by priority, either within a group of rules, or across the entire policy file. *Emailto* enables email notification as specified by the `tw.cfg` file and the

command line. *Recurse* governs how deeply into the directory tree to conduct integrity checking.

Variable definitions appear at or near the top of the file and will be similar to the following:

```
# Variable Definitions
#
SIGHIGHEST =   $(ReadOnly) +H +C  ;   #Use all four hashes.
SIGHIGH    =   $(ReadOnly) +H     ;   #Use three hashes.
SIGMED     =   $(ReadOnly)        ;   #Use two hashes.
```

Figure 5: Defining Variables

- SIGHIGHEST means the highest level of security defined in this policy file.
- $(ReadOnly) is a predefined variable that specifies a group of properties to check, which include the SHA and MD5 hashes. (See “Predefined Variables” on page 96 for a complete list of predefined variables.)
- +H adds the HAVAL hash as a property.
- +C adds the CRC-32 hash as a property. See the properties table for a complete list of properties you can use to define variables.

2.4.1 How Tripwire Monitors Itself

The rules Tripwire uses to monitor itself appear deeper into the policy text file. These rules will be similar to ones shown in the following figure:

```
 Tripwire Binaries
(rulename = "Tripwire Binaries", severity = $(SIG_HI))
{
  $(TWBIN)/siggen   -> $(ReadOnly);
  $(TWBIN)/tripwire -> $(ReadOnly);
  $(TWBIN)/twadmin  -> $(ReadOnly);
  $(TWBIN)/twprint  -> $(ReadOnly);
}

# Tripwire Data Files - Configuration Files, Policy Files, Keys,
# Reports, Databases
(rulename = "Tripwire Data Files", severity = $(SIG_HI))
{
  $(TWDB)/$(HOSTNAME).db            -> $(Dynamic);
  $(TWPOL)/tw.pol                   -> $(SEC_BIN);
  $(TWBIN)/tw.cfg                   -> $(SEC_BIN);
  $(TWLKEY)/$(HOSTNAME)-local.key   -> $(SEC_BIN);
  $(TWSKEY)/site.key                -> $(SEC_BIN);

  #don't scan the individual reports
  $(TWREPORT)                       -> $(Dynamic) (recurse=0);
}
```

Figure 6: Monitoring Tripwire Binary Files

where SIG_HI = 100 and SEC_BIN = $(ReadOnly)

For performance reasons, you may choose to specify only one or two hashes. The number of hashes you assign for file protection depends upon balancing your performance needs against your security needs. See Chapter 3 to get an idea of relative hash performance and related dependencies.

2.4.2 First-Time Edit of the Policy File

If you have not used Tripwire before, please read through this section to the end of the chapter before you proceed. Any decisions you have recorded about what files to protect and what file properties you want to monitor will be useful in this procedure. This procedure tells you how to express your decisions for Tripwire operations. Once you have deployed your first policy file, please refer to the `tripwire` Policy Update Mode section to change it.

Editing the policy file can take several iterations as you establish by experience what information you want to appear in the Tripwire reports. Experienced users of Tripwire software offer these suggestions for preparing to edit the policy file for the first time:

- Consider categorizing files and directories into groups that change under similar circumstances (such as boot).
- Consider categorizing items such as `setuid` and `setgid`, shared libraries, static libraries, and devices.
- Consider grouping files based on all the files associated with a given software application.
- Consider grouping files by a user-assignable characteristics, such as a numeric or alphabetic severity level.

1. Open the policy file `twpol.txt` with a text editor.
2. Follow the syntax shown and read the comments carefully.
3. Edit the file as needed. Refer to the Policy Reference chapter for details on options.
4. When you are ready to use your policy file for the first time, create it with the following command:

```
./twadmin --create-polfile ../policy/twpol.txt
```

This is the only time you will use the `--create-polfile` mode. For very important information about why not, please see the detailed discussion of the Policy Update mode in Chapter 3.

After you have created the policy file, you are ready to initialize the Tripwire database.

2.5 Initializing the Tripwire Database

After creating your first policy file, the next step is to create the baseline Tripwire database. In Database Initialization mode, `tripwire` reads the policy file, generates a database based on its contents, and then signs the resulting database, putting it in the location specified by the configuration file's DBFILE variable. (Additional command-line arguments can be entered to specify the policy, configuration, and key files used to create the database if you don't want to use the defaults set in the configuration file.)

Make sure that the integrity of the system you are running has not been compromised. For maximum confidence in your baseline database, you should generate operating system and application files from original media to ensure that you get a clean baseline. Run the `tripwire` program in Database Initialization mode using the following command (for command options, see information about the Database Initialization mode in Chapter 3):

```
./tripwire --init
```

When this command has executed, the database is ready and you can check system integrity and review the report file. The next chapter describes all Tripwire commands and operations, or you can refer to the Quick Reference Card for command summaries.

3

Operations and Command Reference

3.0 Overview

This section is especially for new users to help further clarify the syntax among commands and modes and the use of modes for specific functions; experienced Tripwire software customers may prefer to skip directly to the subsections for each command.

The commands that will be covered next are `tripwire`, `twadmin`, `twprint`, and `siggen`. All of the commands have multiple modes except for the signature generator, `siggen`.

3.1 Getting Command Help

You can get usage, version, and copyright information from any command. All Tripwire commands support the help arguments.

Table 6: Help Arguments

Argument	Meaning

`-?`
`--help`
`--help all`
`--help [mode]`
	Display usage and version information.
	Display all command modes.
	Display help for all command modes.
	Display help for current command mode.

`--version`	Display version information.

For example, to get help with the Create Configuration File mode, enter:

```
./twadmin --help --create-cfgfile
```

Although Tripwire software accepts wildcard expansion by the shell in some command modes, Tripwire Security Systems discourages them. Using wildcards on the command line creates a small but significant security exploit. By inserting a file with the same name as a command-line argument, an intruder could adversely affect Tripwire operation. Simi-

larly, commands should be executed with an absolute pathname, and not rely on the search path. If a falsified copy of Tripwire software were to be inserted and found by the operating system before the authorized copy, the false copy would execute, not the true copy.

3.2 The `tripwire` Command Modes

All Tripwire command modes have a short form for convenience, and a long form with a meaningful name.

The `tripwire` command has five functions. It:

- Creates the initial database file
- Performs integrity checks
- Maintains the database file
- Maintains the policy files
- Tests email

Each of these functions has its own mode. All modes have a short and a long form as shown in the braces { }.

Creating the database for the first time. In Database Initialization mode, Tripwire software builds a database of filesystem objects, based on the rules in the policy file. This database will serve as the baseline for later integrity checks. The syntax for Database Initialization mode is:

```
tripwire { --m i | --init }
```

Running the integrity check. The Integrity Check mode compares the current file system objects with their properties as recorded in the Tripwire database. Violations will be printed to `stdout`; the report file will besaved and can later be accessed by `twprint`. An email option enables you to send email. The syntax for Integrity Check mode is:

```
tripwire { -m c | --check }
```

Updating the database after an integrity check. Database Update mode enables you to update the Tripwire database after an integrity check, if the violations discovered are actually valid. This update process saves you time by enabling you to update the database without having to regenerate it; even more importantly it enables selective updating, which cannot be done through regeneration. The syntax for Database Update mode is:

```
tripwire { -m u | --update }
```

Updating the policy file. You can change the rules in the policy file, which will change the way that Tripwire software scans the system, and update the database without requiring a complete re-initialization. This can save a significant amount of time; even more importantly, it preserves security by keeping the policy file synchronized with the database it uses. The syntax for Policy Update mode is:

```
tripwire { -m p | --update-policy }
```

Testing email functions. Test mode tests Tripwire's email notification system, using the settings currently specified in the configuration file. The syntax for Email Test Reporting mode is:

```
tripwire { -m t | --test }
```

3.2.1 tripwire Database Initialization Mode

When you initialize a database, you are creating the snapshot of the file-system against which all future integrity checks shall be run, so it is essential to be sure that your system objects are uncompromised. Tripwire software cannot detect a malicious program if that program has been initialized into the Tripwire database, for example.

In the following example, the `tripwire` command uses the policy file named *LIGHTHOUSE.pol* to initialize a database:

```
./tripwire -m i -p  ../policy/LIGHTHOUSE.pol -S  ../key/site.key
```

Some Tripwire software customers find that maintaining multiple custom configuration files makes it easier to organize multiple types of integrity checks for their environments. In the following example, the `tripwire` command uses an alternate configuration file to initialize a database:

```
./tripwire --init -c alt-config.cfg
```

If you do not specify any arguments for the policy, database, configuration file, or report file, Tripwire software uses those values specified in the configuration file`tw.cfg`. You can specify custom files with the arguments in the next table.

3.2.2 tripwire Integrity Check Mode

When you run an integrity check, Tripwire software scans the system for violations to the current policy file rules. Using these rules, Tripwire software will compare the state of the current file system against the values stored in the database file with the following command and one or more command-line arguments:

```
./tripwire -m c
```

After an integrity check, Tripwire software will print a text report of discovered violations to `stdout` and save a binary copy of the report in the location specified by the REPORTFILE setting in the Tripwire configuration file. You can also specify an alternative location to store the report when you start the integrity check:

```
./tripwire -m c --twrfile myreport.twr
```

How to Send Integrity Check Results in Email

If you want email to be sent with the results of an integrity check, enter:

```
./tripwire -m c -M
```

(You must have rules in your policy file with `emailto` attributes to do this; see 4.3.2“emailto” on page 85.)

How to Screen Violations by Severity or Rule

If you’ve written rules in your policy file to scan for violations by medium priority, or *severity level*, and you want to run an integrity check just for violations at that severity or higher, enter the command:

```
./tripwire -m c -l medium
```

If you want to run a specific rule, the command would look similar to this:

```
./tripwire -m c -R rulename
```

If you wanted to run several rules, you would need to nest them in the policy file or assign them all a unique name using the rulename attribute and use that name to run the command. If you want to check specific objects only, enter the command:

```
./tripwire -m c -I myfile1 myfile2 myfile3 directory1 directory2
```

If forensics are important to you, you should be aware of the following: If an intruder successfully changes the permission of the directory where the current Tripwire database is located, such that the software cannot write to that directory after an integrity check, `tripwire` can’t rename the existing database with the `.bak` extension. What will happen is that the Tripwire software will overwrite the existing database, eliminating the data you would need for forensic analysis. You can prevent this by keeping regular backups on read-only media.

How to Examine Specific Properties During Integrity Check

To ignore some file properties, which is sometimes useful if you want to examine system objects in a very particular way from the command line, list the properties to ignore. For example:

```
./tripwire -m c -i “p,c,m”
```

where the characters in quotation marks are properties. See Table 18, “Property Mask Characters,” on page 78 for a complete table of properties.

Table 7: tripwire Integrity Check Mode Arguments

Argument	Meaning
`{ -m c | --check }`	Mode selector.
`[-I]` `[--interactive]`	At the end of integrity check, the resulting report is opened in an editor for database updates.
`[-v]` `[--verbose]`	Verbose sends additional status information. Mutually exclusive with `--silent`.

`[-s]`
`[--silent]`
`[--quiet]`
	Silent suppresses status information. Mutually exclusive with `--verbose`.

`[-p` *`polfile`*`]` `[--polfile` *`polfile`*`]`	Use the specified policy file.
`[-d` *`database`*`]` `[--dbfile` *`database`*`]`	Use the specified database file.
`[-c` *`cfgfile`*`]` `[--cfgfile` *`cfgfile`*`]`	Use the specified configuration file.
`[-S` *`sitekey`*`]` `[--site-keyfile` *`sitekey`*`]`	Use the specified site key file to read the configuration and policy files.
`[-L` *`localkey`*`]` `[--local-keyfile` *`localkey`*`]`	Use the specified local key file to read the database file. Also used to write the database file when `--interactive` is used.

Table 7: tripwire Integrity Check Mode Arguments

Argument	Meaning
[-V *editor*] [--visual *editor*]	Use the specified editor to edit the report ballot box. Only applies with the `--interactive` option.
[-P *passphrase*] [--local-passphrase *passphrase*]	Use the specified passphrase with the local key. Also used to write the database file in `--interactive` mode. Valid only with `--signed-report` or `--interactive`.
[-n] [--no-tty-output]	Suppress printing the report at the console.
[-r *report*] [--twrfile *report*]	Writes the output report file to the specified file.
[-M] [--email-report]	Specifies that reports be emailed to the recipients designated in the policy file, using the options in the default configuration file.
[-E] [--signed-report]	Specifies that the Tripwire report will be signed. If no passphrase is specified on the command line, Tripwire software will prompt for the local passphrase.
[-t { 0|1|2|3|4 }] [--email-report-level { 0|1|2|3|4 }]	Specifies the amount of detail to include in a Tripwire email report.
[-l { *level* | *name* }] [--severity { *level* | *name* }]	Check only policy rules equal to or greater than the given severity level or name. Three predefined severity values are available: Low (33), Medium (66), and High (100). Mutually exclusive with the `--rule-name` option.
[-R *rulename*] [--rule-name *rulename*]	Run only the specified policy rule. Mutually exclusive with `--severity`.
[-i *list*] [--ignore *list*]	Do not compute or compare the properties specified in list. This will apply to the entire policy file during the integrity check.
[-x *section*] [--section *section*]	Check policy rules in this section of the policy file.

Table 7: tripwire Integrity Check Mode Arguments

Argument	Meaning
[*object1 object2…*]	List of file system objects that should be checked. If not specified, every object specified in the policy file will be integrity checked. This option overrides the `--severity` and `--rule-name` options.

3.2.3 tripwire Database Update Mode

When you run Tripwire software in Database Update mode, the report on violations between the current policy file and the current state of the system will be specified on the command line. The editor defaults the one specified by -V on the command line, or the value in the configuration file, $VISUAL, or $EDITOR, in that order.

Database Update mode enables you to change the Tripwire database to reflect the current file system value. You can invoke the Database Update mode with the command:

```
./tripwire -m u
```

If you do not specify a report on the command line, `tripwire` will read in the report specified by the REPORTFILE variable in the Tripwire configuration file. If the configuration file specifies the report filename using time-based variables, the report will not be found. This happens because the $(DATE) variable will have changed to reflect the current time. If this is the case, the report file must be specified on the command line with the `-r` or `--twrfile` argument.

Table 8: tripwire Database Update Mode Arguments

Argument	Meaning
`{ -m u | --update }`	Mode selector.

Table 8: tripwire Database Update Mode Arguments

Argument	Meaning
[-v] [--verbose]	Verbose sends additional status information. Mutually exclusive with --silent.
[-s] [--silent] [--quiet]	Silent suppresses additional status information. Mutually exclusive with --verbose.
[-p *polfile*] [--polfile *polfile*]	Use the specified policy file.
[-d *database*] [--dbfile *database*]	Update the specified database file.
[-c *cfgfile*] [--cfgfile *cfgfile*]	Use the specified configuration file.
[-S *sitekey*] [--site-keyfile *sitekey*]	Use the specified site key file to read the configuration and policy files.
[-L *localkey*] [--local-keyfile *localkey*]	Use the specified local key file to read and write the database and to read the report file.
[-V *editor*] [--visual *editor*]	Use the specified editor to edit the update report. The absolute pathname to the editor must be specified. Mutually exclusive with --accept-all.
[-r *report*] [--twrfile *report*]	Read the specified report file. This option is required if the REPORTFILE variable in the current configuration file uses $(DATE).
[-P *passphrase*] [--local-passphrase *passphrase*]	Use the specified passphrase with the local key to sign the database file.
[-a] [--accept-all]	Specifies that all the entries in the report file will be updated without any prompting. Mutually exclusive with --visual.
[-Z { high|low }] [--secure-mode { high|low }]	Specifies Tripwire behavior if the current database is inconsistent with the information in the .twr file used to update the database. High (default): A violation causes a warning to stderr and the database is not changed. Low: Inconsistencies are reported as warnings, but the changes are made to the database.

You may encounter errors when updating the database file if:

- The report file specified has already been used to update the database file
- The report file specified was generated using a different database file than the database currently being updated
- The database has been updated with another report file since the specified report file was generated

If you are updating the database with `--secure-mode high` (default setting), any of these situations will cause the program to exit without updating the database. If `--secure-mode is low`, a warning will be printed, but the database will be updated with the new information. Because any of the conditions described above could lead to corruption of the database file, the recommended security setting in Database Update mode is `high`.

In Database Update mode or Interactive Check mode, Tripwire software displays the report with a ballot box next to each policy violation. You can approve a change to the file system by leaving the "x" next to each policy violation. If you remove the "x" from the ballot box, the database will not be updated with the new value(s) for that object. After you exit the editor and provide the local passphrase, Tripwire software will update and save the database.

3.2.4 tripwire Policy Update Mode

You can use Policy Update mode to change the contents of the policy file and to synchronize an existing database with this new policy file information. The process follows the following steps:

1. The `tripwire` executable compares the new, plain text policy file specified on the command line to the existing version of the policy file.
2. The `tripwire` executable runs an integrity check using the rules in the new policy file, to gather information about the current state of the filesystem.
3. As data is collected, any violations (additions, deletions, or changes) of the rules in the old policy file **that are also covered by rules in the new policy file** will be detected and reported.

How these violations are interpreted depends on the security mode specified with the -Z or --secure-mode option:

- In **high** security mode (the default), tripwire will print a list of violations and exit without making changes to the database.
- In **low** security mode, the violations are still reported, but changes to the database are made automatically. If you want to capture these warnings, redirect stderr to a log file.

After the policy update process is complete, the old version of the policy file is replaced with the new version, and the new database file reflects the current state of the system.

Security Issue: Conflicts discovered during the Policy Update process should be treated with the same seriousness as integrity checking violations. For this reason, it is recommended that you always run Policy Update mode with --secure-mode high, so that these situations can be detected, and appropriate actions taken.

Resolving Policy Update Violations

To update the default policy file tw.pol with the text file policy.txt:

```
./tripwire -m p ../policy/policy.txt
```

By default, Policy Update mode runs with --secure-mode high. You may encounter errors when running in high security mode if the file system has changed since the last database update, and if the changes still cause a violation in the new policy. This may happen if another administrator is modifying files during the policy update process, for example.

To accommodate this situation, after determining that all of the violations reported in high security mode are authorized, you can update the policy file in low security mode:

```
./tripwire -m p --secure-mode low ../policy/policy.txt
```

Policy Update vs. Create Policy mode

Although you can use the `twadmin --create-polfile` command to create a new policy file, doing so requires you to re-initialize the database. This is necessary for new Tripwire installations, but can create a security risk if you have been monitoring a system for any length of time. When you re-initialize the Tripwire database, any files that have been modified since the last integrity check will be included in the newly-created "baseline" database.

For example, suppose that sometime after the last integrity check, an intruder modifies an important security file. Meanwhile, the Tripwire administrator decides to modify the policy file rule for that file.

When run in Policy Update mode, Tripwire 2.2.1 gathers information from the file system for this "new" rule, and compares it to the information collected by the "old" rule. Because the information collected for this file differs from the information already in the database, the conflict will be detected and reported.

If the Create Policy mode were used to change the policy file, and a new database was initialized based on that new policy file, the modification would not be detected. In fact, the modified file would be stored in the database file and used as the standard for later integrity checks.

To change the policy file, enter the command:

```
./tripwire -m p -p ../policy/tw.pol --secure-mode high ../policy/policy.txt
```

Table 9: twprint Print Report Mode Arguments

Argument	Meaning
`{ -m r | --print-report }`	Mode selector.
`[-v]` `[--verbose]`	Verbose sends additional status information. Mutually exclusive with `--silent`.

`[-s]`
`[--silent]`
`[--quiet]`
	Silent suppresses additional status information.
	Mutually exclusive with `--verbose`.

`[-c` *`cfgfile`*`]` `[--cfgfile` *`cfgfile`*`]`	Use the specified configuration file.
`[-r` *`report`*`]` `[--twrfile` *`report`*`]`	Print the specified report file.
`[-L` *`localkey`*`]` `[--local-keyfile` *`localkey`*`]`	Use the specified local key file to verify the report file, if it was signed.

`[-t { 0|1|2|3|4 }]`
`[--report-level`
`{ 0|1|2|3|4 }]`
	Print the specified report level. This option helps control the amount of detail in a printout.

3.2.5 Test Email

Use Test mode to verify the operation of Tripwire software's email notification system. Tripwire software will use the email notification settings specified in the configuration file to send a test email message. If MAILMETHOD is set to SENDMAIL, the MAILPROGRAM value will be used.

```
./tripwire -m t
```

If email notification is working correctly, the address specified on the command line will receive the following message:

```
To: user@domain.com
From: user <user@domain.com>
Subject: Test email message from Tripwire

If you receive this message, email notification
from Tripwire is working correctly.
```

Test mode only tests email notification for the address specified on the command-line. It does not check for errors in the syntax used with the `emailto` attribute in the policy file.

Table 10: tripwire Test Mode Arguments

Argument	Meaning
`{ -m t | --test }`	Mode selector.
`-e` *`user@domain.com`* `--email` *`user@domain.com`*	Specifies email address to test.

3.3 twprint Command Overview

The `twprint` command has two functions: it prints report files and database files. These two functions have their own modes.

Tripwire database files are binary-encoded and signed. Tripwire report files are encoded, and may be signed if you prefer to sign them. The `twprint` application provides a way to view signed files in text form. (If you want to print a policy or configuration file, refer to `twadmin`.)

3.3.1 twprint Print Report Mode

The `twprint --print-report` mode prints the contents of a Tripwire report. If you do not specify a report with the `--twrfile` or `-r` command-line argument, the default report file specified by the configuration file REPORTFILE variable will be used.

The default filename for report files is $(HOSTNAME)-$(DATE).twr, where the $(DATE) variable includes the current time to the nearest second. Unless you used the Print Report mode within one second of a report's creation, `twprint` will be unable to find the report because the $(DATE) variable will have changed to reflect the current time. In other words, the time used is the start time of the `tripwire` execution. For this reason, when you want to print a report, use the Print Report mode argu-

ments to specify the report. On a machine named *LIGHTHOUSE*, the command would be:

```
./twprint -m r --twrfile LIGHTHOUSE-19990622-021212.twr
```

To help you manage the size and content of Tripwire reports, five new reports are available and may be specified in the configuration file:

- Single line report for email subject lines, logs, phones, or pagers (REPORTLEVEL=0)
- Parsable list of violated files (REPORTLEVEL=1)
- Summary (REPORTLEVEL=2)
- Concise report (REPORTLEVEL=3)
- Full report (REPORTLEVEL=4)

The default level is the Concise report, REPORTLEVEL=3. For examples of how reports look, see the Appendix.

3.3.2 twprint Print Database Mode

The `twprint` Print Database mode can print the contents of a Tripwire database to the screen, or redirect it to a file. You might want to do this to validate the data in your database, or, for forensic purposes, to validate what had been there.

```
./twprint --print-dbfile > db.txt
```

If no database is specified under the `--dbfile` command-line argument, the default database will be used. The default database is specified by the

DBFILE variable in the configuration file (either `tw.cfg`, or the configuration file specified under the `--cfgfile` command-line argument).

Table 11: twprint Print Database Mode Arguments

Argument	Meaning
`{ -m d | --print-dbfile }`	Mode selector.
`[-v]` `[--verbose]`	Verbose sends additional status information. Mutually exclusive with `--silent`.

`[-s]`
`[--silent]`
`[--quiet]`
	Silent suppresses additional status information.
	Mutually exclusive with `--verbose`.

`[-c` *`cfgfile`*`]` `[--cfgfile` *`cfgfile`*`]`	Use the specified configuration file. The default is `tw.cfg`.
`[-d` *`database`*`]` `[--dbfile` *`database`*`]`	Print the specified database file.
`[-L` *`localkey`*`]` `[--local-keyfile` *`localkey`*`]`	Use the specified local key file to read the database file.
`[-S` *`sitekey`*`]` `[--site-keyfile` *`sitekey`*`]`	Use the specified site key to read the configuration file.
`[`*`object1 object2`*`…]`	List of file system objects in the database to print. If not specified, every object in the the database file will be printed.

3.4 twadmin Command Overview

The `twadmin` utility provides the following functions:

- Replacing and printing configuration and policy files
- Encoding Tripwire files from text
- Decoding Tripwire files that have been binary-encoded
- Verifying Tripwire files
- Generating local and site keys

Each of these functions has its own mode, and some of them have more than one.

3.4.1 Replacing a Configuration File

This `twadmin` mode designates an existing text file as the new configuration file for Tripwire. The text configuration file must be specified on the command line. Using the site key, the new configuration file is encoded, signed, and saved with this command.

If you need to change the configuration after installation, you can generate a text copy of the current configuration file with the following command:

```
./twadmin --print-cfgfile > config.txt
```

When you have made the changes, create the encoded file again:

```
./twadmin --create-cfgfile --site-keyfile ../key/site.key config.txt
```

Table 12: twadmin Create Configuration File Mode Arguments

Argument	Meaning
`{ -m F | --create-cfgfile }`	Mode selector.
`[-v]` `[--verbose]`	Verbose sends additional status information. Mutually exclusive with `--silent`.

`[-s]`
`[--silent]`
`[--quiet]`
	Silent suppresses additional status information. Mutually exclusive with `--verbose`.

`[-S` *`sitekey`*`]` `[--site-keyfile` *`sitekey`*`]`	Use the specified site key file to sign the new configuration file. Mutually exclusive with the `--no-encryption` option. Either `--no-encryption` or `--site-keyfile` must be specified.
`[-c` *`cfgfile`*`]` `[--cfgfile` *`cfgfile`*`]`	Specify the destination configuration file.
`[-e]` `[--no-encryption]`	Do not sign the configuration file. Mutually exclusive with the `--site-keyfile` and `--site-passphrase` options. Either `--no-encryption` or `--site-keyfile` must be specified.
`[-Q` *`passphrase`*`]` `[--site-passphrase` *`passphrase`*`]`	Specifies passphrase to be used with site key for signing the configuration file. Valid only with `--site-keyfile`.
`configfile.txt`	Specifies the text configuration file that will become the binary-encoded, signed configuration file.

3.4.2 Printing a Configuration File

After a configuration file has been created, it is stored in a binary-encoded form. The `twadmin --print-cfgfile` command provides a way of printing the current contents of the configuration file in a readable text format.

Table 13: Print Configuration File Arguments

Argument	Meaning
`{ -m f | --print-cfgfile }`	Mode selector.
`[-v]` `[--verbose]`	Verbose sends additional status information. Mutually exclusive with `--silent`.

`[-s]`
`[--silent]`
`[--quiet]`
	Silent suppresses additional status information.
	Mutually exclusive with `--verbose`.

`[-c` *`cfgfile`*`]` `[--cfgfile` *`cfgfile`*`]`	Print the specified configuration file.

3.4.3 Replacing a Policy File

This mode enables you to replace or create policy files. (Using `twadmin --create-polfile` requires a database re-initialization, because the records in the old database will no longer match the rules specified in the policy file.)

Table 14: Create Policy File Arguments

Argument	Meaning
`{ -m P | --create-polfile}`	Take the specified text file and store it as a binary-encoded Tripwire policy file.
`[-v]` `[--verbose]`	Verbose sends additional status information. Mutually exclusive with `--silent`.

`[-s]`
`[--silent]`
`[--quiet]`
	Silent suppresses additional status information. Mutually exclusive with `--verbose`.

`[-c` *`cfgfile`*`]` `[--cfgfile` *`cfgfile`*`]`	Use the specified configuration file.
`[-S` *`sitekey`*`]` `[--site-keyfile` *`sitekey`*`]`	Use the specified site key file to sign the new policy file. Mutually exclusive with `--no-encryption`.
`[-p` *`polfile`*`]` `[--polfile` *`polfile`*`]`	Specifies the destination policy file.
`[-e]` `[--no-encryption]`	Does not sign the policy file. The policy file will still be stored in a binary-encoded form and will not be human-readable. Mutually exclusive with `--site-keyfile` and `--site-passphrase`.

`[-Q` *`passphrase`*`]`
`[--site-passphrase`
`passphrase``]`
	Use the specified passphrase with the site key to sign the policy file. Mutually exclusive with `–no-encryption`.

`policyfile.txt`	Specifies the text policy file that will become the new binary-encoded and signed policy file.

3.4.4 Printing a Policy File

After a policy file has been created, it is stored in a binary-encoded form. This command mode provides a way of printing out the current contents of the policy file in a human-readable text format.

Table 15: Print Policy File Arguments

Argument	Meaning
`{ -m p | --print-polfile }`	Mode selector.
`[-v]` `[--verbose]`	Verbose sends additional status information. Mutually exclusive with `--silent`.

`[-s]`
`[--silent]`
`[--quiet]`
	Silent suppresses additional status information.
	Mutually exclusive with `--verbose`.

`[-c` *`cfgfile`*`]` `[--cfgfile` *`cfgfile`*`]`	Use the specified configuration file.
`[-p` *`polfile`*`]` `[--polfile` *`polfile`*`]`	Print the specified policy file.
`[-S` *`sitekey`*`]` `[--site-keyfile` *`sitekey`*`]`	Use the specified site key file.

3.4.5 Removing Signatures from a File

This mode enables you to remove signatures from signed configuration, policy, database, and report files. You will need to enter the appropriate local or site passphrase, or both if a combination of files is to be modified. After the signature has been removed, these files will still be in a binary-encoded form that is human-readable. This operation examines the type of each file and uses the appropriate key to remove signing from it. The file will then be rewritten in a binary-encoded format.

Table 16: twadmin Removing Encryption Arguments

Argument	Meaning
`{ -m R | --remove-encryption }`	Mode selector.
`[-v]` `[--verbose]`	Verbose sends additional status information. Mutually exclusive with `--silent`.

`[-s]`
`[--silent]`
`[--quiet]`
	Silent suppresses additional status information.
	Mutually exclusive with `--verbose`.

`[-c` *`cfgfile`*`]` `[--cfgfile` *`cfgfile`*`]`	Use the specified configuration file.
`[-L` *`localkey`*`]` `[--local-keyfile` *`localkey`*`]`	Specify the local key file to use to remove the signature for database files and reports.
`[-S` *`sitekey`*`]` `[--site-keyfile` *`sitekey`*`]`	Specify the site key file to use to remove the signature for configuration and policy files.

`[-P` *`passphrase`*`]`
`[--local-passphrase`
`passphrase``]`
	Specify passphrase to use with the local key file when removing signatures from database files and reports.

`[-Q` *`passphrase`*`]`
`[--site-passphrase`
`passphrase``]`
	Specify passphrase to use with the site key file when removing signatures from configuration and policy files.

`file1` `[`*`file2`*`…]`	List of Tripwire files for which the signature is to be removed. At least one file must be specified. Multiple files should be separated by spaces. Wildcards can be used to specify files, although wildcard use is discouraged for security reasons.

3.4.6 Signing a File

This mode enables you to sign unsigned configuration, policy, database, or report files. You can specify multiple files, although you should specifically name them on the command line rather than use a wildcard. The files will be signed using either the site or local key, as appropriate. To automate the process, you can include the passphrase for the key files on the command line. Each file type will be examined and the appropriate key (local key for databases and reports; site key for configuration and

policy files) will be used to sign it. The key file may be overridden using the `--site-keyfile` or `--local-keyfile` arguments.

Table 17: twadmin Encryption Mode Arguments

Argument	Meaning
`{ -m E | --encrypt }`	Mode selector.
`[-v]` `[--verbose]`	Verbose sends additional status information. Mutually exclusive with `--silent`.

`[-s]`
`[--silent]`
`[--quiet]`
	Silent suppresses additional status information.
	Mutually exclusive with `--verbose`.

`[-c` *`cfgfile`*`]` `[--cfgfile` *`cfgfile`*`]`	Use the specified configuration file.
`[-L` *`localkey`*`]` `[--local-keyfile` *`localkey`*`]`	Specify the local key file to use to sign database files and reports.
`[-S` *`sitekey`*`]` `[--site-keyfile` *`sitekey`*`]`	Specify the site key file to use to sign configuration and policy files.
`[-P` *`passphrase`*`]` `[--local-passphrase` *`passphrase`*`]`	Specify passphrase to be used with the local key file.
`[-Q` *`passphrase`*`]` `[--site-passphrase` *`passphrase`*`]`	Specify passphrase to be used with the site key file.
`file1` `[`*`file2`*`…]`	List of Tripwire files to sign using the site or local key, depending on the file type. Multiple files should be separated by spaces. Wildcards can be used to specify files, although wildcard use is discouraged for security reasons.

3.4.7 Getting the Encryption Status of a File

This mode enables you to examine specific files and report their encryption status. This report displays the following information:

- Filename
- File type, and whether binary-encoding is used
- Whether or not a file is signed
- What key, if any, was used to sign it

For each file specified, a report will be given displaying whether or not it is signed, and, if it is signed, with what key. Signing type for files will be determined by a trial and error method, using first the site key, and then the local key.

Table 18: twadmin Examine Encryption Mode Arguments

Argument	Meaning
`{ -m e | --examine }`	Mode selector.
`[-v]` `[--verbose]`	Verbose sends additional status information. Mutually exclusive with `--silent`.

`[-s]`
`[--silent]`
`[--quiet]`
	Silent suppresses additional status information.
	Mutually exclusive with `--verbose`.

`[-c` *`cfgfile`*`]` `[--cfgfile` *`cfgfile`*`]`	Use the specified configuration file.
`[-L` *`localkey`*`]` `[--local-keyfile` *`localkey`*`]`	Specify the key to use as the local key when examining database or report files.
`[-S` *`sitekey`*`]` `[--site-keyfile` *`sitekey`*`]`	Specify the key to use as the site key when examining policy or configuration files.
`file1` `[`*`file2`*`…]`	List of Tripwire files to examine. Wildcards can be used to specify files, although wildcard use is discouraged for security reasons.

3.4.8 Generating Keys

This mode enables you to create site or local keys for Tripwire files. Although the installation process generates site and local keys, you can use this command to regenerate keys at any time. This might be advisable if you have staff changes or if you suspect that the keys may have been compromised. The site and local keys may be generated simultaneously, or one at a time with two separate invocations of `twadmin`.

If you overwrite a site or local key, any files signed with that key become permanently unusable. Tripwire Security Systems cannot help you recover such files.

When choosing passphrases for keyfiles, remember that effective passphrases should be at least 8 characters in length, should not be words, and should not use quotes. See Chapter 2 for detailed passphrase rules.

Table 19: twadmin Generate Key Mode Arguments

Argument	Meaning
`{ -m G | --generate-keys }`	Mode selector.
`[-v]` `[--verbose]`	Verbose sends additional status information. Mutually exclusive with `--silent`.

`[-s]`
`[--silent]`
`[--quiet]`
	Silent suppresses additional status information.
	Mutually exclusive with `--verbose`.

`[-L` *`localkey`*`]` `[--local-keyfile` *`localkey`*`]`	Generate keys into the specified file. At least one key file, `--local-keyfile` or `--site-keyfile`, must be specified.
`[-S` *`sitekey`*`]` `[--site-keyfile` *`sitekey`*`]`	Generate keys into the specified file. Either `--local-keyfile` or `--site-keyfile` must be specified.
`[-P` *`passphrase`*`]` `[--local-passphrase` *`passphrase`*`]`	Specify passphrase to be used when generating a local key.
`[-Q` *`passphrase`*`]` `[--site-passphrase` *`passphrase`*`]`	Specify passphrase to be used when generating a site key.

3.5 Siggen Command Overview

You can use the `siggen` utility to display hashes for any specified file. This utility displays some or all of the signature values for any specified file(s) in base64 notation, as described in RFC 1521. This is a different base 64 notation than that used by Tripwire 1.2 or 1.3 academic source releases, so signature values for the same file will appear different between the academic source and commercial release versions.

Table 20: siggen Display Signature Arguments

Argument	Meaning
`[-t]` `[--terse]`	Terse mode. Prints requested signatures for a given file on one line, delimited by spaces, one file per line.
`[-h]` `[--hexadecimal]`	Display results in hexadecimal rather than base64 notation.
`[-a]` `[--all]`	Display all signature function values (default).
`[-C]` `[--CRC32]`	Display CRC-32, POSIX 1003.2 compliant 32-bit Cyclic Redundancy Check.
`[-M]` `[--MD5]`	Display MD5, the RSA Data Security, Inc.® Message Digest Algorithm.
`[-S]` `[--SHA]`	Display SHA value.
`[-H]` `[--HAVAL]`	Display HAVAL value, a 128-bit signature code.
`file1` `[`*`file2`*`…]`	List of filesystem objects for which values should be shown. Wildcards can be used to specify files, although wildcard use is discouraged for security reasons.

3.5.1 Hash Throughput Performance

The following graph summarizes the mean throughput realized by the four hashes on a 100 MB file, on a platform featuring a 200 MHz Motorola Power3 processor, with 256 MB RAM, running AIX 4.3. These numbers give you an idea of how the different hashes may perform. Throughput depends upon many factors, and your results may be different, depending on file size, processor type, processor load, memory, and operating system version.

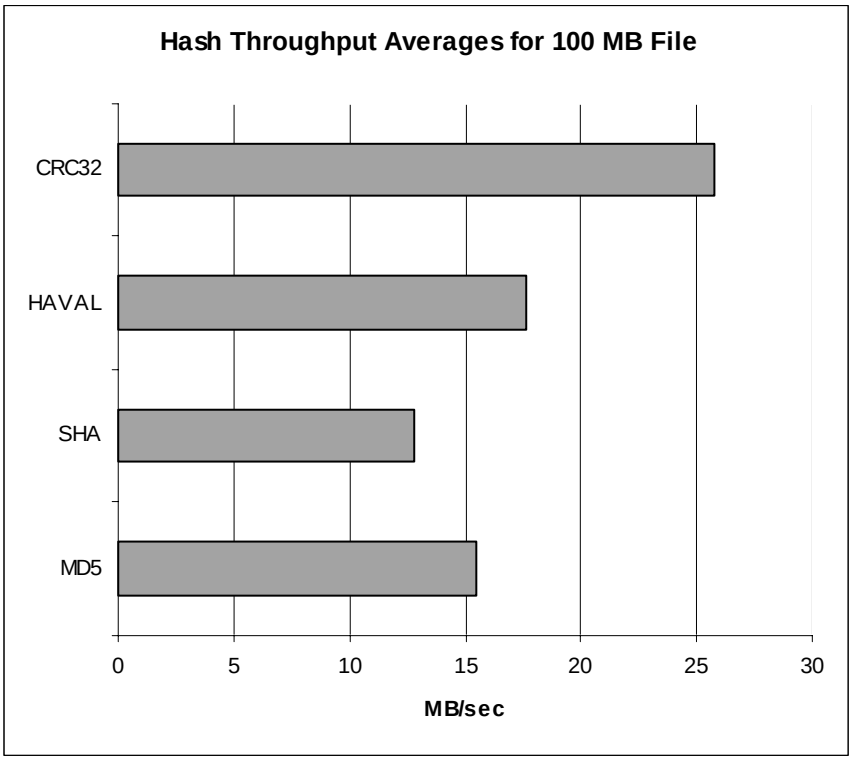

Figure 7: Hash Throughput Approximations

4

Policy Reference

4.0 Overview

The policy file describes which system objects Tripwire should monitor. This chapter defines policy file components. If you need information about the policy update mode itself, please see “tripwire Policy Update Mode” on page 57.

In Tripwire 2.2.1, objects are defined as files and directories. Each object has an *object name* that identifies it. A *property* refers to a single characteristic of an object that can be monitored by the Tripwire software. A *rule* specifies a group of properties to be checked for a given object. *Directives* control conditional processing of sets of rules in a policy file. Comments, rules, directives, and variables are the standard components of the policy file, and rules directly govern how Tripwire software checks the integrity of your system.

Policy files that are used for both UNIX and NT systems should have all sections explicitly labelled with the FS, NTFS, or NTREG arguments. Refer to the NT documentation for more information.

4.1 Rules

Policy rules determine whether and to what extent Tripwire will check the integrity of particular files and directories. Tripwire recognizes two kinds of policy rules:

- *Normal rules* define which properties of a particular file or directory tree the Tripwire executable should scan.
- *Stop points* tell Tripwire not to scan a particular file or directory.

Each of these policy rules is described in detail below. The operation of a set of policy rules is unaffected by the order in which the rules appear.

4.1.1 Normal Rules

A normal rule associates a system object with a property mask. The syntax for a normal rule is:

```
object_name -> property_mask;
```

An object name is the fully qualified pathname for a directory or file, and the property mask specifies what properties of the object to examine or ignore. The `->` token separates the object name and the property mask, must have a space preceding the `->`, and a semicolon must terminate the rule. If the pathname specified is a directory, the directory and all of its descendants will be scanned with the indicated property mask. If the pathname refers to an individual file, only that file will be scanned with the specified mask. If an object has more than one rule in a policy file, Tripwire will print an error message and exit without scanning any files.

```
# Defines tripwire behavior for the entire /bin directory tree.
/bin               ->  $(ReadOnly) ;

# Defines tripwire behavior for a single file.  In this case,
# Tripwire watches almost all properties of hostname.hme0.
/etc/hostname.hme0 ->  $(IgnoreNone) -ar;

# Scan the entire /etc directory tree using mask1, except
# the file /etc/passwd, which should be scanned using mask2.
/etc               ->  $(mask1)  ; #mask1 is user-defined
/etc/passwd        ->  $(mask2)  ; #mask2 is user-defined
```

Figure 8: Examples of Normal Rules

Object names

In the policy file, object names are the absolute pathnames of directories and files. Environment variables may not be used in object names for security reasons.

```
/etc              # valid
/etc/passwd       # valid
$HOME             # not valid
```

Figure 9: Express Object Names with Absolute Pathnames

The following characters are not allowed in unquoted object names: exclamation point (!), braces ({ or }), greater-than sign (>), parentheses ( ), newline (\n), tabs (\t), spaces ( ), commas (,) , semicolons (;), equal sign (=), dollar sign ($), pound (#), vertical bar (|), backslash (\), single quote (‘), and plus sign (+).

Because object names may contain characters that are not allowed on the left-hand side of rules, Tripwire supports quoted object names. Object name quoting must be used if Kanji, Kana, or any other double-byte characters appear in the object name. Object names with these characters must be double-quoted:

```
"\/mysubdirectory\/kanji_characters"
```

Alternatively, the following format can also be used:

```
/mysubdirectory/"kanji-text"
```

Object names are concatenated; white space inserted between or within object names is ignored unless the filename is inside a quoted string. Quotes are also ignored, unless inside a quoted string and preceded by a backslash. This allows more flexible handling of variable substitution and quoting.

Therefore, all of the following rules are equivalent:

```
/usr/local          ->          $(ReadOnly) ;
/usr /local         ->          $(ReadOnly) ;
"/usr" "/local"     ->          $(ReadOnly) ;
/usr / local        ->          $(ReadOnly) ;
```

Figure 10: Equivalent Ways to Express Rules

Filenames can contain escape sequences inside quoted strings to handle unprintable characters. The escaped sequences are interpreted in the same way as in the C++ language. The following examples define allowable sequences:

- Octal numbers \412 (1, 2, or 3 octal digits)
- Hex numbers \x2A (2 hex digits)
- Characters: \t, \v, \b, \r, \f, \a, \\, \?, \', and \"
- All other escaped characters are treated as if not escaped.

```
/test           # "/test"
"/te\x73t2"     # "/test2"
"/te\163t3"     # "/test3"
/tes\t          # Invalid:escape sequences must be in double quotes
```

Figure 11: Using Quoted Strings in Object Names

Managing Recursion Across Mount Points with Rules

Tripwire software recurses into directories, but only within the current file system; it does not cross mount points or into subdirectories that have a different device number (`st_dev`) as returned from `lstat(2)`.

For example, if `/usr/local` is a mount point, then the rule

```
/usr        ->          +pinugsmc-a;
```

would cause all of `/usr` to be scanned, except for the directory tree located at `/usr/local`. If the goal is to scan `/usr` in its entirety, including `/usr/local`, you should specify the following rules:

```
/usr       ->         +pinugsmc-a;
/usr/local ->         +pinugsmc-a;
```

Property masks

Property masks describe object properties to examine or ignore. The following regular expression describes the correct syntax:

```
([+-]*[pinugtsldbamcrCMSH])+
```

See “Property Mask Characters” on page 81 for the list of checked properties and what they do.

The following general principles apply when using property masks:

- Only the specified Tripwire properties are examined. Operating system properties are not checked.
- You cannot specify an empty property mask. The property mask must include one or more property symbols.
- Plus character turns *on* a property; minus character turns *off* a property. You cannot specify a property mask that consists only of plus and minus characters. Not specifying a property in the mask and explicitly turning it off with a minus sign are equivalent.
- When property symbols appear in a property mask without any preceding plus or minus sign, then plus is assumed. All three of these property masks are equivalent:

```
+p+n+s; # compare permissions, number of links, and file size
+pns;   # same as above
pns;    # same as above
```

- Once a plus or minus appears in the selection mask, it applies to all successive properties until another plus or minus appears. The minus

sign becomes most useful when you use variables to specify part of the mask. For example:

```
mask1  =  +pinug ;            # define a variable called 'mask1'
/file   ->   $(mask1)-g ;     # use the mask defined by 'mask1', but
                              # turn off property 'g'
```

- In cases of duplicate or contradictory symbols, only the last symbol is acted upon.
- You can specify a property mask in which no properties are turned on. This is equivalent to using the predefined variable $(IgnoreAll) and is useful for monitoring only the deletion or addition of files.

Tripwire only references the properties shown in this table; it does not use operating-system dependent properties in this release.

Table 21: Property Mask Characters

Property	Meaning
-	Ignore the following properties
+	Record and check the following properties
p	File permissions
i	Inode number
n	Number of links (i.e., inode reference count)
u	User id of owner
g	Group id of owner
t	File type
s	File size

Table 21: Property Mask Characters

l	Indicates that the file is expected to grow. If the file is smaller than the last recorded size, it is a violation of this property. This can be useful for log files. However: If a file grows from size A to size B, where B>A, no violation is reported and the database is not updated. The most recent information in the database is that the file has size A. If the file then shrinks in size from B to C, where A<C<B, no violation is reported because C is still larger than A. Without explicitly updating the database, these violations cannot be reported, despite specifying this property.
d	Device number of the disk on which the inode associated with the file is stored
r	Device number of the device to which the inode points. Valid only for device objects.
b	Number of blocks allocated
a	Access timestamp The +a property is incompatible with the hash properties (+CMSH). To calculate the hash, the file must be opened and read, which changes the access timestamp. Specifying any of +CMSH will always cause a violation of the +a property. Since checking a directory's contents changes access timestamp, specifying +a in a directory rule will always cause a violation for the +a property during the next integrity check. To avoid this behavior, use `recurse = false` in the rule attribute, set `LOOSEDIRECTORYCHECKING` = `true` in the configuration file, or add `-a` to the rule.
m	Modification timestamp
c	Inode creation/modification timestamp
C	CRC-32, POSIX 1003.2 compliant 32-bit Cyclic Redundancy Check. Choose this hash for relatively high performance but relatively low security.

Table 21: Property Mask Characters

M	MD5, the RSA Data Security, Inc.® Message Algorithm Choose this hash for high security.
S	SHA, part of the SHS/SHA algorithm. Choose this hash for high security.
H	HAVAL, a strong 128-bit signature algorithm Choose this hash for high security.

The security hashes CRC-32, MD5, SHA, and HAVAL, are often best used in pairs. Customers have found that specifying all four hashes slows throughput significantly and makes frequent integrity checks laborious. See the figure for hash throughput approximations for a comparison of relative hash performance.

4.1.2 Stop Points

Use stop points to bypass certain directories or files. The syntax is:

```
! objectname ;
```

As with normal rules, the object name is the fully qualified pathname for a directory or file. A semicolon must terminate the stop point rule.

Consider the case where a policy rule has been specified for `/etc`. The entire `/etc` directory tree will be scanned recursively. Using stop points, you can bypass particular files in the `/etc` hierarchy.

```
# Scan all of /etc recursively, but do not scan two particular
# files present in the /etc hierarchy.
/etc    ->   $(ReadOnly) -ar;
!/etc/rc.d;        # ignore startup directory
!/etc/mnttab;      # ignore dynamic listing of mounted filesystems
```

Figure 12: Bypassing Specific Files with Stop Points

4.2 Rule Attributes

Rule attributes provide additional information or further modify Tripwire behavior. For example, if a policy rule is broken, the `emailto` rule attribute can be used to specify an email recipient. Rule attributes use the syntax:

```
(attribute_name = attribute_value,...)
```

Attributes must be enclosed in parentheses ( ) and separated by a comma.

Rule attributes can take only one argument. To specify more than one address with the `emailto` rule attribute, the entire semicolon-delimited list of addresses must be quoted:

```
(emailto="one@machine.com;two@machine.com")  # Valid
(emailto=one@machine.com two@machine.com)    # No quote or semicolon
(emailto="one@machine.com";"two@machine.com")# Too many quotes
```

Attribute names are case-insensitive. You cannot specify attributes for a stop point rule or set an attribute without a rule. Attributes are hard-coded in Tripwire software; the following attributes are currently supported:

Table 22: Tripwire Attributes

Attribute	Meaning
rulename	A literal string to be associated with a rule. The default value is the last element of the objectname to which the rule applies.
emailto	Email address(es) to which notification of any violations is sent. The default value is none.
severity	Severity level associated with rule.The default value is 0. Range is 0 to 1,000,000.
recurse	Controls recursive scanning of directories. The default value is true.

Rule attributes are associated with individual normal rules according to the following syntax:

```
objectname -> propertymask (attribute-list) ;
```

For example:

```
/usr/lib    ->     $(ReadOnly) ( emailto = admin@foo.com ) ;
```

Rule attributes can also be specified for a group of rules. For example:

```
( emailto = admin@foo.com )  # Violations of the rules will be sent
{                             # to admin@foo.com
    /usr/lib    ->      $(ReadOnly) ;
    /usr/sbin   ->      $(IgnoreNone) ;
}
```

The following two sets of rules (single and scoped, respectively) are equivalent:

```
/usr/lib    ->    +pinug    (emailto = admin@foo.com) ; # A single rule
/usr/bin    ->    +pinug    (emailto = admin@foo.com) ; # Another rule

(emailto = admin@foo.com)
{                             # Scope of emailto delimited by braces
    /usr/lib    ->  +pinug ;
    /usr/bin    ->  +pinug ;
}                             # End of rule
```

Figure 13: Single and Scoped Rules Using emailto Attribute

You may specify a stop point inside a scoped rule block:

```
/usr/lib    ->  +ping   (emailto = admin1@foo.com) ;
/usr/bin    ->  +ping   (emailto = admin2@foo.com) ;

(emailto = admin3@foo.com)
{                             #begin scoped rule block
    /usr/tab    ->  +ping;
    /usr/lib    ->  +tbm;
    !/usr/lib/abc       ;     #stop point defined
}                             #end scoped rule block
```

Figure 14: Stop Points in Scoped Rule Blocks

When a scoped attribute and a single attribute apply to the same rule, the value of the single attribute replaces that of the scoped attribute. The only exception to this precedence is the `emailto` attributes; these are additive. For example, the following two examples are equivalent in operation:

```
# Scoped and single attributes applied to the same rule, example 1
(emailto = admin1@foo.com, severity = 90, rulename = dog, recurse = 3)

{                               #begin scoped rule block

  /etc/dog->+pingus (severity = 75, emailto = admin2@foo.com);
  /etc/cat->$(Dynamic) (rulename = cat, recurse = true);

}                               #end scoped rule block
```

An equally valid way to express this would be:

```
# Scoped and single attributes applied to the same rule, example 2
    /etc/dog    ->  +pingus
        (   emailto = "admin1@foo.com; admin2@foo.com",
            severity = 75,
            rulename = dog,
            recurse = 3);    #note closing parentheses and semicolon
    /etc/cat    ->  $(Dynamic)
        (   emailto = admin1@foo.com,
            severity = 90,
            rulename = cat,
            recurse = true);
```

4.2.1 rulename

The `rulename` attribute is used to associate a symbolic name with one or more rules. This ability can be used to provide additional information in the report file. For example:

```
/home/.login   ->   $(ReadOnly) (rulename=rcfiles) ;
/home/.cshrc   ->   $(ReadOnly) (rulename=rcfiles) ;
/home/.logout  ->   $(ReadOnly) (rulename=rcfiles) ;
```

Figure 15: Grouping Rules with Rulename Attribute

These three lines associate the symbolic name, *rcfiles*, with the three objects named in the rules. In a report file, the results for these three rules would be flagged as originating from rules named *rcfiles*. This feature is useful if you wish to track certain objects within a large Tripwire database. For example, important files in different directories can be tagged with a unique rule name, such as rulename=*watchme*. You can then run `tripwire` and interpret your data later using the rule name *watchme* as a sorting key.

4.2.2 emailto

The `emailto` attribute enables you to associate one or more email addresses with a rule. When you run an integrity check with the `--email-report` option and rule violations are found, Tripwire sends a summary of those violations in email, using the command-line arguments or the settings in the configuration file if you did not specify arguments. For each rule, only those persons specified with the `emailto` attribute will receive email notification if that rule is violated. If the `emailto` attribute is nested in the policy file, all the recipients will receive email. For example:

```
(emailto   =   admin1)
{
    /bin    ->  $(ReadOnly)    ;
    (emailto=  admin2)
    {
        /tmp/bob        ->      +pinug  ;
        /tmp/bill       ->      $(ReadOnly)  (emailto = admin3);
    }
}
```

Figure 16: Nested emailto Attributes

If the rule `/tmp/bob` is violated, both *admin1* and *admin2* will receive email. If the rule `/tmp/bill` is violated, *admin1*, *admin2*, and *admin3* will all receive email.

To specify multiple email addresses for a single policy rule, include them as a double-quoted, semicolon-delimited list. The names themselves can contain white spaces, but no leading or trailing white spaces should appear.

For example:

```
/bin -> $(ReadOnly)(emailto=admin@foo.com ) ;                #valid
/etc -> $(ReadOnly)(emailto="admin@foo.com;admin2@foo.com" ) ; #valid
```

The `emailto` attribute uses the settings specified by the email notification variables in the configuration file. Email is sent only if the `--email-report` argument of the `tripwire` command is specified. The maximum `severity` value of the rules violated will be reported in the subject line of the email, as well as the number of files or directories added, removed, or changed. If severity is not explicitly set, it defaults to zero and will be reported in email.

4.2.3 severity

This attribute associates a user-assignable severity level with a rule. The default severity value is 0. Tripwire also enables you to specify a “High” (100), “Medium” (66), and “Low” (33) severity for convenience. You can name severity levels as well. The range of valid entries is 0 to 1,000,000.

When you run an integrity check, you can specify that only rules exceeding a certain severity level be used. For example, in the policy file, the following statement establishes a severity level for violations in `/usr/lib`:

```
# In the policy file:
/usr/lib  ->  $(ReadOnly) (severity=80) ;
```

Rules which do not have an explicit severity level set in the policy file have an implicit severity level of 0. If you want the report to show any rules whose importance to you is at least “60” or above, the command line would be:

```
./tripwire --check --severity 60 #the rule above will be run
```

4.2.4 recurse

The `recurse` rule attribute controls the way that Tripwire monitors directories. Valid entries for `recurse` are `true`, `false`, or a number from -1 to 1,000,000.

For rules that refer to a directory, if `recurse` is set to `true` (or -1), the `tripwire` operation will recursively scan the entire contents of the directory, including both files and subdirectories. When `recurse` is set to `false` (or 0), tripwire will scan the inode corresponding to the directory, but none of the files or subdirectories in the directory. For positive recurse value *n*, the rule will monitor all objects up to *n* levels below the start point.

For example, if a policy file contains the rule:

```
/temp/dog -> $(ReadOnly)(recurse = 2);
```

then all of the contents of `/temp/dog` will be monitored, but the file `/temp/dog/shepherd/bark/loudly.txt` will not. Stop points still apply; adding a stop point for `/temp/dog/retriever` would prevent that directory from being checked.

4.3 Directives

Tripwire supports a small set of preprocessor-like directives that allow conditional interpretation of the policy file and perform certain diagnostic and debugging operations. The primary intent of this mechanism is to support sharing a policy file among multiple machines. Directives have the following syntax:

```
@@directive-name [arguments]
```

White space may precede or follow the @@ construct, but non-white space characters may not appear on the line before the @@ construct, nor may any characters intervene between the two @ characters. Directive

names are case-sensitive. Directives cannot be derived by variable expansion. For example:

```
machine = spock;                #define variable machine
@@ifhost  $(machine)            #If the host is the machine spock...


IFHOST = ifhost;
@@ $(IFHOST)spock               # Invalid use of directive & variable
```

Figure 17: Valid and Invalid Use of Directives

The following directives are supported:

Table 23: Tripwire Directives

Directive	Meaning
@@section	Designates a section of the policy file. Used for NT systems, but does not cause errors in UNIX installations.
@@ifhost @@else @@endif	Allow conditional interpretation of the policy file.
@@print @@error	Prints a message to *stdout*. Prints a message to *stdout* and exits.
@@end	Marks the logical end-of-file.

The @@section GLOBAL directive is used to designate a section of the policy file for the definition of global variables. Global variables have scope throughout the policy file, while user-defined variables defined in one section only have scope within that section. If a global variable and a local variable have the same name, the local variable definition will take precedence in its section, temporarily masking the global variable. The default policy file uses the GLOBAL section to store the paths of tripwire files. Another use for this section in UNIX might be to store email addresses.

4.3.1 Conditional Interpretation

The @@ifhost, @@else, and @@endif directives are used to allow conditional interpretation of a policy file. Conditional directives have the following syntax:

```
@@ifhost host1 || host2 || …

@@else

@@endif
```

where *host1*, *host2*, … are unqualified hostnames, and the || construct is interpreted as the logical OR operation.

The following example illustrates how you might use directives to use one policy file with multiple hosts:

```
@@ifhost spock || kirk
/bin       ->      $(ReadOnly) ;
@@endif

@@ifhost chekov || uhura
/usr/bin   ->      +pinug ;
@@else
/usr/bin   ->      +pinugsmC ;
@@endif
```

Figure 18: Applying One Policy File to Several Hosts

If the unqualified hostname of the machine running Tripwire matches any of the hosts listed in the @@ifhost directive, all the lines between the @@ifhost and the matching @@endif are interpreted.

If there is no match, any lines between the @@ifhost and @@endif are skipped. However, if there is an @@else in those skipped lines, any lines between the @@else and @@endif are interpreted. There is no @@elseif directive.

Note that only the logical OR operation is supported.

The @@ifhost and @@else directives can be nested. The syntax would be:

```
@@ifhost chekov || uhura

    RULE1

    @@else

    @@ifhost bones

    RULE2

    @@endif

@@endif
```

4.3.2 Message Reporting

The @@print and @@error directives are intended for debugging and remote diagnostics. The syntax for these directives is:

```
@@print string

@@error string
```

You can only use one string as a parameter with these directives, so quote them if you want to include spaces in the message.

```
@@print string               # Valid
@@print "Two strings"        # Valid
@@print two strings          # Invalid
```

The @@print directive prints its arguments to *stdout*, while the @@error directive prints its arguments to *stdout* and then causes the calling program to exit with a status of 1.

4.3.3 Indicating End-of-File

The @@end directive marks the logical end of the policy file. Any text appearing after this directive will be ignored by Tripwire. The @@end directive may not appear within a scoped rule attribute block or an @@ifhost, @@else, or @@endif block.

4.4 Variables

Variables can be defined anywhere between rules, using the following syntax:

```
variable  =  value  ;
```

Variable names are case-sensitive, and may contain all alphanumeric characters, underscores, the characters plus (+), minus (-), at sign (@), colon (:), ampersand (&), per cent sign (%), hat (^), and period (.). The regular expression for variable names is:

```
[A-Za-z0-9+\-@:&%^\.]+
```

Variable Scope

The @@section directive designates a section of the policy file for global variables. The scope of a variable begins at the point where it is defined and extends to the end of the file. Examples of variable definition are:

```
path    =  /usr/local/lib/bigproject;
mask1   =  +pinugC-a;
```

Variable substitution is valid anywhere that a string could appear. The syntax for variable substitution is:

```
$(variable)
```

Variables may be used on the lefthand side of rules:

```
# Define the variable…
path   =   /usr/local/lib/bigproject  ;
# …and now use it.
$(path)/src   ->   +pug  ;
$(path)/exe   ->   +pugntmc  ;
```

Variables may also be used on the righthand side of rules:

```
# Define the variable…
mask1   =   +pinugC-a   ;
# …and now use it.
/home/projectA   ->   $(mask1)  ;
/home/projectB   ->   $(mask1)+MSH-db  ;
```

Variables may be used in directives:

```
#Define a machine
server = jupiter;

@@ifhost $(server)
@@end
```

However, tokens cannot be included in variable substitutions, and therefore the following would not work (the "||" construct is a token).

```
# Define a variable listing all machines in sales.
sales_department  =  jupiter || mars || pluto || mercury;

@@ifhost $(sales_department)     # ERROR
@@endif
```

Variables may not replace literal tokens, directives, or predefined variables.

Table 24: Predefined Variables

Variable	Use
ReadOnly	This variable is good for files that are widely available but are intended to be read-only. Expands to: `+pinugsmtdbCM-raclSH`
Dynamic	This variable is good for monitoring user directories and files that tend to change frequently. Expands to: `+pinugtd-rsacmblCMSH`
Growing	This variable is useful for files that can grow, but not shrink, such as log files: Expands to: `+pinugtdl-rsacmbCMSH`
IgnoreAll	This variable tracks a file's presence or absence, but doesn't check any other properties. Expands to: `-pinusgamctdrblCMSH`
IgnoreNone*	This variable turns on all properties and provides a convenient starting point for defining your own property masks. Expands to: `+pinusgamctdrbCMSH-l`
Device	This variable is useful for devices or other files that Tripwire should not attempt to open. Expands to : `+pugsdr-intlbamcCMSH`

*A recommended usage for IGNORENONE is to specify it as

```
$(IGNORENONE) -ar
```

to prevent spurious access time violations.

Glossary

Attribute

Attributes modify Tripwire behavior on a per-rule basis.

Command modes

Command modes are functionally distinct aspects of Tripwire commands. Command modes define which command-line arguments are valid.

Configuration file

The file that stores information and settings, such as the location of data files, that Tripwire requires to function properly. By default, the configuration file `tw.cfg` is located in the Tripwire `/bin` directory, and is encoded and signed with the site key file. It is not the same file as the installation's configuration file.

CRC-32

Cyclic Redundancy Check algorithms are fast, robust, and provide reliable detection of errors associated with data transmission. 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. For three runs on files 100MB in size, its throughput mean was 25.795 MB/sec, on a 200MHz Motorola Power3 processor with 256 MB RAM running AIX 4.3.

Create Configuration File mode

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

Create Policy File mode

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

Cryptographically signed file

A file that has been encoded in a binary format and whose hash has been encrypted. The expression is often abbreviated to “signed.”

Database file

A Tripwire file representing the expected properties of monitored files. The database must be created from a system in a known secure state for Tripwire software to operate as expected. The database file is encoded and signed with the local key file, and is in the location specified by the DBFILE variable in the configuration file.

Database Initialization mode

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

Database Update mode

A command mode of the `tripwire` command that updates the objects in the Tripwire database file with object properties as recorded in a report file.

Encryption mode

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

Examine Encryption mode

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

File integrity assessment

A technology in which message digest hashing algorithms are used to render files and directories tamper-evident.

Generate Keys mode

A command mode of the `twadmin` command that generates the site key and local key that Tripwire uses to sign and verify Tripwire files.

Growing file

A file that usually increases in size during normal operations. Log files are an example of growing files.

Hash

See *message digest algorithm.*

HAVAL

HAVAL is a message-digest algorithm that was written by Yuliang Zheng at the University of Wollongong, and is described in:

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

For three runs on files 100MB in size, its throughput mean was 17.655 MB/sec, on a 200MHz Motorola Power3 processor with 256 MB RAM on an AIX 4.3.

Host-based intrusion detection

A strategy for detecting intrusions or policy violations by collecting information about an individual host, or system.

Integrity assessment

Establishes that local system *objects* are the same as they are expected to be by comparing a "known good" copy of the object to

the current system object, monitoring changes to the files that govern the host system's environment.

Integrity Check mode

A command mode of the `tripwire` command that compares the current state of the file system against the values stored in the Tripwire database file, and writes violations to a report file.

Key file

Stores the public and private keys for signature functions. Tripwire has two key files, the site key file and the local key file, which are used to sign critical Tripwire files. If the site and local key files are overwritten or otherwise destroyed, any files signed with those keys will be rendered unusable.

Key pair

A public key and a private key stored together to support the El Gamal signature process. Key pairs are stored in the key file.

Local key file

A file containing the public and private keys used to sign and verify machine-specific Tripwire files. Tripwire database files are always signed with the local key, and Tripwire report files can be signed with the local key. Writing to a file protected with the local key requires the local passphrase. The local key file's location is specified by the LOCALKEYFILE variable in the configuration file.

MD5

MD5 is the RSA Data Security Inc. message-digest algorithm, a proposed data authentication standard. The Internet draft submission can be found as Internet working draft RFC 1321, available from http://www.merit.edu/internet/documents/. MD5 generates a 128-bit signature using four passes to ensure pseudo-random output. For three runs on files 100MB in size, its throughput mean was 15.499 MB/sec, on a 200MHz Motorola Power3 processor with 256 MB RAM running AIX 4.3.

Message-digest algorithm

Algorithms that may be used to render files tamper-evident. Small changes in the input file will cause large changes in the output file.

Object name

The part of a policy file rule that specifies a directory or file to be monitored by Tripwire.

Passphrases

Long passwords that control user access to important Tripwire files. Tripwire uses passphrases to encrypt the site and local *key files*, which are used to sign and verify Tripwire files. Secure passphrases should be longer than 8 characters in length and include both upper and lower case letters and numbers. Site and local passphrases should differ, so that an intruder who compromises the local key on one machine cannot use it to compromise other machines.

Policy file

A file, consisting of a series of rules, that controls the way that Tripwire checks the integrity of a system. Each rule in the policy file specifies a system object that Tripwire monitors, and describes which changes to the object should be reported, and which ones can safely be ignored. The site key signs the policy file, and its location is specified by the POLFILE variable in the configuration file.

Policy Update mode

A `tripwire` command mosw that updates the Tripwire policy file and synchronizes an existing Tripwire database file with the new policy file information.

Print Configuration File mode

A command mode of the `twadmin` command that prints the binary-encoded Tripwire configuration file in plain text form.

Print Database mode

A command mode of the `twprint` command that prints the binary-encoded Tripwire database file in plain text form.

Print Policy File mode

A command mode of the `twadmin` command that prints the binary-encoded Tripwire policy file in plain text form.

Print Report mode

A command mode of the `twprint` command that prints the Tripwire report file in plain text form.

Private key

A component of Tripwire's site and local key files. The private key is used to sign and verify files.

Property mask

The part of a policy file rule that specifies the file characteristics to monitor during integrity checks.

Public key

A component of Tripwire's site and local key files that verifies and reads files that are signed. In Tripwire operations, public keys allow verification only, while private keys enable signing as well as verification.

Recurse

The act of scanning one or more levels below a specified directory. Recursion starts with the specified directory.

Rule

A policy file statement that specifies which file properties will be monitored or suppressed during integrity checks (*see Property mask*). Only one rule may be specified for a given object file.

Rule name

The name given to a rule, which is used in on command lines as an argument, and in reporting.

SHA/SHS

SHS is the NIST Digital Signature Standard, called the Secure Hash Standard, and is described in NIST FIPS 180. Tripwire refers to it as the SHA, or Secure Hash Algorithm, because Tripwire uses a non-certified implementation and cannot claim standards confirmance. SHS generates a 160-bit signature. For three runs on files 100MB in size, its throughput mean was 12.811 MB/sec, on a 200MHz Motorola Power3 processor with 256 MB RAM running AIX 4.3.

siggen

A file utility that displays hash values for files. Siggen provided the performance output cited for the four signatures included in Tripwire: CRC-32, HAVAL, MD5, and SHA.

Signed file

A Tripwire file that has been signed with either the site or local key. Either the site or local passphrase is required to edit or write to a signed file. The file itself is not encrypted.

Site key file

A file containing the public and private keys used to sign and verify Tripwire configuration and policy files. Writing to a file protected with the site key requires the site *passphrase*. The site key file's default location is specified by the SITEKEYFILE variable in the configuration file.

Stop point

A component of the policy file that specifies directories or files that should not be scanned during an integrity check. A "!" marks stop points.

System object

A file, directory, or other discrete item in the filesystem that can be monitored by Tripwire.

Test mode

A mode of the `tripwire` command that checks the operation of Tripwire's email notification system, using the settings in the Tripwire *configuration file*.

`tripwire`

A command used for most basic Tripwire operations, including the creation and updating of the database file, and checking the integrity of the filesystem against that database.

`twadmin`

A command used to create signed versions of Tripwire files, and for various administrative functions.

`twprint`

A command that prints signed Tripwire database and report files.

Violation

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

Appendix

Appendix A: Exit Codes

Exit codes for `tripwire` Integrity Checking mode:

Table 25: Exit Codes for `tripwire` Integrity Checking Mode

Exit Code	Meaning
`0-7`	Success Any of the following values may be combined to produce a return result for integrity checking: 1 - files were added 2 - files were removed 4 - files were changed
`> 7`	Failure 8 - an error occurred that prevented the report file from being written. **Note:** A return value that is less than 8 does not always mean that errors did not occur or that any integrity checking actually did occur. The report file itself could contain error messages if, for instance, a file could not be accessed.

Table 26: Exit codes for all other command modes:

Exit Code	Meaning
`0`	Success
`>0`	Failure

Exit codes can be viewed by entering

```
echo %ERRORLEVEL%
```

on the command line immediately after running a Tripwire command.

Appendix B: Sample Tripwire Reports

This section contains samples of the various formats for Tripwire reports. The report files generated after an integrity check always contain full details, but you can view a subset of the report content by choosing a different report format.

You can specify the report levels described here with the REPORTLEVEL and EMAILREPORTLEVEL variables in the configuration file, or by using command-line options.

Level 0: Single Line Report

This is an unencrypted, single-line (80-character maximum) report. This is useful for syslog reports or as the subject line in an email report. Information in the single line consists of:

```
Note: Report is not encrypted.
TWReport web3 19991017183915 V:4 S:100 A:1 R:0 C:3
```

which lists the following information:

- Hostname
- Date and time the report was generated
- Total number of violations
- Maximum severity of violations
- Number of objects added, removed, and changed

Level 1: Parsable List of Violated Files

This type of report could be used to direct a backup program to automatically restore tampered files, or to automate some other response. The format for this report is:

- Type of violation (added, removed, changed)
- Name of object violated

```
Note: Report is not encrypted.
Added:  /usr/bin/bash
Modified:       /usr/bin
Modified:       /etc/passwd
Modified:       /var/sysadm/salog
```

Level 2: Summary Report

This format lists the violations that occurred by machine and rule, using minimal amounts of text. This type of report makes it very easy to see violations. Summary level reporting consists of:

- A brief header specifying the machine the report was run on
- List of rules violated
- Files in each rule that were added, removed, or modified

```
Note: Report is not encrypted.
Tripwire(R) 2.2.1 Integrity Check Report

Report generated by:          root
Report created on:            Sun Oct 17 18:39:15 1999
Database last updated on:     Never
Report Summary:
========================================================================

Host name:                    web3
Host IP address:              192.168.1.30
Host ID:                      c0aff41e
Policy file used:             /usr/tw221/policy/tw.pol
Configuration file used:      /usr/tw221/bin/tw.cfg
Database file used:           /usr/tw221/db/web3.db
Command line used:            tripwire --check -r report.twr
========================================================================

========================================================================
Rule Summary:
========================================================================
------------------------------------------------------------------------
  Section: Unix File System
```

```
-------------------------------------------------------------------

       Rule Name                  Severity    Adde    Remove   Modifi
                                    Level       d        d        ed

                      ----   ---------   ----   ------   ------
 ----------                     -----       -        -        --

 Invariant Directories          66          0        0        0

 Tripwire Data Files            100         0        0        0

 Temporary directories          33          0        0        0

 Critical devices               100         0        0        0

 Root config files              100         0        0        0

 Home directory per-            50          0        0        0
 missions

 Tripwire Binaries              100         0        0        0

 *OS executables and            100         1        0        1
 libraries

 *setuid/setgid                 100         0        0        1

 Shell Binaries                  0          0        0        0

 Libraries                      66          0        0        0

 Low security impact            10          0        0        0
 directories

 User binaries (/usr/           66          0        0        0
 local)

 *Configuration Files            0          0        0        1

 System boot changes            100         0        0        0

 Login Scripts                   0          0        0        0

 Kernel                         100         0        0        0

 (/unix)
Total objects scanned:  6037
Total violations found:  4
===============================================================
Object Summary:
===============================================================
```

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

---------------------------------------------------------------
Rule Name: OS executables and libraries (/usr/bin)
Severity Level: 100
---------------------------------------------------------------

Added:
"/usr/bin/bash"

Modified:
"/usr/bin"

---------------------------------------------------------------
Rule Name: Configuration Files (/etc/passwd)
Severity Level: 0
---------------------------------------------------------------

Modified:
"/etc/passwd"

---------------------------------------------------------------
Rule Name: setuid/setgid (/var/sysadm/salog)
Severity Level: 100
---------------------------------------------------------------

Modified:
"/var/sysadm/salog"

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

(C) Copyright 2000 Tripwire Security Systems, Inc. Tripwire is a
registered trademark of the Purdue Research Foundation and is
licensed exclusively to Tripwire Security Systems, Inc.
```

Level 3: Concise Report

This report provides all of the information in the Summary Report, and also lists expected and observed values for objects that have been modified. No additional details are provided for added or removed objects.

```
Note: Report is not encrypted.
Tripwire(R) 2.2.1 Integrity Check Report

Report generated by:          root
Report created on:            Sun Oct 17 18:39:15 1999
Database last updated on:     Never

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

Host name:                    web3
Host IP address:              192.168.1.30
Host ID:                      c0aff41e
Policy file used:             /usr/tw221/policy/tw.pol
Configuration file used:      /usr/tw221/bin/tw.cfg
Database file used:           /usr/tw221/db/web3.db
Command line used:            tripwire --check -r report.twr

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

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

Rule Name	Severity Level	Added	Removed	Modified
Invariant Directories	66	0	0	0
Tripwire Data Files	100	0	0	0
Temporary directories	33	0	0	0
Critical devices	100	0	0	0
Root config files	100	0	0	0

Home directory permissions	50	0	0	0
Tripwire Binaries	100	0	0	0
* OS executables and libraries	100	1	0	1
* setuid/setgid	100	0	0	1
Shell Binaries	0	0	0	0
Libraries	66	0	0	0
Low security impact directories	10	0	0	0
User binaries (/usr/local)	66	0	0	0
* Configuration Files	0	0	0	1
System boot changes	100	0	0	0
Login Scripts	0	0	0	0
Kernel	100	0	0	0
(/unix)				

```
Total objects scanned:  6037
Total violations found:  4

=====================================================
Object Detail:
=====================================================
-----------------------------------------------------------------
  Section: Unix File System
-----------------------------------------------------------------

-----------------------------------------------------------------
Rule Name: OS executables and libraries (/usr/bin)
Severity Level: 100
-----------------------------------------------------------------
  ---------------------------------------
  Added Objects: 1
  ---------------------------------------

Added object name:  /usr/bin/bash

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

Modified object name:  /usr/bin

  Property:        Expected                 Observed
  -------------    -----------              -----------
* Modify Time    Tue Oct 05 16:00:45 1999   Sun Oct 17 18:38:42 1999


-----------------------------------------------------------------
Rule Name: Configuration Files (/etc/passwd)
Severity Level: 0
-----------------------------------------------------------------
  ---------------------------------------
  Modified Objects: 1
  ---------------------------------------

Modified object name:  /etc/passwd

  Property:          Expected                Observed
  -------------      -----------             -----------
* Inode Number       1079649                 1079654
* Mode              -rwxr-xr-x              -r--r--r--


-----------------------------------------------------------------
Rule Name: setuid/setgid (/var/sysadm/salog)
Severity Level: 100
```

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

Modified object name:  /var/sysadm/salog

  Property:     Expected                  Observed
  ------------  -----------               -----------
* Size          4299                      5568
* Modify Time   Wed Oct 13 13:41:36 1999  Sun Oct 17 18:27:53 1999
* Change Time   Wed Oct 13 13:41:36 1999  Sun Oct 17 18:27:53 1999
* CRC32         Dx7Vne                    DW1kxa
* MD5           D3MQPm68hJGIxtNkCV2nwY    CUtxE/yRJEcd+LOy+jDUHn



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

(C) Copyright 2000 Tripwire Security Systems, Inc. Tripwire is a
registered trademark of the Purdue Research Foundation and is
licensed exclusively to Tripwire Security Systems, Inc.
```

Level 4: Full Report

This report format provides the maximum level of detail, including all observed property values for added and removed objects. For modified objects, all observed property values that have changed are listed.

```
Note: Report is not encrypted.
Tripwire(R) 2.2.1 Integrity Check Report

Report generated by:          root
Report created on:            Sun Oct 17 18:39:15 1999
Database last updated on:     Never

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

Host name:                    web3
Host IP address:              192.168.1.30
Host ID:                      c0aff41e
Policy file used:             /usr/tw221/policy/tw.pol
Configuration file used:      /usr/tw221/bin/tw.cfg
Database file used:           /usr/tw221/db/web3.db
Command line used:            tripwire --check -r report.twr

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

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

 Rule Name               Severity    Adde   Remove   Modi-
                         Level       d      d        fied

 ---------               ---------   ----   ------   ------
                         -----       -      -        --

 Invariant Directories   66          0      0        0

 Tripwire Data Files     100         0      0        0

 Temporary directories   33          0      0        0

 Critical devices        100         0      0        0
```

Root config files	100	0	0	0
Home directory permissions	50	0	0	0
Tripwire Binaries	100	0	0	0
* OS executables and libraries	100	1	0	1
* setuid/setgid	100	0	0	1
Shell Binaries	0	0	0	0
Libraries	66	0	0	0
Low security impact directories	0	0	0	10
User binaries (/usr/local)	66	0	0	0
* Configuration Files	0	0	0	1
System boot changes	100	0	0	0
Login Scripts	0	0	0	0
Kernel (/unix)	100	0	0	0

```
Total objects scanned:  6037
Total violations found:  4

========================================================================
Object Summary:
========================================================================
------------------------------------------------------------------------
# Section: Unix File System
------------------------------------------------------------------------
------------------------------------------------------------------------
Rule Name: OS executables and libraries (/usr/bin)
Severity Level: 100
------------------------------------------------------------------------
Added:
“/usr/bin/bash”

Modified:
“/usr/bin”

------------------------------------------------------------------------
Rule Name: Configuration Files (/etc/passwd)
Severity Level: 0
------------------------------------------------------------------------
Modified:
“/etc/passwd”

------------------------------------------------------------------------
Rule Name: setuid/setgid (/var/sysadm/salog)
Severity Level: 100
------------------------------------------------------------------------
Modified:
“/var/sysadm/salog”

========================================================================
Object Detail:
========================================================================
------------------------------------------------------------------------
Section: Unix File System
------------------------------------------------------------------------
------------------------------------------------------------------------
Rule Name: OS executables and libraries (/usr/bin)
Severity Level: 100
------------------------------------------------------------------------
---------------------------------------
  Added Objects: 1
---------------------------------------

Added object name:  /usr/bin/bash
```

```
  Property:          Expected                Observed
  -------------      -----------             -----------
* Object Type          ---                   Symbolic Link
* Device Number        ---                    87
* Inode Number         ---                    6292003
* Mode                 ---                    lrwxr-xr-x
* Num Links            ---                    1
* UID                  ---                    root (0)
* GID                  ---                    sys (0)
* Size                 ---                    4
* Modify Time          ---                  Sun Oct 17 18:38:42 1999
* Blocks               ---                    0
* CRC32                ---                    CqaHZi
* MD5                  ---                  AuIilKQK6LrR4kyXmG7OJ8

  ----------------------------------------
  Modified Objects: 1
  ----------------------------------------
Modified object name:  /usr/bin

  Property:          Expected                Observed
  -------------      -----------             -----------
  Object Type        Directory               Directory
  Device Number       87                      87
  Inode Number       6291626                 6291626
  Mode              drwxr-xr-x              drwxr-xr-x
  Num Links           3                       3
  UID               root (0)                root (0)
  GID               sys (0)                 sys (0)
  Size              12288                   12288
* Modify Time     Tue Oct 05 16:00:45 1999  Sun Oct 17 18:38:42 1999
  Blocks             24                      24

------------------------------------------------------------------------
Rule Name: Configuration Files (/etc/passwd)
Severity Level: 0
------------------------------------------------------------------------
  Modified Objects: 1
----------------------------------------

Modified object name:  /etc/passwd

  Property:          Expected                Observed
  -------------      -----------             -----------
  Object Type        Regular File            Regular File
  Device Number       87                      87
* Inode Number       1079649                 1079654
* Mode              -rwxr-xr-x              -r--r--r--
  Num Links           1                       1
  UID               root (0)                root (0)
```

```
  GID               sys (0)                 sys (0)
--------------------------------------------------------------------
Rule Name: setuid/setgid (/var/sysadm/salog)
Severity Level: 100
--------------------------------------------------------------------
   ---------------------------------------
   Modified Objects: 1
   ---------------------------------------

Modified object name:  /var/sysadm/salog

  Property:          Expected                Observed
  -------------      -----------             -----------
  Object Type        Regular File            Regular File
  Device Number       87                      87
  File Device Number   0                       0
  Inode Number        7434092                 7434092
  Mode               -rw-r-lr--              -rw-r-lr--
  Num Links           1                       1
  UID                root (0)                root (0)
  GID                sys (0)                 sys (0)
* Size                4299                    5568
* Modify Time     Wed Oct 13 13:41:36 1999  Sun Oct 17 18:27:53 1999
* Change Time     Wed Oct 13 13:41:36 1999  Sun Oct 17 18:27:53 1999
  Blocks              16                      16
* CRC32             Dx7Vne                   DW1kxa
* MD5           D3MQPm68hJGIxtNkCV2nwY     CUtxE/yRJEcd+LOy+jDUHn

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

(C) Copyright 2000 Tripwire Security Systems, Inc. Tripwire is a
registered trademark of the Purdue Research Foundation and is
licensed exclusively to Tripwire Security Systems, Inc.
```

Index

Symbols

- 67
!, 83
$(DATE)
    Report file variable, 55
$(Device) variable, 96
$(Dynamic) variable, 96
$(Growing) variable, 96
$(IgnoreAll) variable, 96
$(IgnoreNone) variable, 96
$(ReadOnly) variable, 96
-> token, 77
||, 92

A

Access time
    and directory rules, 82
    Overlook changes to, 40
    Property, 82
Address multiple emails, 84
Argument
    Help with command mode, 48
    In rule attribute, 84
    One per rule attribute, 84
Attribute
    Case-insensitive, 84
    Emailto, 88
    Modifying rule with, 84
    One argument limit, 84
    Recurse, 90
    Rulename, 87
    Scoped and single in same rule, 86
    Specify with object name and property mask, 85
    Stop point within, 85

B

Ballot box
    Accessible in interactive mode, 53
    Editing, 54
Best practice
    Delete plain text copies, 37
    Experienced users organize policy files, 44
    LATEPROMPTING setting, 39
    LOOSEDIRECTORYCHECKING setting, 40
    MAILNOVIOLATIONS setting, 40
    Run binaries from read-only media, 31
    See also Security; How to
Blocks, specify number allocated in property mask, 82

C

Change
    Approve file system change for database, 57
    Approving change to database, 55
    Policy file, 50
    from Tripwire ASR, 23
Character
    Double-byte require quotation marks, 78
    in escape sequences, 79
Command
    Display mode help, 48
    Execute from absolute pathname, 49
    Wildcards discouraged, 48
Comment syntax, 38

Configuration file
Change after installation, command for, 64
Location requirement, 37
Minimum requirements for, 38
Name requirement, 37
Sample, 38
Signed by site key, 28
Signing cryptographically, 37
Text example, 38
Using custom at install, 30
Using default, 29
Cryptography, 19

D

Database
Build on policy file rules, mode, 49
--dbfile option, 62
DBFILE variable, 45
Integrity check mode, 49
Keeping backups, 52
Save time by updating, 50
Update specific entry, example, 55
Update with report file, 56
Update without regenerating, 50
DATE variable, 41
Debugging, directive for, 93
Device
Number property, 82
Predefined variable, 96
Directives
Debugging, 93
GLOBAL, 91
Nesting, 93
Purpose, 90
Variables defined in, 95
Directory
Control monitoring with recurse, 90
Recursion into, 79
Scanning with indicated property mask, 77
Tree, 34
Display signed files in plain text, 61

E

EDITOR variable, 40
El Gamal, 19
Email
Default severity setting sends, 89
EMAILREPORTLEVEL variable, 40
Header requirement, 39
Requires policy, configuration, and command line, 89
Specify multiple addresses for single rule, 88
Test mode checks notification, 60
Troubleshooting variables, 40
Using severity level with, 41
End-of-file, 91
Error condition
During database update, 56
Undefined variable, 39
Escape sequences, 79

F

File integrity assessment, 16
Firewall, deploy Tripwire with, 17
Forensics
Prevent intruder from destroying evidence, 52
Using print database mode for, 62

G

GLOBAL directive, 91
Group id of owner, 81
Growing files, monitoring, 82

H

Hash
- Incompatible with +a property, 82
- Performance-security tradeoff, 43
- Recommended use, 83
- Use recommendation, 43, 83

Hex numerals in object names, 79
HOSTNAME variable, 41
How to
- Avoid spurious violations of access timestamp property, 82
- Change the policy file, 59
- Check every object, 55
- Conveniently organize multiple integrity checks, 51
- Define your own property mask, 81
- Ensure that valid Tripwire installation executes, 49
- Get online help with modes, 48
- Ignore specific files in a directory, 83
- Ignore specific properties during integrity check, 80
- Monitor only addition or deletion of files, 81
- Organize integrity check results by host and rule, 111
- Plan policy file changes, 44
- Print an integrity check report, 61
- Print readable configuration file, 65
- Set up multiple masks in a directory tree, 77
- Share one site key across multiple systems, 39
- Sign a report file, 54
- Specify multiple email addresses in a single rule, 88
- Specify common policy file for both UNIX and NT, 76
- Suppress low severity rules during an integrity check, 90
- Tell that email notification is working, 60
- Toggle property on or off, 80
- Track certain objects within a large database, 87
- Use directives for debugging, 93
- Use one policy file for multiple hosts, 76, 92
- Use pre-2.2.1 Tripwire files at install, 29
- Use severity level to prioritize rules, 89

I

Ignore
- Specified files or directories, 83
- Ignore specified properties in rule, 81

IGNORENONE variable, 96
Inode
- Creation/modification timestamp, 82
- Device number of disk, in rule, 82
- Record reference counts in rule, 81
- Specify number in property mask, 81

Installing
Custom configuration file 30
Directory tree created, 34
Unattended install, command, 30
Use pre-existing configuration file, 29
Use pre-existing policy file, 29
with custom configuration file, 30
with default configuration file, 29
Integrity check
Check specific list of objects, 55
Organize files prior to, 44
Prevent overwrite of backup database, 52

K

Key
Keypair and key file, 19
Local key file, 19
Local key passphrase assignment 28
Overwriting risk, 71
Private used to write signed files, 19
Public used to read files, 19
Regenerate, 71
Site key file, 19
Site key passphrase, 28
Unusable keyfile, 71
Keyword
Case-sensitive in configuration file, 38

L

LATEPROMPTING variable, 39
Local key
for database file, 28
for report file, 28
Local key (*continued*)
LOCALKEYFILE variable, 38
Optional for report file 28
Passphrase needed for, 28
Logical OR operand symbol, 92
LOOSEDIRECTORYCHECKING variable, 39

M

Mail
Email attribute is additive, 86
Emailto attribute and configuration file, 89
Emailto attribute in rule, 54
MAILMETHOD variable, 39
MAILNOVIOLATIONS variable, 40
MAILPROGRAM variable, 39
MAPI, not supported, 39
Nesting attributes, 88
Nesting email in policy file, 88
Requires --email-report argument, 89
SMTP value, 39
SMTPHOST variable, 39
Specify host domain name, 39
Specify multiple addresses for one rule, 88
Specify recipients, 39
Verify that Tripwire executed as scheduled, 40
Mode
Database initialization, 49
Database update, 50
Integrity check, 55
Integrity checking, 49
Online help for, 48
Policy update, 50

Mode (*continued*)
Syntax of, v
Mount points not crossed, 79

N

Normal rule
Object name in, 77
Property mask in, 77
Purpose, 76

O

Object
Device number of, 82
Name, valid characters in, 78
Only one rule for each, 78
Object name
Japanese characters in, 78
No environmental variables in, 78
Purpose, 77
Specify with property mask and attributes, 85
Suppress scan of, 83
Wildcards in, 78
Octal numerals in object name, 79
Operating system requirements, 26

P

Passphrase
Assign two, 29
Forgotten, 29
Security requirement, 29
Site, purpose, 28
Valid characters, 28
Path
Absolute pathname for security, 49
Specifying for configuration variable, 41
Policy file
@@else, 92
@@endif, 92
@@ifhost in, 92
After installation 41
Create with twadmin, example, 44
Customizing, 20
Only Logical OR supported, 92
Signed by site key, 28
Specifying multiple hosts, 92
Synchronize database with new, 50
Update failes in high security, 59
Use directives to administer multiple machines, 90
Wildcards in, 78
POSIX, property designation, 82
Print
Database with twprint, 62
Report, can't find, 61
--twrfile option, 55, 61
twprint, 55, 61
Private key
Enables encryption and decryption, 103
Property
Definition, 76
Ignore during integrity check, 54
Ignored if not specified, 80
Ignoring changes to, 40
Mask, defining, 77
See also Property Mask
Turn off one in a mask of several, example, 81
Watch all for specified file, example, 77
Property Mask
File type, in rule, 81
Group id of owner, in rule, 81

Property Mask (*continued*)
No properties in, 81
Permission and mode bit specification, 81
Purpose, 80
Requirements, 80
User id of owner, in rule, 81
Public key decrypts, 19, 103

Q

Quotes
for Kana and Kanji characters, 78
in object names, 78
Invalid for passphrase, 28

R

Read-only media, 19
Record and check properties in rule, 81
Recurse
Into subdirectory, 79
Stop points and, 83
Report
Can't find specified report file, 56
Create, 61
Default location after integrity check, 51
Display during Tripwire operation, 61
Email, 88-89
Local key for, 28
Location, 51
Printing contents, 61
Rulenames in, 41
Single line, example, 109
Sort by rulename, 87
Report
check, 51
Report (*continued*)
Update database entries automatically, 56
Variables in policy file, 41
REPORTLEVEL variable, 40
Rule
+a property incompatible with hash, 82
and quoted object name, 78
Assign one attribute to multiple, 84
Attributes case-insensitive, 84
Attributes syntax, 84
Directives in scoped attribute block, 94
Environmental variables in, 78
Multiple email addresses for, 88
Multiple objects grouped by rulename, 87
Name with rulename, 87
Normal, purpose of, 76
One per file, 41
Rulename attribute, 87
Rulename and severity, 54
Run only at given severity, 54
Run only one, example, 52
Run only the specified rule, argument, 54
Scoped rule block, 85
Sort by rulename, 87
Specify multiple attributes, 85
Stop point, limiting scan in rule, 83
Using rulename, 41
Variables in, 95

S

Section
@@section directive, 91
Explicit labelling of, 76

Section (*continued*)
Naming for UNIX and NT systems, 76
Separate object names, 55
Security
Commands should use absolute pathname, 49
Hazard using wildcard, 48
Hazard when using growing file property, 82
Lowering for a policy update, 59
of passphrases, 29
Organizing files to be integrity-checked, 44
Performance trade-off, 43
Specify warnings without changing database, 56
Unusable files if key file is overwritten, 71
Using severity level for, 41
Severity
Default for rule, 89
Default sends email, 89
Purpose, 41
SHA/SHS
Defining use in policy file 42
Example as policy file variable 42
Implementation variance, 104
Signature
Display values, 72
Incompatible with access timestamp, 82
Suppress on policy file, 67
Values different from ASR files, 72
Site key
and configuration file, 28
and policy file, 28
Site key (*continued*)
Command-line argument, 56
Passphrase needed, 28
SMTP variable, 39
SMTPHOST, 39
Sort report data with rulename, 87
Stop point
Purpose, 83
Specifying, 83
Takes precedence over recurse, 90
Syslog
in configuration file, 38
Settings for, 40
System lockdown, 22
System object definition, 41

T

Time
Access time, overlook change to, 40
Modification time, overlook change to, 40
Tokens in variable substitution, 95
tripwire command, 49
Tripwire files overview, 18
Troubleshooting
Email, check variables, 40
Symbols in policy file, 81
See also Security; Best practice

U

Unusable file, possible cause of, 71
updating
database file 20
policy file 20
User id of owner, specify in rule, 81

V

Variable
- Configuration file, 38
- DBFILE, 38
- Define before using, 41
- MAILMETHOD specification, 39
- MAILMETHOD, 39
- POLFILE, 38
- Predefined, 41
- Regular expression of, 94
- REPORTFILE, 38
- Right-hand substitution, 41
- Scope of, 94
- SITEKEYFILE, 38
- Substitution of, 94
- Time-based, 55
- Undefined, 39
- Valid characters in name, 41

Violation
- During Policy Update, 58
- Not all are intrusions, 59
- Report by machine and rule, 111

W

Warnings
- Report differences found without changing database, 56

Wildcard
- Expansion rules, 48
- in passphrases, 28
- in policy file, 78
- Security risk using, 48

Writing files, prevention of, 19