Tripwire, Inc.
326 SW Broadway
3rd Floor
Portland, OR 97204
www.tripwire.org
opensource@tripwire.org

About This Guide

Purpose and Scope

This User's Guide is for the administrator responsible for installing and running Tripwire Open Source software. It is specifically designed for users who are familiar with the conventions of their operating system software, but are inexperienced with Tripwire software.

Conventions

The following typographic conventions are used in this manual:

Bold	is used for ftp and http URLs, and to emphasize security issues in regular text.
Italic	is used for file names, and to denote the first use of terms that are specific to Tripwire software. Definitions for these terms are in the Glossary.
`Fixed Width`	is used in examples to show actual user input on the command line, and in regular text to show commands and command-line options, rule attributes, directives, and variables.
`Fixed Italic`	is used in examples to show variables which should be replaced with a context-specific value.
[Brackets]	encase optional arguments.
{Braces}	encase multiple options, of which one must be chosen. The "|" token delimits each option: {1 | 2 }.

Command-line examples in the manual assume that you have the */usr/sbin* directory in your PATH, and don't need to specify the Tripwire executables explicitly.

Related Documents

The Tripwire Quick Reference Card provides easy access to information on Tripwire commands and policy file syntax.

Latest Information

For the latest information and support for open source Tripwire products, check:

http://www.tripwire.org

Contents

Tripwire Command Reference 33

Policy Reference 61

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 Chapter 2, Using Tripwire Software. This section describes:

- what Tripwire software is
- how to deploy Tripwire software
- Tripwire components and security features
- how Tripwire software works
- applications of Tripwire software

1.1 What Tripwire Software Is

Tripwire software 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 software 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.

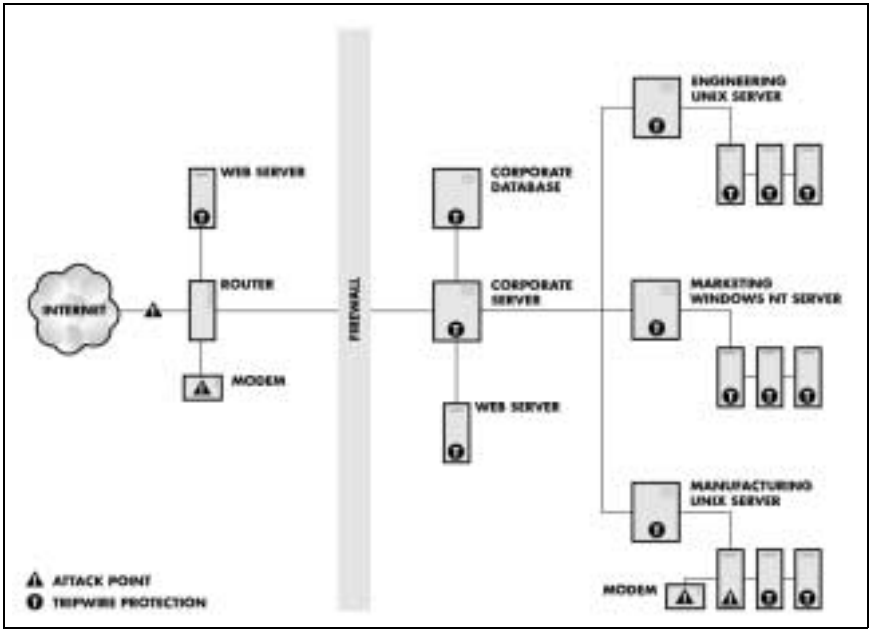


Figure 1: Deploying Tripwire Software in a Network

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 software 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. Appendix B 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 a public/private key pair. 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, Inc. 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 software on a system, you will need to configure the software for normal operation. These steps are described in greater detail in section 2.1, Configuration.

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 section 2.2, Regular Operation.

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, 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.

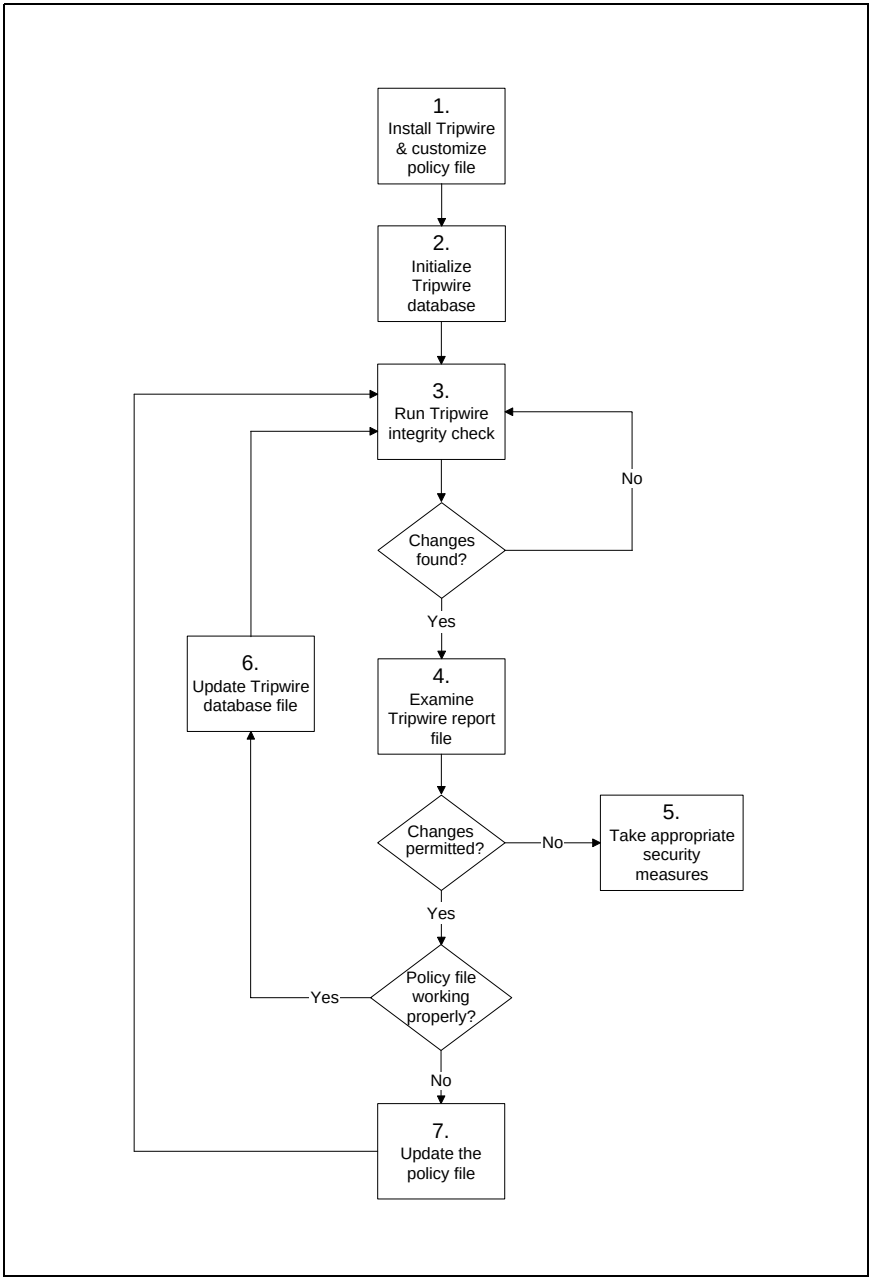


Figure 2: Tripwire Operation Process

1.5 Other Tripwire Applications

Tripwire software 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.

2

Using Tripwire Software

2.0 Overview

This chapter describes the process for configuring and using Tripwire software after you have installed it. For information on installing Tripwire software, see the INSTALL file in the Tripwire root directory.

The Configuration section describes the tasks you need to perform to prepare Tripwire software for normal operation, including:

- editing the configuration file
- testing e-mail reporting
- customizing the default policy file
- initializing the database file

The Regular Operation section describes the tasks you need to perform for regular operation of Tripwire software, including:

- running an integrity check
- checking report files
- updating the database file
- updating the policy file
- changing passphrases

2.1 Configuration

The Configuration section describes tasks you need to perform to prepare Tripwire software for normal operation. Normally, you will only need to perform these procedures once, before you initialize the database file.

After you initialize the database, the final step in the configuration process, use the procedures in the Regular Operation section for daily operation of Tripwire software.

2.1.1 Editing the Configuration File

You edit the configuration file to controls many aspects of Tripwire software operation. For security reasons, the configuration file is stored on the system in an encoded and signed form.

The encoded configuration file is named *tw.cfg*, and is located in the */etc/tripwire* directory, by default. A plain text copy of the same file, named *twcfg.txt*, is in the same directory. To make changes to the configuration file, edit the plain text version and then encode and sign that file.

Security Issue: After editing, you should remove any plain text copies of the configuration file to prevent unauthorized access.

You may need to edit the configuration file settings to change:

- the location of Tripwire files
- the settings used to send e-mail reports
- directory parsing during an integrity check
- the default level of detail for Tripwire report files

See section 5.1 for a complete description of the components of the Tripwire configuration file.

Security Issue: Any plain text copies of the configuration file should be deleted or stored in a secure location to prevent unauthorized access.

To edit the Tripwire configuration file:

1. If no plain text version of the configuration file exists, print one with the `twadmin` Print Configuration File mode:

```
twadmin --print-cfgfile > myconfig.txt
```

2. Edit and save the text file. See section 5.2 for a description of configuration file options.
3. Use the `twadmin` Create Configuration File mode to sign and install the text file as the new configuration file:

```
twadmin --create-cfgfile --site-keyfile /etc/tripwire/site.key
  myconfig.txt
```

See section 3.3 for more information on the `twadmin` command.

2.1.2 Testing E-mail Reporting

Email reporting enables you to send the results of an integrity check via email. The mechanism used to send email notification is specified during installation, but can be changed at any time by modifying the MAILPROGRAM, MAILMETHOD, and other settings in the configuration file. See section 5.2 for more information on these settings.

Email notification settings can be tested at any time with the `tripwire` Test mode:

```
tripwire --test --email user@domain.com
```

To use email notification successfully:

- You must have valid settings for the email reporting variables in the configuration file.
- The `emailto` rule attribute must be specified for one or more rules in the policy file.
- The `-M` or `--email-report` option of the `tripwire` command must be specified on the command line when the integrity check is run.

See section 3.1.5 for more information on the `tripwire` Test mode.

2.1.3 Customizing the Default Policy File

Tripwire software ships with a default policy file that monitors a wide range of directories and files. You can edit this policy file to suit your own system, or use it as a model to create your own policy file.

After you run your first integrity checks, you will probably need to update the rules in the policy file (section 2.2.4) several more times, as you add new rules and change existing rules.

When you are first setting up the policy file, you can edit a new policy file text and identify it as described in this section. After you have generated the database, you should **always** use the `tripwire` policy update mode (section 3.1.4) to change the policy file.

To customize the default policy file:

1. Open the file *twpol.txt*, located in the */etc/tripwire* directory, with the text editor of your choice.

   Read through the rules in the policy file. Lines that begin with the '#' character are comments, and are not parsed by Tripwire software.
2. Using the comments in the policy file, add or remove the '#' character from rules, based on your system configuration and the files that you want to monitor.
3. Save the policy file.
4. Use the `twadmin` create policy file mode to encode and sign the text file and install it as the new policy file:

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

5. Delete the plain text copy of the policy file or store it in a secure location to prevent unauthorized access.

2.1.4 Initializing the Database

After you edit the configuration and policy files, the last step in the configuration process is to create the baseline Tripwire database. **You only need to initialize the database once, when you first set up Tripwire software.**

During this process, the `tripwire` executable reads the rules in your policy file, collects information from your system based on these rules, and stores this information in a database file. The database file is binary-encoded and signed to prevent unauthorized modification.

Because the database file serves as the baseline for all later integrity checks, make sure that you generate the database on a machine that has not been compromised. For maximum security, you should create your baseline database immediately after installing your operating system and application files from original media.

To initialize the Tripwire database file:

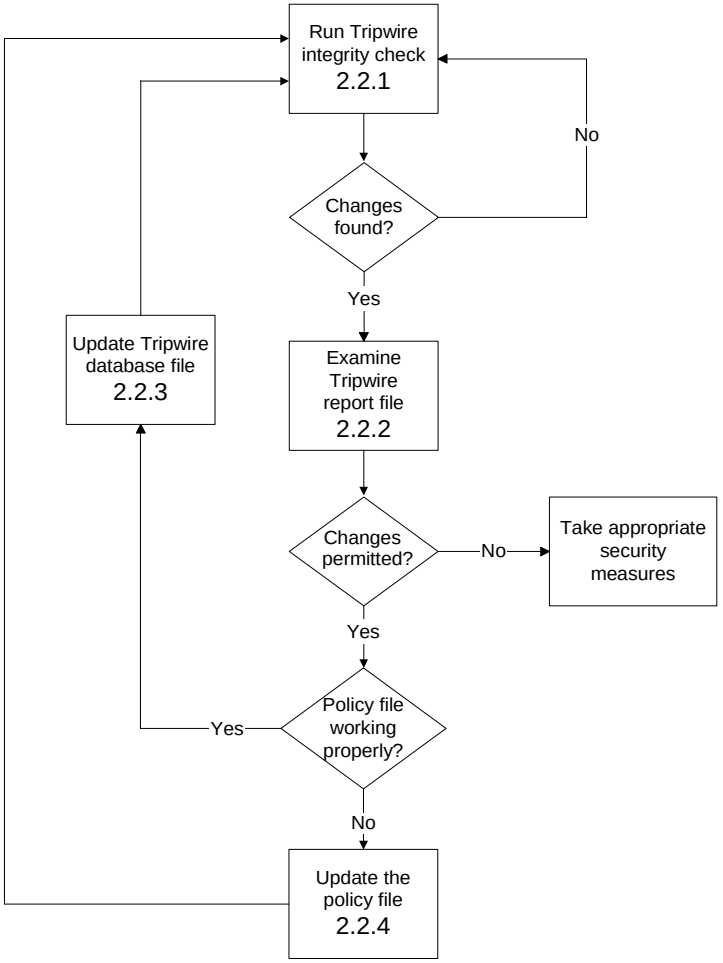
```
tripwire --init
```

This command saves the database file to the location specified by the DBFILE setting in the configuration file. By default, this is */var/lib/$(HOSTNAME).twd*.

After initializing the database file, you can run integrity checks with Tripwire software at any time. See section 2.2 for more information about regular Tripwire operation.

2.2 Regular Operation

The figure below shows Tripwire operation after the software has been installed and configured, and the database file has been initialized. Each of the steps in the process is described in greater detail in the following sections.



2.2.1 Checking Integrity

After you have initialized the database file, you can check integrity at any time with the `tripwire` Integrity Check mode. In this mode, the `tripwire` executable compares the state of the current file system against the values stored in the database file to find any violations.

To run a basic integrity check:

```
tripwire --check
```

After an integrity check, a text report of discovered violations will be printed to *stdout* and a binary copy of the report will be saved in the location specified by the REPORTFILE setting in the configuration file.

To run an integrity check, and specify the destination report file:

```
tripwire --check --twrfile /var/lib/report/myreport.twr
```

To send email with the results of an integrity check:

```
tripwire --check --email-report
```

With the command specified above, an email report will be sent to all recipients specified in the policy file, using the report format specified by the EMAILREPORTLEVEL variable in the configuration file.

To specify the report level to be sent for email reports:

```
tripwire --check --email-report --email-report-level 2
```

Samples of each level of report appear in Appendix B.

To specify the severity level of rules for an integrity check:

If you've used the `severity` rule attribute to categorize the rules in your policy file by importance, you can run an integrity check using only the rules at the `severity` level or higher:

```
tripwire --check --severity 80
```

To specify rules by name for an integrity check:

If you use the `rulename` rule attribute to name the rules in your policy file, you can run only a specific rule with the following command:

```
tripwire --check --rule-name rulename
```

If you want to run several rules, you can 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.

To specify objects to be checked:

If you only want to check specific directories or files, use the command:

```
tripwire --check object1 object2 object3 ...
```

where *object* is the fully-qualified path.

To ignore properties during an integrity check:

When running an integrity check, collecting data for some properties (particularly hashes) can be time-consuming. To run an integrity check that ignores specific properties for all objects, you can list the properties to ignore using their policy file masks:

```
tripwire --check --ignore "property,property,property"
```

where *property* is a Tripwire property (see section 4.2.2).

See section 3.1.2 for a complete listing of command-line options for the `tripwire` Integrity Check mode.

2.2.2 Examining a Report File

Whenever you run an integrity check, the `tripwire` executable will print a copy of the report file to *stdout*, and may optionally email reports to the recipient(s) specified in the policy file. You can use the information in these reports to decide if the violations found are malicious. If you wish to examine a report file in greater detail, you can use the `twprint` Print Report mode to print any existing report file.

To print an existing report file to the screen:

```
twprint --print-report --twrfile /var/lib/report/report.twr
```

To print an existing report file as a text file:

```
twprint --print-report --twrfile /var/lib/report/report.twr >
  myreport.txt
```

Both of the previous commands will produce a report at the level of detail specified by the REPORTLEVEL variable in the configuration file.

To specify the report level when printing a report file:

```
twprint --print-report --report-level 4
  --twrfile /var/lib/report/report.twr
```

Samples of each level of report appear in Appendix B.

2.2.3 Updating the Database

After an integrity check, if the Tripwire software finds any changes, you should update the database to reflect the current state of the system, and to prevent authorized changes from being interpreted as violations in future integrity checks. Database Update mode displays a report, and enables you to choose which, if any, of the database records should be updated:

```
tripwire --update --twrfile /var/lib/report/report.twr
```

If you run an integrity check using the `--interactive` option, you can update the database immediately after the integrity check is complete.

To launch a database update immediately after an integrity check:

```
tripwire --check --interactive
```

Either of these commands will open an editor session, using the program specified by the EDITOR setting in the configuration file. Each violation discovered by the integrity check will be displayed with a corresponding "ballot box":

```
Modified:
[x] "/usr/local/tw"
      drwxr -xr -x root(0)              512 Tue Nov 22 17:19:15 1999
```

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, the `tripwire` executable will update and save the database.

See section 3.1.3 for a complete listing of command-line options for the `tripwire` Database Update mode.

2.2.4 Updating the Policy File

You may wish to change the rules in the policy file if:

- you add new files to your system that you wish to monitor
- policy file rules are generating a lot of false positives or report file "noise"
- you want to change the recipients of email reports, or the way that rules in the policy file are grouped

After the Tripwire database file has been initialized, it is important that all changes to the policy file are made with the `tripwire` Policy Update mode, rather than with the `twadmin` Create Policy File mode. Whenever a new policy file is created, the database must be re-initialized. If an intruder has modified files since the last integrity check, these changes will not be detected, and will be included as part of the new baseline database. See section 3.1.4 for more information on `tripwire` Policy Update Mode.

To update the policy file:

1. Print out a copy with the `twadmin` Print Policy File mode:

```
twadmin --print-polfile > mypol.txt
```

2. Edit and save the text policy file.
3. Use the `tripwire` Policy Update mode to apply the changes to the existing policy file:

```
tripwire --update-policy mypol.txt
```

This process will add, remove, or update any database entries, sign the new policy file with the site key and install it in the directory specified by the POLFILE variable in the configuration file. Policy changes can be confirmed by running `tripwire` in Integrity Checking mode.

2.2.5 Changing Passphrases

Tripwire site and local key files are generated during installation, but you can change the key used to sign files at any time. Since the passphrases used to sign files are inextricably linked with key files, changing the keys is the only way to change your passphrases if you have staff changes, or if you believe that your passphrases have been compromised.

Important: If you delete or overwrite the key file used to encrypt a file, that file will become permanently unusable. Always make backup copies of key files before changing encryption options.

To change encryption options for an individual file or files:

1. If you don't know what key file is used to sign a file, use the `twadmin` Examine Encryption command to find out:

```
twadmin --examine file1 file2
```

2. Use the `twadmin` Remove Encryption mode to remove the signature. You will need to enter the appropriate passphrase.

```
twadmin --remove-encryption file1 file2
```

3. If you want to generate a new key file, use the `twadmin` Generate Keys mode:

```
twadmin --generate-keys --local-keyfile /etc/tripwire/sitekey.key
OR
twadmin --generate-keys --site-keyfile /etc/tripwire/localkey.key
```

4. Use the `twadmin` Encrypt mode to sign the files:

```
twadmin --encrypt --local-keyfile /etc/tripwire/sitekey.key file1 file2
OR
twadmin --encrypt --site-keyfile /etc/tripwire/sitekey.key file1 file2
```

3

Tripwire Command Reference

3.0 Tripwire Commands

This section describes the commands and command modes that Tripwire software uses, and lists the command-line options for each of those commands. The following four executables are used for all operations:

`tripwire` creates the baseline database, checks file system integrity, and updates the database or policy files.

`twadmin` creates configuration and policy files and performs cryptographic operations with Tripwire files.

`twprint` verifies and prints database and report files.

`siggen` generates and prints hashes for specified files.

All commands take their default values from the Tripwire configuration file, unless otherwise specified.

3.0.1 Command Conventions

All Tripwire applications (except `siggen`, which only has one mode) observe the following convention for command-line syntax:

```
command mode-selector [options] [files]
```

That is, the mode selector must always be the first argument on the command line, and any files not associated with the command-line options must appear last on the command line. Specifying command-line arguments in any other order will generate a syntax error.

Valid command-line arguments for each mode are shown in the sections below. The following conventions are used:

[-f *filename*]	Optional arguments are in square brackets. All other arguments are required. Variables that should be replaced with a relevant value are in italics.
-Z {high | low}	Arguments that must be chosen from a set are in brackets.

3.0.2 Command-Line Help

All Tripwire applications support the following arguments for obtaining usage, version, and copyright information. If this argument is present on the command line, the help message will be displayed and all other command-line arguments will be ignored.

Table 1: 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.

You can use wildcards to specify directories or files for Tripwire commands that accept multiple objects as command-line arguments. Wildcards are expanded by the shell, not based on the data stored in the database file.

Security Issue: Using wildcards on the command line creates a small but significant security exploit. By inserting a file that mimics a command-line option, an intruder could adversely affect the operation of Tripwire software.

3.1 tripwire

You can run the `tripwire` command in one of five modes: Database Initialization, Integrity Checking, Database Update, Policy Update, or Test mode.

In **Database Initialization** mode, the `tripwire` executable builds a database of file system objects, based on the rules in the policy file. This database will serve as the baseline for later integrity checks. This step only needs to be performed when Tripwire software is first installed.

The **Integrity Checking** mode compares the actual file system objects residing on the system with information stored in the baseline database. A report of any violations discovered will be printed to *stdout*, and a binary copy of the report will be stored on disk.

After you run an integrity check, **Database Update** mode enables you to update the database with changes from an integrity check report. This process enables you to update the database with new information without having to regenerate the entire database.

Policy Update mode enables you to synchronize the database and policy files after changes are made to the policy file. This mode enables you to change the rules in the policy file, and thereby change the way that Tripwire software scans the system, without having to do a complete re-initialization of the database.

Test mode tests the Tripwire email notification system, using the settings currently specified in the configuration file.

3.1.1 tripwire Database Initialization Mode

In Database Initialization mode, the Tripwire software generates a database based on the policy file rules and then signs it. Because this database will become the baseline for later integrity checks, it is essential that you create the database from a system that has not been compromised.

The default location for the generated database is specified by the DBFILE variable in the configuration file, unless you specify another location on the command line. Additional command-line options can be entered to specify the policy, configuration, and key files used to create the database. If no options are specified, the default values from the current configuration file are used.

Table 2: tripwire Database Initialization Arguments

Argument	Meaning
{-m i | --init}	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.
[-p *polfile*] [--polfile *polfile*]	Use the specified policy 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 write the database file. Mutually exclusive with --no-encryption.
[-d *database*] [--dbfile *database*]	Write to the specified database file.
[-P *passphrase*] [--local-passphrase *passphrase*]	Use the specified passphrase with the local key to sign the database file. Mutually exclusive with --no-encryption.
[-e] [--no-encryption]	Do not sign the database file. The file will still be binary-encoded. Mutually exclusive with --local-passphrase and --local-keyfile.

3.1.2 tripwire Integrity Check Mode

When you run an integrity check, the `tripwire` executable scans the current file system and compares it with the values stored in the database file.

After an integrity check, the Tripwire software will print a summary report of any discovered violations to *stdout* and save a binary copy of the report in the location specified by the REPORTFILE setting in the configuration file. You can use command-line arguments to override configuration file variables and customize the integrity checking process. See section 2.2.1 for examples.

The report files generated by an integrity check can contain a large amount of information. For this reason, you can view report files at 5 different levels of detail. Samples of these reporting levels are presented in Appendix B.

If an integrity check finds changes, you should take necessary steps to diagnose the problem. If the change is potentially malicious, you can replace the affected file and take appropriate security measures. For valid changes (e.g., another system administrator installed new software), you should update the database to reflect the new state of the system.

Running an integrity check using the `tripwire --interactive` option is the same as running an integrity check immediately followed by running `tripwire` in Database Update mode. See section 3.1.3 for more details.

Table 3: tripwire Integrity Check 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.

Table 3: tripwire Integrity Check Arguments

Argument	Meaning
[-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.
[-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 report will be signed. If no passphrase is specified on the command line, the software will prompt for the local passphrase.

Table 3: tripwire Integrity Check Arguments

Argument	Meaning
[-t { 0|1|2|3|4 }] [--email-report-level { 0|1|2|3|4 }]	Specifies the amount of detail to include in an 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.
[*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.

3.1.3 tripwire Database Update Mode

The Integrity Check mode creates a report of changes between actual system objects and the values for those objects stored in the database. Database Update mode displays this report, letting you choose which (if any) records in the database should be updated to reflect valid system changes.

When you run `tripwire` in Database Update mode, the report specified on the command line will be displayed on screen. The editor used to update the database is specified by the `--visual` option, the EDITOR value in the configuration file, or the $VISUAL or $EDITOR environment variables, in that order.

Every rule violation listed in the Object Summary section of the report will have a corresponding “ballot box”:

```
Modified:
[x] "/usr/local/tw"
      drwxr -xr -x root(0)              512 Tue Nov 22 17:19:15 1999
```

To specify entries to be updated, leave the “x” next to each policy violation, indicating that you approve this change to the file system. 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 correct local passphrase, `tripwire` will update and save the database.

Note: If you don’t 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 this value using any time-based variables [$(DATE), etc.], the report will not be found, and a report file must be specified with the `-r` or `--twrfile` option.

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` (the default), 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, we recommend that Database Update mode always be run with `--secure-mode high`.

Table 4: tripwire Database Update Arguments

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

`[-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`.

Table 4: tripwire Database Update Arguments

Argument	Meaning

`[-Z { high|low }]`
`[--secure-mode`
`{ high|low }]`
	Specifies 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.

3.1.4 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 /etc/tripwire/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 /etc/tripwire/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 installations, but can create a security risk if you have been monitoring a system for any length of time. When you re-initialize the 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 administrator decides to modify the policy file rule for that file.

When run in Policy Update mode, Tripwire software 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.

Table 5: tripwire Policy Update Arguments

Argument	Meaning
`{-m p | --update-policy}`	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`.

`[-p` *`polfile`*`]` `[--polfile` *`polfile`*`]`	Update 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 file and to read and write the policy file.
`[-L` *`localkey`*`]` `[--local-keyfile` *`localkey`*`]`	Use the specified local key file to read and write the database file.

`[-P` *`passphrase`*`]`
`[--local-passphrase`
`passphrase``]`
	Use the specified passphrase with the local key to sign the database file.

`[-Q` *`passphrase`*`]`
`[--site-passphrase`
`passphrase``]`
	Use the specified passphrase with the site key to sign the policy file.

Argument	Meaning

`[-Z { high | low }]`
`[--secure-mode`
`{ high | low }]`
	Specifies the security level, which affects whether the policy and database files are saved if a violation of the old policy exists. Violations are always printed to *stderr*, but if security is set to high, no files are changed if any errors occur. The default is high.

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

3.1.5 tripwire Test Mode

You can use Test mode to check the operation of the Tripwire email notification system. When run in this mode, the email notification settings specified in the configuration file will be used to send a test email message.

If MAILMETHOD is set to SMTP, the SMTPHOST and SMTPPORT values will be used to send email. If MAILMETHOD is set to SENDMAIL, the MAILPROGRAM value will be used. 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, and does not check for errors in the syntax used with the `emailto` attribute in the policy file.

Table 6: tripwire Test Arguments

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

3.2 twprint

The `twprint` command enables you to view and print database and report files in plain text form. Tripwire database files are binary encoded and signed. Tripwire report files are encoded and may optionally be signed. The `twprint` application provides a way to view these signed files in text form. If you want to print a policy or configuration file, see section 3.3 for information about the `twadmin` command.

To redirect the screen output from the `twprint` command to a file, use the following command:

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

This file can then be printed to generate a hard copy version of any report or database file.

3.2.1 twprint Print Report Mode

The `twprint` Print Report mode enables you to print the contents of a report. If you do not specify a report with the `--twrfile` command-line option, the default report level specified by the configuration file REPORTFILE variable will be used.

The default filename (as installed) for report files is

```
$(HOSTNAME)-$(DATE).twr
```

where the $(DATE) variable includes the current time to the nearest second. Unless you run the Print Report mode within one second of such a report's creation, `twprint` will be unable to find the report because the $(DATE) variable will have changed to reflect the current time. Therefore, when printing a report, you should always use the `--twrfile` command-line option to specify the report explicitly.

Table 7: twprint Print Report 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.

3.2.2 twprint Print Database Mode

The `twprint` Print Database mode enables you to print the contents of a database to the screen, or to redirect it to a file. If you don't specify a database with the `--dbfile` command-line option, 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 option).

To print the default database file as a text file:

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

To specify another database file to print:

```
twprint --print-dbfile --dbfile otherfile.twd > db.txt
```

To print only certain objects in the database file:

```
twprint --print-dbfile object1 object2 object3...
```

Table 8: twprint Print Database 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.
[*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.

3.3 twadmin

The `twadmin` command has eight command modes, and is used to perform certain administrative functions related to Tripwire files and configuration options.

Create Configuration File mode enables you to designate an existing plain text file as the new Tripwire configuration file.

Print Configuration File mode enables you to print out the current contents of the configuration file in a readable text format.

Create Policy File mode enables you to designate an existing plain text file as the new Tripwire policy file.

Print Policy File mode enables you to print out the current contents of the policy file in a readable text format.

Remove Encryption mode enables you to remove cryptographic signatures from configuration, policy, database, or report files.

Encryption mode enables you to sign configuration, policy, database, or report files cryptographically.

Examine Encryption mode enables you to examine files and report their encryption status.

Generate Keys mode enables you to create site or local keys for Tripwire files.

Security Issue: To prevent unauthorized reading of configuration and policy files, you may want to move the `twadmin` executable to a floppy disk after installation.

3.3.1 twadmin Create Configuration File Mode

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

Table 9: twadmin Create Configuration File 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.

3.3.2 twadmin Print Configuration File Mode

After a configuration file has been created, it is stored in a binary-encoded form. This mode enables you to print out the current contents of the configuration file in a readable text format.

Table 10: twadmin 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.

3.3.3 twadmin Create Policy File Mode

This mode let you designate an existing plain text file as the new policy file. The plain text policy file must be specified on the command line. Using the site key, the new policy file is encoded, signed, and saved.

Although you can modify and save a policy file with this command mode, it is strongly recommended that you use the `tripwire` Policy Update mode instead. 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. This gives tacit (and possibly incorrect) approval that the current state of the file system is an appropriate baseline for future integrity checks.

The `tripwire --update-policy` command updates the policy and the database simultaneously, checking for possible policy violations as it goes. See section 3.1.4 for more details on Policy Update mode.

Table 11: twadmin 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.

3.3.4 twadmin Print Policy File Mode

After a policy file has been created, it is stored in a binary-encoded form. This mode enables you to print out the current contents of the policy file in a readable text format.

Table 12: twadmin 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.

3.3.5 twadmin Remove Encryption Mode

This mode enables you to remove cryptographic signatures from configuration, policy, database, or report files. Multiple files may be specified on the command line, and wildcards can be used to specify files. You will need to enter the appropriate local or site passphrase, or both if a combination of files is to be verified. Even with cryptographic signatures removed, these files will be in a binary-encoded form which is unreadable.

Table 13: twadmin Remove 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.

3.3.6 twadmin Encrypt a File Mode

This mode enables you to sign configuration, policy, database, or report files cryptographically. You can specify multiple files on the command line, and wildcards can be used to specify files. The files will be signed using either the site or local key, as appropriate for the file type. To automate the process, you can include the passphrase for the keyfiles on the command line.

Table 14: twadmin Encryption 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.

3.3.7 twadmin Examine Encryption Mode

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

- filename
- file type
- what key (if any) is used to sign it

Signing type for files will be determined by a trial and error method, using first the site key, and then the local key.

Table 15: twadmin Examine Encryption 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.

3.3.8 twadmin Generate Keys Mode

This mode enables you to create site or local keys for Tripwire files. Although keys are generated by the install process, you can use this command to regenerate keys at any time. The site and local keys may be generated simultaneously or one at a time with two separate invocations of `twadmin`.

Note: Whenever a site or local keyfile is overwritten, any files signed with that key become permanently unusable. Tripwire, Inc. cannot help you recover such files.

When choosing passphrases for keyfiles, remember that effective passphrases should be at least 8 characters in length, and should contain upper- and lowercase letters and numbers. Symbols can be used in passphrases, but 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.

Table 16: twadmin Generate Key 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.

3.4 siggen

You can use the `siggen` utility to display hashes for any specified file. More detailed information about each of the signature functions can be found in the Glossary.

`siggen` displays one or more hash values for any specified file(s) in base 64 notation. 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.

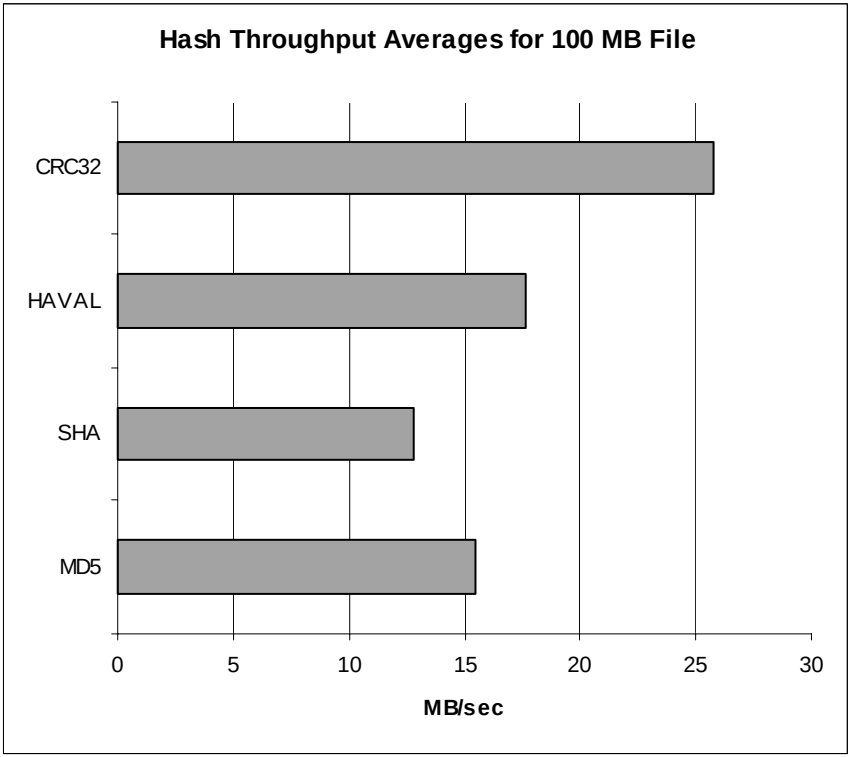
Table 17: siggen Display Signature Arguments

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 base 64 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.

3.4.1 Hash Throughput Performance

The following graph summarizes the mean throughput realized by the four hashes that Tripwire software supports for integrity checking. All of the tests were performed using the `siggen` command and a 100 MB test file on 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. See the glossary for more detailed information about each of the signature functions.



4

Policy Reference

4.0 Overview

This chapter describes the components of the Tripwire policy file. The policy file consists of a series of rules controlling the way that Tripwire software checks the system on which it is installed. For UNIX and Linux systems, the *system objects* that can be scanned are directories and files.

4.1 Policy File Components

Rules, stop points, attributes, directives, and variables are the standard components of the policy file. Each of these components is described in detail in the following sections.

Rules are the basic components of the policy file, specifying the properties to monitor for each system object during an integrity check.

Stop points specify a system object that should not be scanned during an integrity check.

Attributes modify the behavior of rules by controlling recursion or sending email messages.

Directives organize groups of rules, and allow you to apply rules conditionally.

Variables allow you to change information conveniently.

In a policy file, any text following a "#", up to the next line break, is considered a comment.

For example:

```
# This is a comment.
/temp  ->  $(ReadOnly);   # A comment can go here, too.
```

4.2 Rules

Rules associate a system object with a set of properties to be monitored. The format for a rule is:

```
object name -> property mask;
```

where *object name* is the fully qualified pathname for a directory or file, and *property mask* specifies what properties of the object to examine or ignore.

- The `->` token must seperate the object name and the property mask.
- A semicolon must terminate the rule.
- If the object is a directory, the directory, its contents, and all of its descendants will be scanned with the indicated property mask.
- If the object is a file, only that file will be scanned with the specified mask.
- Only one rule may be associated with any given object. If any object has more than one rule in a policy file, any integrity check that uses that policy file will fail.

Examples:

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

# Defines behavior for a single file.  In this case,
# watch 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
```

4.2.1 Object Names

UNIX 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
```

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) ;
```

Special Characters in Object Names

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 ($), hash (#), 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 software 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”
```

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
```

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 recurse 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;
```

4.2.2 Property masks

Property masks describe object properties to examine or ignore. Table 18 lists UNIX property mask characters and a description of what they mean. The following regular expression describes the correct syntax for UNIX property masks:

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

The following general principles apply when using property masks:

- 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.

Table 18: UNIX 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
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
m	Modification timestamp
c	Inode creation/modification timestamp

Table 18: UNIX 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.
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 the 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.
C	CRC-32, POSIX 1003.2 compliant 32-bit Cyclic Redundancy Check. Choose this hash for relatively high performance but relatively low security.
M	MD5, the RSA Data Security, Inc.® Message Digest 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. Specifying all four hashes slows throughput significantly and makes frequent integrity checks laborious.

4.3 Stop Points

Stop points are used to specify objects that should not be scanned during an integrity check. The format for stop points is:

```
! object name;
```

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
```

4.4 Rule Attributes

Rule attributes work with normal rules to modify their behavior or provide additional information. You can also use rule attributes to organize the rules in the policy file into groups. Rule attributes can only be specified for a normal rule, not a stop point or any other item in the policy file.

Rule attributes are case insensitive, and are associated with individual rules with the following syntax:

```
object name -> property mask (attribute = value, attribute = value, ...);
```

For example, if a policy rule is violated, rule attributes can specify an email address to which notification should be sent:

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

Rule attributes can be specified for groups of rules:

```
(attribute = value)
{
    rule1;
    rule2;
    ...
}
```

For example, this policy file text is equivalent to the example above:

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

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 is the `emailto` attribute; these attributes are additive.

For example, given the following policy file rules:

```
(emailto = admin1@foo.com, severity = 90)
{
  /etc/dog-> +pingus (severity = 75);
  /etc/cat-> $(Dynamic) (emailto = admin2@foo.com);
}
```

if the */etc/dog* rule is violated, an email report will be sent to admin1, with a severity level of 75, not 90. If the */etc/cat* rule is violated, email reports will be sent to both admin1 and admin2, and the severity level will be 90.

Table 19: Rule Attributes

Attribute	Description
`rulename`	Associates a name with a rule. The default value is the last element of the object name to which the rule applies.
`emailto`	Specifies email address(es) to which notification of any violations is sent. The default value is *none*.
`severity`	Associates a numeric severity level with a rule. The default value is *0*. The valid range is from 0 to 1000000.
`recurse`	Controls recursive scanning of directories. True (-1), false (0), and numerical values > 0 are valid. The default value is *true*.

4.4.1 rulename

The `rulename` attribute is used to associate a rule or set of rules with a specific name. In the report file created after an integrity check is run, this name will be associated with violations to the specified rule. This feature is useful to track certain objects within a large database file.

You can use the `rulename` attribute to group the rules in your policy file, letting you run integrity checks using only specific rules or groups of

rules. For instance, if you wanted to protect a group of important project files, you might group the following rules in the policy file.

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

If you wanted to run an integrity check using only these rules, you could use the following command:

```
tripwire --check --rule-name “rcfiles”
```

4.4.2 emailto

The `emailto` attribute lets you associate one or more email addresses with a rule. When an integrity check is run with the `--email-report` option and rule violations are found, a report of those violations will be sent via email.

You specify email recipients using their SMTP address:

```
/bin -> +pinug (emailto=admin@mycompany.com);
```

To specify multiple email addresses for a policy rule, include them as a double-quoted, semicolon-delimited list. The names themselves can contain whitespaces, but leading and trailing whitespaces are removed.

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

For each rule, only those persons specified with the `emailto` attribute will receive email notification if that rule is violated. For example, with the following rules in the policy file

```
(emailto=”admin1@mycompany.com;admin2@mycompany.com”)
{
    /temp -> $(IgnoreAll);
    /payroll -> $(ReadOnly) (emailto=admin3@mycompany.com);
}

/projects -> $(ReadOnly) (emailto=admin4@mycompany.com);
```

if the “temp” rule is violated, admins 1 and 2 will receive email notification. If the “payroll” rule is violated, admins 1, 2, and 3 will receive email, and if the “projects” rule is violated, only admin 4 will receive email.

Email reports will be sent only if the `--email-report` argument of the `tripwire` command is specified when the integrity check is run:

```
tripwire --check --email-report
```

All email reports will use the format specified by the EMAILREPORTLEVEL variable in the configuration file, unless another level is specified with the `--email-report-level` option:

```
tripwire --check --email-report --email-report-level 2
```

See Appendix B for sample reports in each of the report formats.

4.4.3 severity

The `severity` attribute associates a numeric severity level with a rule. When `tripwire` is run in Integrity Checking mode, it is possible to specify that only rules exceeding a certain `severity` level be used. For

example, if the following rules are in the policy file

```
/usr/bin -> $(ReadOnly)(severity=90);
/usr/lib -> $(ReadOnly)(severity=75);
```

and an integrity check is run with the `--severity` option,

```
tripwire --check --severity 80
```

the “/usr/bin” rule will be used, but the “/usr/lib” rule will not be used.

You can also specify a value for the `severity` option on the command line with `high`, `medium`, or `low`, which correspond to `severity` levels 100, 66, and 33, respectively.

Values for the `severity` attribute can range from 0 to 1,000,000. If no `severity` is specified for a rule in the policy file, the value defaults to 0.

4.4.4 recurse

The `recurse` rule attribute controls the way that Tripwire software monitors directories. Valid values 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), the inode corresponding to the directory will be scanned, but none of the files or subdirectories in the directory will be scanned. 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.

4.5 Directives

Tripwire software 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 [arguments]
```

Directive names are case sensitive. White space may precede or follow the @@ construct, but no other characters may appear in front of the @@ construct or between the two @ characters. Variables cannot be used to form the directive itself, but variables may be used as arguments to the directive. For example:

```
machine = spock;
@@ifhost  $(machine)      # This is correct syntax...

IFHOST = ifhost;
@@ $(IFHOST)spock     # ...but this will produce an error.
```

The following directives are supported. Each is described in detail below.

Table 20: Tripwire Directives

Directive	Description
`@@section`	Designates a section of the policy file. Used for Windows 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.

4.5.1 @@section

The `@@section` directive is used to label a section of the policy file. The only valid arguments for `@@section` are:

```
FS          for UNIX file system objects
NTFS        for Windows NT file system objects
NTREG       for Windows NT registry objects
GLOBAL      for global variable definitions
```

Each section may be declared only once, and all rules for that section must be specified within that block of the policy file. Tripwire software running on UNIX systems will only process policy file rules from the first `@@section FS` directive to the next `@@section` directive.

If no `@@section` directive is specified in the policy file, the entire policy file will be interpreted as UNIX rules. This may create problems if a policy file designed for Windows NT systems is used with UNIX machines.

The @@section directive may not appear within a scoped rule attribute block:

```
( severity=90 )
{
@@section FS                          # ERROR: @@section may not
/usr/bin -> $(ReadOnly);              # be declared inside a block
}
```

In addition, the @@section directive may not appear within an @@ifhost, @@else, or @@endif block:

```
@@ifhost LIGHTHOUSE
    @@section FS                      # ERROR: @@section may not
    /usr -> $(Dynamic);               # be declared inside a block
@@endif
```

4.5.2 @@ifhost, @@else, @@endif

The @@ifhost, @@else, and @@endif directives are used to permit conditional interpretation of a policy file. The syntax for each is:

```
@@ifhost host1 || host2 || …
@@else
@@endif
```

Where *host1*, *host2*, … are unqualified hostnames and the || notation 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
```

If the unqualified hostname of the machine running Tripwire software 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.

The `@@ifhost` and `@@else` directives can be nested. For example:

```
@@ifhost chekov || uhura
    /bin -> $(ReadOnly);
    /usr/bin -> $(IgnoreNone) -ar;
        @@else
        @@ifhost bones
            /bin -> $(IgnoreNone) -ar;
        @@endif
@@endif
```

4.5.3 @@print, @@error

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

```
@@print string
@@error string
```

Only one string is permitted as a parameter to these directives, so quoting must be used if spaces are included in the message.

```
@@print string         # okay
@@print "Two strings"  # okay
@@print two strings    # ERROR
```

The `@@print` directive merely 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. For example:

```
@@ifhost romeo
    /bin        ->      $(ReadOnly);
@@else
    @@ifhost juliet
        @@print “Scanning projects on juliet”
        /projects       ->      $(ReadOnly) +H;
    @@else
        @@error “This policy file not written for this machine”
    @@endif
@@endif
```

4.5.4 @@end

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

4.6 Variables

For convenience, you can define and use two types of variables in the policy file. Global variables have scope everywhere in the policy file, while local variables defined in the UNIX section of the policy file have scope only within that section.

In the case where 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.

4.6.1 Variable Definition

Global variables must be defined in the global variables section of the policy file (see section 4.6.1). Local variables can be defined anywhere in their section. The syntax for defining global and local variables is the same:

```
variable  =  value;
```

where *variable* is the case-sensitive name for the variable, and *value* must be quoted if it contains any white space or escaped characters.Variable names may contain all alphanumeric characters, underscores, and the characters plus (+), minus (-), at sign (@), colon (:), ampersand (&), percent sign (%), hat (^), and period (.).

Examples of variable definition are:

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

4.6.2 Variable Substitution

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 21: Predefined Variables

Variable	Definition
`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 software 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.

5

Configuration File Reference

5.0 Overview

This chapter describes the configuration file *tw.cfg*, which stores configuration information for Tripwire software, such as the location of the files used for integrity checking.

The installation configuration file *install.cfg* is used only during the installation process, to specify destinations for installed files and the initial values for the *tw.cfg* file. The installation configuration file is described in the INSTALL file in the Tripwire root directory.

5.1 The Configuration File

During installation, Tripwire software uses the values in the *install.cfg* shell script to construct an encoded and signed configuration file. This file, named *tw.cfg*, is located in the */etc/tripwire* directory. A plain text copy of the same file, named *twcfg.txt*, is located in the same directory.

You may wish to edit the Tripwire configuration file to change:

- the way that Tripwire software checks system integrity
- the operation of email reporting
- the default location of Tripwire files

The process for editing the Tripwire configuration file is described in section 3.1.7.

5.1.1 Configuration File Format

The figure below shows a sample Tripwire configuration file. The file is structured as a list of keyword-value pairs. Any line with a # character in the first column is treated as a comment.

```
#
# Sample Tripwire configuration file for a UNIX machine
#
POLFILE = /etc/tripwire/tw.pol
DBFILE = /var/lib/$(HOSTNAME).twd
REPORTFILE = /var/lib/report/$(HOSTNAME)-$(DATE).twr
SITEKEYFILE = /etc/tripwire/site.key
LOCALKEYFILE = /etc/tripwire/$(HOSTNAME)-local.key
EDITOR = /bin/vi
LATEPROMPTING = false
LOOSEDIRECTORYCHECKING = false
MAILNOVIOLATIONS = true
EMAILREPORTLEVEL = 3
REPORTLEVEL = 3
MAILMETHOD = SENDMAIL
MAILPROGRAM = /usr/lib/sendmail -oi -t
# SMTPHOST =
# SMTPPORT =
SYSLOGREPORTING = false
```

The general syntax for variable definition is:

```
keyword = value
TWROOT = /etc/tripwire
```

Variable substitution on the right hand side is permitted using the syntax:

```
$(varname)
$(TWROOT)/policy/tw.pol
```

Variable names must use uppercase characters; variable values are not case-sensitive, and may include all valid ASCII characters. Variables must be defined before they are used.

Two variables are predefined and may not be changed:

- HOSTNAME is the unqualified hostname of the computer on which Tripwire software is running
- DATE is a string representation of the date and time (e.g. 20000127-061033).

5.2 Configuration File Variables

Removing a non-required variable from the configuration file is equivalent to setting the value for that variable to false.

Required Variables

The following five variables specify the location of Tripwire files used to check integrity. These variables must have a value in the configuration file for proper operation. Relative pathnames are permitted, expressed in relation to the directory where the Tripwire executables reside.

POLFILE
Default policy file.
Initial value: /etc/tripwire/tw.pol

DBFILE
Default database file.
Initial value: /etc/tripwire/$(HOSTNAME).twd

REPORTFILE
Specifies name of generated reports.
Initial value: /var/lib/report/$(HOSTNAME)-$(DATE).twr

SITEKEYFILE
Specifies default site key file.
Initial value: /etc/tripwire/site.key

LOCALKEYFILE
Specifies default local key file.
Initial value: /etc/tripwire/$(HOSTNAME)-local.key

Optional Variables

EDITOR
Defines the absolute pathname to an editor to be used in interactive modes. Valid editors for use with Tripwire files must be able to accept a file on the command line, must support multi-byte characters, and must exit with 0 status on success and non-0 status on error.

If EDITOR is not defined, and no editor is specified on the command line, Tripwire software will look at the $VISUAL or $EDITOR environment variables. If no editor is defined by these sources, specifying interactive mode will cause an error.

Initial value: /bin/vi

LATEPROMPTING
If set to true, Tripwire software will prompt for passphrases as late as possible to minimize the amount of time that the password is stored in memory.

Initial value: false

SYSLOGREPORTING
When this option is set to true, Tripwire software will post records of database initializations, integrity checks, database updates, and policy file updates to the syslog. UNIX syslog messages are sent from the “user” facility at the “notice” level. See syslogd(1) for more information.

Initial value: true.

LOOSEDIRECTORYCHECKING
When a file is added or removed from a directory, both the changes to the file itself, and the modification to the directory are reported. This can create redundant entries in Tripwire reports. If the value for this variable

is true, directory properties that would change when a file was added or deleted (last write and last access times) will be ignored.

Initial value: false

REPORTLEVEL
Specifies the level of detail (0-4) for reports that are printed with the `twprint --print-report` command. If a different report level is specified on the command line, this setting will be overridden.

Initial value: 3

Email Notification Variables

MAILMETHOD
Specifies the protocol used by Tripwire software for email notification. The only valid values for this variable are SMTP and SENDMAIL.

Initial value: SENDMAIL

SMTPHOST
Specifies the domain name or IP address of the SMTP server used for email notification. Ignored unless MAILMETHOD is set to SMTP.

Initial value: (*commented out*)

SMTPPORT
Specifies the port number used with SMTP. Ignored unless MAILMETHOD is set to SMTP.

Initial value: (*commented out*)

MAILPROGRAM
Specifies the full path to the program used for email reporting if MAILMETHOD is set to SENDMAIL. Any program specified must take an RFC822 style mail header, and 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 specified must be able

to ignore lines that consist of a single period (the -oi option to sendmail produces this behavior).

Initial value: /usr/lib/sendmail -oi -t

EMAILREPORTLEVEL
Specifies the level of detail (0-4) for email reports that are sent as a result of an integrity check. If a different email report level is specified on the command line, this setting will be overridden.

Initial value: 3

MAILNOVIOLATIONS
This option controls the way that Tripwire software sends email notification if no rule violations are found during a check:

If this option is set to false and no violations are found, no report will be sent. If set to true and no violations are found, an email message stating that no violations were found will be sent.

Mailing reports of no violations enables an administrator to distinguish between unattended integrity checks that are failing to run and integrity checks that are running but are not finding any violations. However, mailing no violations reports will increase the amount of data that must be processed.

Initial value: true

Appendices

Appendix A: Tripwire Exit Codes

Exit codes for `tripwire` Integrity Checking mode:

Table 22: 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 23: Exit codes for all other command modes

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

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 choose to view only a subset of the report information 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 information will always appear as the subject line of any Tripwire email report. The format for this report is:

```
TWReport LIGHTHOUSE 19991021134026 V:45 S:100 A:2 R:1 C:6
```

- 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 restore tampered files automatically, or to automate some other response. The format for this report is:

```
Added:  /usr/bin/bash
Modified:       /usr/bin
Modified:       /etc/passwd
Modified:       /var/sysadm/salog
```

Level 2: Summary Report

This report lists the violations that occurred by rule name. The Object Summary section provides more detail on the affected files or registry entries.

```
Tripwire(R) 2.3.0 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:             /etc/tripwire/tw.pol
Configuration file used:      /etc/tripwire/tw.cfg
Database file used:           /var/lib/web3.twd
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                  66                0        0        0
  (/usr/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 ***

Tripwire 2.3 Portions copyright 2000 Tripwire, Inc. Tripwire is a registered
trademark of Tripwire, Inc. This software comes with ABSOLUTELY NO WARRANTY;
for details use --version. This is free software which may be redistributed
or modified only under certain conditions; see COPYING for details.
All rights reserved.
```

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.

```
Tripwire(R) 2.3.0 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:             /etc/tripwire/tw.pol
Configuration file used:      /etc/tripwire/tw.cfg
Database file used:           /var/lib/web3.twd
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                  66                0        0        0
  (/usr/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 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 ***

Tripwire 2.3 Portions copyright 2000 Tripwire, Inc. Tripwire is a registered
trademark of Tripwire, Inc. This software comes with ABSOLUTELY NO WARRANTY;
for details use --version. This is free software which may be redistributed
or modified only under certain conditions; see COPYING for details.
All rights reserved.
```

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.

```
Tripwire(R) 2.3.0 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:             /etc/tripwire/tw.pol
Configuration file used:      /etc/tripwire/tw.cfg
Database file used:           /var/lib/web3.twd
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                  66                0        0        0
  (/usr/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"

================================================================================
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 ***

Tripwire 2.3 Portions copyright 2000 Tripwire, Inc. Tripwire is a registered
trademark of Tripwire, Inc. This software comes with ABSOLUTELY NO WARRANTY;
for details use --version. This is free software which may be redistributed
or modified only under certain conditions; see COPYING for details.
All rights reserved.
```

Glossary

Asymmetric Cryptography

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

Attribute

See *Rule attribute*.

Command modes

The various functionally distinct aspects of Tripwire commands. Every Tripwire command except `siggen` has modes, which define which command-line arguments are valid and what they will do.

Configuration file

The file that stores information and settings, such as the location of data files, that Tripwire software 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 keyfile*.

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.

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*.

Database file

A Tripwire file representing a “snapshot” of a system that serves as the baseline for Tripwire integrity checks. The database file is used for most Tripwire operations, and should be created from a system in a known secure state. The database file is encoded and signed with the *local key file*, and 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 attributes of the objects 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. This is the definition used by the market research group’s ICSA paper “Intro to Intrusion Detection and Assessment.”

Generate Keys mode

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

Global variable

A variable in the Tripwire policy file that has scope everywhere in the policy file. Global variables must be defined in the GLOBAL section of the policy file, and cannot be defined using any other variable. 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.

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 software configured with a 128-bit signature using four passes to ensure pseudo-random output.

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 and registry against the values stored in the Tripwire *database file*. The Integrity Check mode generates a *report file* listing all of the violations discovered by the integrity check.

Key file

Stores the public and private keys for a key pair. Tripwire software 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 unusable.

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.

Local variable

A variable in the Tripwire policy file that has scope only within the section of the policy file (filesystem or registry) where it is defined. 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.

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.

Message-digest algorithm

Algorithms used to render files tamper-evident. If an input data file is changed, the checksum for that file will also change. Small changes in a file will cause large changes in the message-digest value.

Object name

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

Passphrases

Long passwords that control user access to important Tripwire files. Tripwire software 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 does not necessarily have the ability to compromise other machines.

Policy file

A file, consisting of a series of *rules*, that controls the way that Tripwire software checks the integrity of a system. Each rule in the policy file specifies a system object that is monitored, 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 command mode of the `tripwire` command 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 site and local key files that signs files with the site or local key.

Property mask

The part of a policy file rule that specifies the property or properties of a directory or file to monitor.

Public key

A component of Tripwire site and local key files that verifies files that are signed. In Tripwire software, public keys permit reading only, while private keys permit writing.

Recurse

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

Rule

A statement in the policy file 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. Rules must end with a semicolon or a syntax error will occur.

Rule attribute

Rule attributes modify the *rules* in the Tripwire policy file on a per-rule basis.

Rule name

The name given to a rule, which is used 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. It is referred to here as the SHA, or Secure Hash Algorithm, because Tripwire software uses a non-certified implementation and cannot claim standards conformance. SHS generates a 160-bit signature.

`siggen`

A file utility that displays hash values for system objects. `siggen` provided the performance output cited for the four signatures included in Tripwire software: 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, registry key, or registry value that can be monitored by Tripwire software.

Test mode

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

Tripwire

The uppercase form of this name refers to the product's files and programs.

`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 and registry against that database file.

`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 system object that violates a *rule* in the Tripwire policy file.

Index

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

w