TRIPWIRE
SECURITY-SYSTEMS-INC

Tripwire™ Academic Source Release 1.3.1 for Unix®

User Manual

April 30, 1999

TRIPWIRE
SECURITY - SYSTEMS - INC

615 SW Broadway, Second Floor
Portland, OR 97205, USA
tel: 503.223.0280
fax: 503.223.0182
(www.tripwiresecurity.com)
Email: tripwire@tripwiresecurity.com

Configuring and Installing Tripwire

Follow the steps below to properly configure, install, and use Tripwire.

1. Read the “BUILDING TRIPWIRE ASR 1.3.1 for Unix” section.
2. Read the “INSTALLING AND RUNNING” section.
3. Read the FAQ section.
4. Edit the include/config.h file to set the appropriate values for your site.
5. Go to single user mode for the remaining steps.
6. Reinstall your operating system binaries to be certain they have not been compromised.
7. Run Tripwire in database initialization mode.
8. Set the destination directory's disk to *read-only* in hardware, and/or take standalone signatures of all the files using the *siggen* utility.
9. Resume normal operations.

It is strongly recommended that you take these steps so that in the event of a security incident, you can repair any damaged data in the shortest possible amount of time.

TABLE OF CONTENTS

OPEN LETTERS TO TRIPWIRE CUSTOMERS

Open Letter to Tripwire Customers:

January, 1999

Dear Customer,

As President of Tripwire Security Systems, Inc., I am pleased to announce our acquisition of exclusive worldwide rights of Tripwire, developed at Purdue University in conjunction with many individual, corporate and institutional testers and users. Please find an announcement and transition letter from Professor Spafford of Purdue below.

Since its original release in 1992, Tripwire has grown in popularity and is now one of the most widely deployed UNIX security tools. Several hundred thousand software downloads have been recorded, and we are aware of many active large government and commercial sites using the product.

Our goals for Tripwire are to continue with the excellent effort put forth by the original authors and many contributors since the code was originally released. We are planning several incremental releases in the immediate term adding value to the core engine and user interface. In January 1999 we released our first major new commercial version, Tripwire 2.0.

We encourage existing and new users of Tripwire to stay tuned to our web site at www.tripwiresecurity.com for more information about future releases of Tripwire and other marketing related activities. If you have any specific input or questions, please direct these inquiries to tripwire@tripwiresecurity.com.

Sincerely,

W. Wyatt Starnes
President
Tripwire Security Systems, Inc.

Open Letter from Professor Eugene Spafford, Ph.D.:

Dear Tripwire customer,

As you may know, Purdue University has elected to transfer management and product responsibility for the Tripwire security system to its co-developer Mr. Gene Kim and his firm Tripwire Security Systems, Inc. of Portland, Oregon.

This decision was made to ensure the continued development and support of Tripwire and to maintain the integrity of its design goals.

The results of the arrangement will benefit you, Purdue University and Tripwire Security Systems, Inc., by allowing the product to evolve and receive the necessary focus and resources needed to maintain its functionality in years to come.

Furthermore, by placing Tripwire with a focused commercial entity, we believe that there will be resources and motivation to enhance and port Tripwire to a wider range of uses and platforms. This can only help the user community concerned with security.

As a business, Tripwire Security Systems, Inc. will be able to invest the time and resources for quality technical support. They will also be in a better position to respond to customer suggestions and integrate valuable new features into future Tripwire releases.

A representative of Tripwire Security Systems, Inc. will be available to discuss transferring any current use of Tripwire into a formal site license arrangement.

Thank you for making Tripwire one of the most successful security programs ever written. I look forward to reviewing the progress of our decision to focus the resources needed to further advance the program's functionality and value.

Gene Spafford, Ph.D.

About This Document

This documentation is a Tripwire primer. It contains information needed to build, test, and run Tripwire. For a full description of the design and rationale of the technology, see the ./docs directory for the Tripwire design document, which is also available as a technical report (TR-CSD-93/71).

About Tripwire

Tripwire is a file integrity assessment tool, a utility that compares a designated set of files and directories against information stored in a previously generated database. The utility flags all differences, including added or deleted entries. When run against system files on a regular basis, any changes in critical system files will be visible, and appropriate damage control measures can be taken immediately. System administrators can conclude with a high degree of confidence that a given set of files remain free of unauthorized modifications if Tripwire reports no changes.

Tripwire was built and tested on a wide variety of Unix variants. As of this writing, Tripwire has run successfully on BSD, OSF/1, Mach, Xenix, and late System V variants of Unix.

How to Build Tripwire

1. Please review the README file before you modify any files or try to build the program. This text describes various settings and strategies for configuration and operation. Once you've considered your options for system settings and configuration strategies, return to this section and follow the instructions.
   The file 'Ported' contains a list of platforms and operating systems where Tripwire has already been successfully ported. If you find your system in the list, note the system settings that were used to build Tripwire.
2. Look through the Makefile and make sure that the C compiler and all flag settings are appropriate for your configuration. Most of the system settings should be listed in the 'Ported' file.
3. Look in the './configs' directory for a predefined 'conf-<os>.h' header file that matches your operating system. (You will be inserting this filename in the './include/config.h' file.) If no such file exists, pick one similar to your system type and modify it appropriately. Mail the modified type to support@tripwiresecurity.com.

   Do <u>not</u> copy this file over the config.h file.
4. Edit the './include/config.h' file to tailor Tripwire to your system by including the name of the predefined header file most similar to your system at the appropriate line in config.h.

   Paths and names of Tripwire configuration files are also set in the config.h file. Make sure you note the locations that Tripwire looks for its configuration and database files; change them for your system, as appropriate.

Important: Place the Tripwire configuration files on a disk that can be made read-only with a hardware setting. This will prevent the files from being altered by an attacker. The run-time version of Tripwire should be located in the same place. If you are unable to mark a disk (or diskette) as read-only, you might also consider putting it on a remote partition of a more secure machine, and import it read-only. See the design document for the rationale if you have any questions about why this is so important.

5. Look in the './configs' directory again for a tw.config file that matches your operating system. These files were tailored to match the file layouts of various vendor-supplied operating systems. If no file in this subdirectory matches your system, choose the one that is closest in nature.
6. Edit this file to include local binaries and critical databases, and any additional files you want to monitor. Correct path specifications if you have moved things or if they are mounted from a remote location (check them only on the server!), and exclude locally-active files from the check. You should add the Tripwire binary itself to the configuration file to ensure that any tampering with it can be detected in the future. Details for how to edit the configuration file are in the section entitled "How to configure Tripwire."

   After you have customized your tw.config file, copy it to the location that you specified in your config.h file.
7. After you have configured the build environment, simply type 'make' at the top level. Note that all Makefiles in the subdirectories are driven by the top-level Makefile. (i.e., typing 'make' in the ./src directory will probably not work.)

Common Tripwire compilation problems

Tripwire was originally written using ANSI C. However, Tripwire now compiles with K&R, too. All of the prototypes remain embedded between "#ifdef __STDC__" directives. Compiling under ANSI is sometimes noisier than with K&R. We recommend that you compile with K&R unless your site requires ANSI. (The code lints completely clean, excepting the malloc() and exit() return values.)

Common compilation issues are the dirent(S5)/direct(BSD) funkiness and #defines that changed for POSIX compliance.

If the Tripwire test suite fails, do the following:

- Check for the proper conf-*.h file in your ./include/config.h file.
- Change the CFLAGS definition in the Makefile so no optimization is done (i.e., remove the "-O" option).
- Do a "make clean".
- Try to compile again.

If you use flex and bison, make sure you use the gcc compiler to avoid compile-time errors. If this fails, try a different C compiler (e.g., GCC).

It has been noted that newer versions of flex and bison, the GNU replacements for lex and yacc, do not generate code that passes all the test suites using the default Tripwire sources. Calvin Page helped contribute replacement config.pre.l and config.pre.y files that correct this problem. Using lex usually solves the problems that made the Tripwire test suite fail.

Please note that the SHA code in ./sigs/sha seems to be poorly handled by many optimizing C compilers. For example, the stock C compiler included with SunOS 4.x takes almost two minutes to compile this file with the -O option on a Sparcstation10. Other compilers (such as GCC) do not have this problem.

Testing Tripwire

Tripwire includes a script-driven test suite that checks the top-level build directory against the distribution package.

In the ./tests directory, there is a Tripwire database of the entire Tripwire source distribution and a tw.config file. The test script automatically converts the pathnames in these Tripwire files to match those of your system. After converting the files, it then runs Tripwire in Integrity Checking mode.

To run the test, simply type ‘make test’ at the top level. This will invoke the script, and if all goes well, the output of Tripwire matches the expected values that the script provides.

In addition to checking all the files in the Tripwire distribution, a number of signature and functional tests are run to ensure the correct operation of the compiled program.

HOW TO CONFIGURE TRIPWIRE™

Some Tripwire scaling hints for using Tripwire in large sites

The tw.config.5 manual page describes in detail the syntax supported by the tw.config file. Tripwire includes features that offer similar functionality to the C-preprocessor, and offer other directives that assist in the use of Tripwire at sites consisting of hundreds of workstations with local disk storage.

The tw.config grammar

These commands are briefly described below:

```
@@define VAR VALUE
@@undef VAR

@@ifhost HOSTNAME
@@ifnhost HOSTNAME
@@ifdef VAR
@@ifndef VAR
@@else
@@endif

@@include FILENAME
```

Furthermore, the tw.config grammar also supports logical expressions. For example, you could have something like this in your tw.config file:

```
@@ifhost spam.cc.purdue.edu || weiner.cc.purdue.edu
...entries...
@@endif
```

Besides the cpp-like functionality, you can use @@define to create strings that are interpreted at run-time.

For example:

```
@@ifhost mentor.cc.purdue.edu
@@ define TEMPLATE_S   +pinug-cas0123456789
@@else
@@ define TEMPLATE_S   +pinug012-cas3456789
@@endif

/etc/tw.loginfo               @@TEMPLATE_S
```

How you might use these directives

Because Tripwire allows run-time interpretation of the tw.config file, it is possible for many different hosts to share the same tw.config file. This allows the maintenance of Tripwire configuration files to be manageable in a large, heterogeneous environment. Although each host must still have different database files, this has few consequences, except for disk space.

Notes on signature routines

The RSA Data Security, Inc. MD5, MD4, and MD2 Message Digesting Algorithm, Snefru (the Xerox Secure Hash Function), SHA (the Secure Hash Algorithm), and Haval code have been changed to eliminate big-endian and little-endian run-time specific routines. These changes have been sent back to the authors, but we are not aware of any buy-backs yet. Until then, there will remain some differences between the code in this package and their respective original distributions.

Performance vs. security

Balancing issues of security and speed is a decision best made by the administrator.

Normally, only one checksum per file would be enough to detect changes. For purposes of speed, an easily calculated checksum would be preferred. However, most easy-to-calculate signatures are also easy to defeat if a determined attacker wished to do so (see the chart in the design document to see how easy this is to do with random comparisons).

Tripwire includes the MD2, MD4, SHA, and Haval signature algorithms, as well as the 16 and 32-bit CRC algorithms. Using the default setup of recording two signatures (MD5 and Snefru) for each database entry gives very, very strong assurance that a file has not been tampered with. For tampering to have succeeded, the attacker would have had to have changed the file and added appropriate padding characters to recreate **both** checksums without also altering the size of the file. To do this at random might not even be possible with the MD5 and Snefru checksums used. These two algorithms have not been exhaustively analyzed, but both are known to be strong message authentication codes.

This added assurance is at a heavy price, however. The two algorithms, and Snefru in particular, are expensive to calculate. To run the MD5 and Snefru algorithms against every file is likely to be overkill for almost all systems. Both checksums should be run over only the most critical files...like the Tripwire database and program, and perhaps each setuid and setgid file on your system. All other files can be checked with MD5 (or Haval) alone for much faster operation and a high level of assurance. The task of altering a file and recreating the original MD5 checksum is also very difficult, and it is unlikely that any but the most determined, sophisticated, and well-equipped attacker would be able to do it in finite time.

To decrease the execution run-times of Tripwire, consider modifying your tripwire.config entries to ignore the Snefru (signature 2) attribute on files that do not need such stringent monitoring. This will skip the computationally-expensive Snefru signature collection entirely.

Be forewarned, however, that MD2 is an order of magnitude slower than even Snefru, and probably guarantees no greater integrity checking. We include all these routines, however, so you can pick what you feel to be most appropriate for your site.

You may wish to add other routines as checksum/signature generators. For instance, if you have a fast DES implementation (including chip-based generation), you might wish to encrypt the file using CBC mode and some fixed key, saving the final 128 bits of output as the signature. The configuration file routines have several signature flags that are currently bound to a null function, so there is room for this expansion if you wish.

Clearly, with eight different signature algorithms at your disposal, Tripwire offers considerable flexibility in ensuring data security. Tripwire makes maintaining a trivial CRC database equally easy to administer and check as a full (but perhaps less practical) eight-signature database.

The following section describes each of the eight signature algorithms.

Signature routines

This section briefly describes each signature routine. This is by no means an authoritative list, but it does attempt to give some background on each of the signature routines provided:

MD5, Snefru, MD4, MD2, SHA, and Haval are all examples of message-digest algorithms (also known as one-way hash functions, fingerprinting routines, message authentication codes, or manipulation detection codes). They employ cryptographic techniques to ensure that any small change in the input stream results in immediate and widely diverging output. This way, even a small change in the input results in large change in the output. Furthermore, because these algorithms use a 128-bit or larger signature, using a brute-force attack to introduce a deliberate change in the file while trying to keep the same signature becomes a computationally infeasible task.

The CRC algorithms, on the other hand, use simple polynomial division to generate the checksums. While this technique is very fast, the mathematics of this technique is well-understood. Additionally, since the signature space is so small (usually 16 or 32 bits), a brute-force search for a CRC collision is well within the capabilities of most workstations. There are currently several programs in the public domain that can, for any given input file, provide a different output file with the same CRC signature in 30 seconds or less.

All observed timing measures provided for the signature routines were performed on a Sequent Symmetry with ten 16 MHz 80386 processors. The numbers provided are simply an informal gauge of throughput, rather than any authoritative metric.

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 via anonymous FTP from NIC.DDN.MIL or from RSA.COM as ~/pub/md5.doc. MD5 attempts to address potential security risks found in the speedier, but less secure MD4, also by RSA Data Security Inc. MD5 was designed as a more conservative algorithm that backs “away from the edge” in terms of risks from successful crypto-analytic attack.

MD5 generates a 128-bit signature, and uses four rounds to ensure pseudo-random output. Observed throughput is about 70 Kbytes/second.

Currently, MD5 is considered by many to be a state-of-the-art signature algorithm.

Snefru

Snefru, the Xerox Secure Hash Function, was developed by Ralph Merkle at Xerox PARC. As an incentive to find a Snefru crack, there is a $1000 cash prize promised to anyone who can find two sets of input that map to the same signature.

This reward has remained unclaimed since April 1990, when the 2-pass version of Snefru was broken by Eli Biham, a Ph.D. student of Adi Shamir. Currently, Ralph Merkle recommends using only the 4-pass version of Snefru, if not the 8-pass version. The Snefru README states, "Further study of the security of Snefru is required before production use is advisable."

As shipped with Tripwire, Snefru is configured to run in 4-passes. Version 2.5 is the latest version available, and is the version included with Tripwire.

Snefru is slower than MD5, but is recommended as a backup for MD5 as a primary signature. As configured, Snefru runs at about 31 Kbytes/second.

Snefru can be obtained via anonymous FTP from arisia.xerox.com in directory /pub/hash.

CRC-32

Cyclic Redundancy Checks have been the long been the de facto error detection algorithm standard. These algorithms are fast, robust, and provide reliable detection of errors associated with data transmission. It has been shown that CRC-32 has a minimum distance of 5 for block lengths of less than 4K. However, this decreases as the size of the blocks increases. Therefore, using CRC-32 on long files is certainly a misapplication of this signature algorithm. However, CRC-32 is provided as a fast and speedy alternative to the slower message-digest algorithms. The version of CRC-32 included with Tripwire was written by Gary S. Brown.

This CRC-32 implementation runs at about 111 Kbytes/second.

CRC-16

CRC-16 is the predecessor to CRC-32, using only 16 bits to store to the remainder of the data and the generator polynomial. CRC-16 is typically at the link level, usually done in hardware to detect transmission errors.

This CRC-16 implementation runs at abut 131 Kbytes/second.

MD4

MD4, the RSA Data Security Inc. Message-Digest Algorithm, is the predecessor to MD5 described above. It was also submitted as a standard data authentication algorithm, and is described in the Internet Working Draft 1320.

The MD4 algorithm was designed to exploit 32-bit RISC architectures to maximize throughput. On a Sun SparcStation, throughput rates of over 1.4 Mbytes/second are achieved.

MD4 can be obtained via anonymous FTP from RSA.COM in ~/pub.

On a Sequent, MD4 throughput is about 332 Kbytes/second.

MD2

The RSA Data Security, Inc. MD2 Message-Digest Algorithm was created as part of the Privacy Enhanced Mail package—a package designed to authenticate and increase the security of electronic mail. Like the other algorithms by RSA Data Security, Inc presented here, MD2 generates a 128-bit signature.

The MD2 algorithm is quite slow. On a 16 MHz 80386, expect only 3 Kbytes/second. It is not clear that using this slower algorithm instead of MD5 brings any comparative advantage.

The license for MD2 specifically states its use is exclusive to the Privancy Enhanced Mail package. Provisions have been made with RSA Data Security, Inc. for its inclusion and use in Tripwire in its present form. Note that MD2 is not in the public domain.

SHA/SHS

SHS is the NIST Digital Signature Standard, called the Secure Hash Standard. It is described in NIST FIPS 180. We refer to it as the SHA, or Secure Hash Algorithm, because we are using a non-certified implementation and we cannot claim standards conformance.

SHA is about one-half as fast as MD5. It has been noted that SHS appears to be largely based on MD4 with several key enhancements, not all implemented in MD5. The mid-1994 correction to the algorithm at the behest of NSA (reflected in the default version compiled into Tripwire; see sigs/sha/sha.h) raises some questions about the overall strength of both SHA and MD4 in the minds of some cryptographers.

Haval

Haval was written by Yuliang Zheng at the University of Wollongong, and is described in Y. Zheng, J. Pieprzyk and J. Seberry:

“HAVAL --- a one-way hashing algorithm with variable length of output”, Advances in Cryptology --- AUSCRYPT’92, Lecture Notes in Computer Science, Springer-Verlag, 1993.

Haval is shipped with Tripwire configured similarly to the other signature algorithms: 128 bit signature using four passes. Configured this way, Haval throughput is approximately 100K/sec. (30% faster than MD5.)

Null signature

This signature, sig_null_get(), is not really a signature algorithm. Instead, it is a place holder for unused slots in the signature array. It will always return a single character, “0”.

Feedback and bug-reports

Please send any bug-reports, questions, feedback, or any comments to:
Support@tripwiresecurity.com

User contributions and experiences

The ./contrib directory contains several programs contributed by users during the beta-test period. Each program is accompanied by a README file written by the program author.

HOW TO RUN TRIPWIRE

Tripwire runs in one of four modes: Database Generation, Integrity Checking, Database Update, and Interactive Update mode. In order to run Integrity Checking, Tripwire must have a database to compare against. To do that, you must first specify the set of files for Tripwire to monitor. This list is stored in tw.config.

Creating your tw.config file

Edit your 'tw.config' file, or whatever filename you defined for the Tripwire config file, and add all the directories that contain files that you want monitored. The format of the config file is described in its header and in the man page. Pay especially close attention to the select-flags and omit-lists, which can significantly reduce the amount of uninteresting output generated by Tripwire. For example, you will probably want to omit files like mount tables that are constantly changed by the operating system.

Next, run Tripwire with 'tripwire -initialize'. This will create a file called 'tw.db_[hostname]' in the directory you specified to hold your databases (where [hostname] will be replaced with your machine hostname).

Tripwire's detection of changes

Tripwire will detect changes made to files from this point on. You must be certain that the system on which you generate the initial database is clean, however --- Tripwire cannot detect unauthorized modifications that have already been made. One way to do this would be to take the machine to single-user mode, reinstall all system binaries, and run Tripwire in initialization mode before returning to multi-user operation.

This database must be moved to a place where it cannot be modified. Because data from Tripwire is only as trustworthy as its database, choose this site with care. We recommend placing all the system databases on a read-only disk (you need to be able to change the disk to writable during initialization and updates, however), or exporting it via read-only NFS from a "secure-server." (This pathname is hardcoded into Tripwire. Any time you change the pathname to the database repository, you must recompile Tripwire. This prevents a malicious intruder from spoofing Tripwire into giving a false "okay" message.)

We also recommend that you make a hardcopy printout of the database contents right away. In the event that you become suspicious of the integrity of the database, you will be able to manually compare information against this hardcopy. We have yet to hear of a way for "crackers" to alter an old piece of printout made before they penetrated the system!

You may also wish to generate a full set of signatures of the database, the configuration file, and the Tripwire executable using the "siggen" utility. (Be certain to generate siggen's signature, too.) Store these on hardcopy for comparison if you need quick confirmation that the files involved have not changed. However, we advise that you do any comparison via a version of siggen stored on read-only media, or encrypted when not in use.

Running Tripwire as an integrity checker

Once you have your database set up, you can run Tripwire in Integrity Checking mode by entering the command ‘tripwire’.

Keeping your database up-to-date

A common setup for running Tripwire would mail the system administrator any output that it generates. However, some files on your system may change during normal operation, and this necessitates update of the Tripwire database.

There are now two ways to update your Tripwire database. The first method is interactive, where Tripwire prompts the user whether each changed entry should be updated to reflect the current state of the file, while the second method is a command-line driven mode where specific files/entries are specified at run-time.

Running Tripwire in Interactive mode

Running Tripwire in Interactive mode is similar to the Integrity Checking mode. However, when a file or directory is encountered that has been added, deleted, or changed from what was recorded in the database, Tripwire asks the user whether the database entry should be updated.

For example, if Tripwire were run in Interactive mode and a file’s timestamps changed, Tripwire would print out what it expected the file to look like, what it actually found, and then prompt the user whether the file should be updated:

```
/homes/research/tw/src/preen.c
      st_mtime: Wed May  5 15:30:37 1993      Wed May  5 15:24:09 1993
      st_ctime: Wed May  5 15:30:37 1993      Wed May  5 15:24:09 1993
---> File: '/homes/research/tw/src/preen.c
---> Update entry?  [YN(y)nh?] y
```

You could answer yes or no, where a capital ‘Y’ or ‘N’ tells Tripwire use your answer for the rest of the files. (The ‘h’ and ‘?’ choices give you help and descriptions of the various inode fields.)

While this mode may be the most convenient way of keeping your database up-to-date, it requires that the user be “at the keyboard.” A more conventional command-line driven interface exists, and is described next.

Running Tripwire in Database Update mode

Tripwire supports incremental updates of its database on a per-file/directory or tw.config entry basis. Tripwire stores information in the database so it can associate any file in the database with the tw.config entry that generated it when the database was created.

Therefore, if a single file has changed, you can:

```
tripwire -update /etc/newly.installed.file
```

Or, if an entire set of files that made up an entry in the tw.config file changed, you can:

```
tripwire -update /usr/local/bin/Local_Package_Dir
```

In either case, Tripwire regenerates the database entries for every specified file. A backup of the old database is created in the ./databases directory.

Note that Tripwire can now handle arbitrary numbers of arguments in Database Update mode. This was added in version 1.0.1.

The script “twdb_check.pl” was added in version 1.2 as an interim mechanism to ensure database consistency. Namely, when new entries are added to the tw.config file, database entries may no longer be associated with the proper entry number. The twdb_check.pl script analyzes the database, and remaps each database entry with its proper tw.config entry.

The twdb_check functionality will be put into the Tripwire program in a future release.

Quick Integrity Checking mode

Tripwire allows you to selectively skip certain signatures at run-time through a command-line option. For example, if you wish to run Tripwire on an hourly basis, even performing only MD5 checks might be computationally prohibitive. For this application, checking only the CRC32 signature might be desirable. To do this, assuming that only MD5, Snefru, and CRC32 were used when the database was initialized, you would type:

```
tripwire -i 1 -i 2
```

This tells tripwire to ignore signature 1 and signature 2. Furthermore, for daily Tripwire runs, you could specify using only MD5 and CRC32. Finally, for weekly runs, you could run Tripwire with all three signatures.

To find added or deleted files, with no signature checking, use:

```
tripwire -i all
```

The siggen utility

The siggen utility is provided so users can get signatures of files without having to run Tripwire. The syntax of siggen is simple.

```
siggen [-0123456789aqv] [ file ... ]
```

By default, siggen prints out all ten signatures. However, the signatures can be printed selectively by specifying the signature number on the command line. See the manual page for details.

FREQUENTLY ASKED QUESTIONS

Listed below are the most frequently asked questions about Tripwire. The first section of the file covers Tripwire concepts and design, while the second section addresses troubleshooting.

Concepts:

Q: Does Tripwire have any Year 2000 (Y2K) problems?

A: Tripwire 1.3.1 ASR has been minimally tested for Y2K compliance. This version of Tripwire uses the POSIX ctime() function common to most other UNIX utilities. Dates are fully represented by 26 characters which allows dates after the year 2000 to be unambiguously displayed. (e.g., Fri Sep 13 00:00:00 2003) Although it has passed Y2K tests and no problems are expected, Tripwire 1.3.1 ASR has not been certified Y2K compliant

Q: Why doesn’t Tripwire traverse mounted file systems?

A: This is intentional. This behavior makes it possible to put a directory (e.g., ‘/’) in your tw.config file, and you won’t have to worry whether it will traverse all the locally-mounted file systems. If you want it include the whole file system, list each partition separately in the configuration file.

Q: What is the difference between pruning an entry in your tw.config file (via “!”) and ignoring everything (via the “E” template)?

A: Ignoring everything in a directory still monitors for added and deleted files. Pruning a directory will prevent Tripwire from even looking in the specified directory.

Q: Tripwire runs very slowly. What can I do to make it run faster?

A: You can modify your tw.config entries to skip the Snefru signatures by appending a “-2” to the ignore flags. Or you can tell Tripwire at run-time to skip Snefru by:

```
tripwire -i 2
```

This computationally expensive operation may not be needed for many applications. (See README section on security vs. performance trade-offs for further details.)

Troubleshooting:

Q: I build Tripwire and the test suite fails. What do I do?

A: Read the README section on "Common Compilation Problems."

Q: Tripwire reports that my database version is out of date. What should I do?

A: The database format used by Tripwire v1.2 changed. You need to rebuild the database with Tripwire v1.2; see the README file for details. If you are monitoring filenames in which there are spaces, you will need to rebuild the database with Tripwire v1.3.1.

Q: When running Tripwire in Integrity Checking mode, Tripwire fails when it tries to find a file with a name consisting of dozens/hundreds/thousands of '/'s. What went wrong?

A: Your setting for the #define DIRENT value in your conf-<os>.h file is probably incorrect. Trying switching the setting and see if the problem goes away. (i.e., switch #define to #undef, or vice versa.)

Q: I have /tmp in my tw.config file, but none of the files in the directory are being read by Tripwire. What's going on?

A: Check to see that your /tmp directory isn't a symbolic link to another file system. When recursing down into directories, Tripwire never traverses symbolic links or enters another file system.

Q: Is there any way I can get Tripwire to print out the names of the files as they are being scanned? I want to know which files Tripwire is spending all of its time crunching.

A: Use 'tripwire -v'.

Q: I try to initialize the database by typing 'tripwire -initialize' but I can't find the binary. Where is the tripwire executable?

A: ./src/tripwire is where the binary is built. 'make install' will install in the $(DESTDIR) of your choice, as defined in the top-level Makefile.

Q: **I have the following line in my tw.config file to do host specific actions. Why doesn't it work?**

```
@@ifhost chapel || chekov || chewie || data || guinan
        ....
@@endif
```

A: You must put the hostnames as returned by 'hostname' or 'uname' (depending on whether you're running a BSD or SYSV derived OS). So, the correct form would be:

```
@@ifhost chapel.enterprise.fed || chekov.enterprise.fed ...
```

The Tripwire preprocessor tries its best to figure out if you have used misformed hostnames.

Q. **As part of my operational security plan for my exported NFS partitions, I want to run "fsirand" regularly. Unfortunately, if I do this, Tripwire will complain that every file has changed (because the i-node numbers will change). I don't want to rebuild the entire system database each time. What can I do?**

A. We have included a Perl script in the distribution that will go through a Tripwire db file (the output database) and update the i-node fields, while leaving everything else the same. To use it, you need to modify the first line to point to your Perl interpreter (if you don't have Perl, you'll need to write your own program in C or get Perl from an ftp site).

The Perl script is ./contrib/twdb_newinode.pl.

The next time you are in single-user mode running fsirand, run this script with the db as input. For example,

```
cd /usr/local/adm/tcheck/databases
./twdb_newinode.pl tw.db_myhost
```

It will automatically create a backup version of the file for you as a "just in case."

Afterwards, be certain to set the disk with the database back to read-only!!
Also, store the Perl script in the same secure place as the Tripwire program.

Last updated: April 30th, 1999. Mail comments to: support@tripwiresecurity.com

PRODUCT RELEASE AND DEVELOPMENT HISTORY

Release 1.3.1

Original Release: April 30, 1999
Current Build: April 29, 1999

Version 1.3.1 reflects a company name change, to Tripwire Security Systems Inc. A database parser error has been fixed, a temp file that was left behind is now deleted, the code comments have been rewritten for clarity, and command lines now have POSIX-style alternatives. Help commands now display command line help information; inode help has its own command-line option.

Release 1.3

Original Release: Fri Jul 10, 1998
Current Build: July 21, 1998

Version 1.3 integrates bug fixes for all known bugs, as well as solving the problem database consistency errors that could creep in when doing database updates. This fix obviates the need for "twdb_check.pl" that was included with earlier releases.

In addition, several Tripwire reports have been changed slightly, reflecting the feedback that sent over the last year: integrity checking reports are more succinct, and much more quiet when there are no notable changes.

- Fixed database entry consistency bug.
- Fixed database filename construction routine.
- Made "loosedir" reporting the default. Makes superfluous directory changes go away.
- Made reports more succinct, and much more quiet when there's nothing worth reporting.
- Updated manual.
- Added Tripwire Security Systems, Inc., oration banner to startup.
- Eliminated RCS banners for any changed files (RCS no longer being the source control system for our source archives).
- Pulled out user manual (.doc and .pdf files) out of Tripwire package. These items will be distributed separately.
- Removed twdb_check.pl from Tripwire package.
- Updated README, README.FIRST, and COAST.info files.
- Aux directory is now util, to accommodate DOS FAT filename restrictions.

Release Version 1.25

Original Release: April 10, 1998
Current Build: April 18, 1998

Version 1.25 for Red Hat Linux adds RPM support, as well as several other changes. Among the changes are:

- Added RPM support for Red Hat Linux and automated build and installation for Red Hat Linux systems.
- Changed default select-masks to use only MD5, instead of MD5 and Snefru.
- Added tw_isalnum() to workaround Linux libc bug.
- Updated tw.conf.linux file.
- Made filename_escape() eight-bit clean.
- twdb_check.pl now works with Perl 5.
- Changed test scripts to sleep 2 before doing Tripwire checks.
- Changed aux to util in src/Makefile.

Release Version 1.2

Original Release: July 15, 1994
Current Build: March 31, 1998

Version 1.2 adds several new features, as well as fixing reported bugs. Among the changes are:

- Signature checking for symbolic link contents has been added.
- Tripwire now correctly runs on Alpha AXPs, and other machines with "long" types that are not 32 bits wide.
- The Haval digital hash routine has been added as the eighth signature routine (faster than MD5, and purportedly more secure).
- The SHA signature routine has been changed to conform to the recent fix introduced in its FIPS definition by NIST/NSA to correct an unspecified weakness.
- The database format changes slightly to correct a boundary condition error. Because database entry numbers change, because the SHA signatures change, and because of Haval, old Tripwire databases must be reinitialized.
- Handling specified configuration and database files (and file descriptors) has been fixed to better accommodate pipes.
- Full support for flex added.
- Signature checking is now considerably faster through the use of the studio library for file I/O.
- A Perl script has been added to update Tripwire databases where all inode numbers were changed by "fsirand" (NFS sites only);
See FAQ.
- Another fix to make database updates more predictable.
- All reported bugs have been fixed in this revision.
- A new README section describes some documented attacks on systems running Tripwire.
- Many small changes have been made to the documentation to correct and update information.

NOTE: The script 'twdb_check.pl' (written in Perl) has been added to the distribution. It checks database consistency after updates of the tw.config file. This functionality will be put into the Tripwire program in the next release. Run this script after Tripwire database updates to ensure that database entry numbers are consistent with the tw.config file. See the README file for details (section 3.5.2).

Numerous other people played a crucial role in shaping this release:

Paul Szabo is responsible for a number of fixes and cleaning up the way filenames are handled internally, leading to a much simpler treatment to filenames with escaped characters in Tripwire.

Paul Hilchey is responsible for rewriting the list routines, fixing a number of bugs and replacing a hideous implementation with one that is elegant and succinct.

Asokan is responsible for going through all the Tripwire code to remove assumptions about the size of certain types (e.g., "long" is not 32 bits on the Alpha).

Casper H.S. Dik pointed out how some signature routines used very small reads, leading to suboptimal performance. He offered a simple fix through the use of the studio library for file I/O.

Cal Page modified the lex and yacc files to accommodate newer versions of GNU flex and bison, which continue to diverge from the traditional tools.

Tom Orban spent many an afternoon on the phone with me, guiding me step-by-step to find elusive database update bugs. Among other things, Tom Orban and Terry Kennedy helped track down the problems that led to the addition of the twdb_check.pl script.

Keith Rickert and Eugene Zaustinsky painstakingly pointed out distribution errors. Keith Rickert and Greg Black helped us with the last batch of fixes that shortly preceded this release. We appreciate their help.

Thanks go to those people who helped test Tripwire:

> Eric Berg, Eric M. Boehm, Lothar Butsch, John Crosswhite, Jason Downs, Peter Evans, Jon Freivald, Kevin Johnson, Lothar Kaul, Terry Kennedy, Chris Kern, Paul Madden, Fred Marchand, Mitchell Marks, Jim Moreno, Tom Orban, Lorraine Padour, Calvin Page, Tom Painter, Roger Peyton, Peter Phillips, Keith W. Rickert, Jim Roche, D Seddon, Paul Szabo, Gene C Van Nostern, John Wiegley, Robert Wilhite, Alain Williams, Eugene Zaustinsky.

Release Version 1.1

Version 1.1 considerably upgrades the functionality of Tripwire. All known bugs have been fixed, and many selected features have been added at the request of Tripwire users.

Among the major changes are:

- rewrite of the "-update" command.
- addition of an "-interactive" command that prompts the user whether a changed file's database entry should be updated.
- addition of a "-loosedir" command for quieter Tripwire runs.
- support for monotonically growing files in tw.config.
- addition of comprehensive test suite to test Tripwire functionalities.

- hooks for external services (i.e., compression, encryption, networking) through "-cfd" and "-dfd" options.
- addition of the new NIST SHA/SHS signature algorithm.
- corrections and changes in the MD2, MD4, MD5, CRC32, and Snefru signature routines.
- addition of a more rigorous signature test suite.
- more error checking in tw.config @@directives.
- siggen replaces sigfetch.
- addition of a tw.config file for Solaris v2.2 (SVR4).
- change of base-64 alphabet to conform to standards.
- preprocessor macro fixes.

New Tripwire database format:

The Tripwire database format has changed since v1.0, using a different base-64 alphabet and encoding scheme. Use the twconvert program to convert v1.0 databases to v1.1 databases (located in the ./src directory).

If you have been using an older version of Tripwire, you will need to use twconvert convert your databases to the new format.

Updating the Tripwire database:

There has been a major rewrite/rethink of the “tripwire -update” command, as well as the addition of a “tripwire -interactive” command which allows the user to interactively select which database entries should be updated. No vestiges of the “-add” or “-delete” command remain, since the “-update” command now automatically deletes and adds files.

However, the preferred way of keeping Tripwire databases in sync with the filesystems is using the “-interactive” command. A Tripwire session using Interactive mode might look like:

```
6:25am (flounder) tw/src 1006 %% tripwire -interactive
    ### Phase 1:   Reading configuration file
    ### Phase 2:   Generating file list
    ### Phase 3:   Creating file information database
    ### Phase 4:   Searching for inconsistencies
    ###
    ###                     Total files scanned:            49
    ###                           Files added:              0
    ###                           Files deleted:            0
    ###                           Files changed:            49
    ###
    ###                     After applying rules:
    ###                           Changes discarded:        47
    ###                           Changes remaining:        2
    ###
    changed: drwx------ genek        1024 May  3 06:25:37 1993
/homes/research/tw/src
    changed: -rw------- genek        7978 May  3 06:24:19 1993 /homes/
/research/tw/src/databases/tw.db_flounder.Eng.Sun.COM.old
    ### Phase 5:   Generating observed/expected pairs for changed files
    ###
    ### Attr        Observed (what it is)         Expected (what it should be)
    ### =========== ============================= =============================
    /homes/research/tw/src
          st_mtime: Mon May  3 06:25:37 1993      Mon May  3 06:11:39 1993
```

```
      st_ctime: Mon May  3 06:25:37 1993      Mon May  3 06:11:39 1993
---> File: '/homes/research/tw/src'
---> Update entry?  [YN(y)nh?] y
```

```
    ### Updating database...
    ###
    ### Phase 1:   Reading configuration file
    ### Phase 2:   Generating file list
    ### Phase 3:   Updating file information database
    ###
    ### Warning:   Old database file will be moved to
`tw.db_flounder.Eng.Sun.COM.old'
    ###            in ./databases.
    ###
    6:25am (flounder) tw/src 1007 %%
```

Tripwire prompts the user whether the database entry of the current file should be updated to match the current file information. Pressing either ‘y’ or ‘n’ either updates the current file or skips to the next file. Pressing ‘Y’ or ‘N’ applies your answer to the entire entry. (I.e., if /etc is changed, typing ‘Y’ will not only update /etc, but it will also files update all the files in /etc.)

Tripwire exit codes:

Tripwire exit status can be interpreted by the following mask:

1:	run-time error. aborted.
2:	files added
4:	files deleted
8:	files changed

For example, if Tripwire exits with status code 10, then files were found added and changed. (i.e., 8 + 2 = 10.)

Tripwire quiet option:

When run with -q option, Tripwire really is quiet, printing only one-line reports for each added, deleted, or changed file. The output is more suitable for parsing with *awk* or *perl*.

Monotonically growing files:

The “>” template is now supported in the tw.config files. This template allows files to grow without being reported. However, if the file is deleted or is smaller than the size recorded in the database, it is reported as changed.

Loose directory checking:

This option was prompted by complaints that Tripwire in Integrity Checking and Interactive mode unnecessarily complains about directories whose nlink, ctime, mtime, or size have changed.

When Tripwire is run with the “-loosedir” option, directories automatically have these attributes included in their ignore-mask, thus quieting these complaints.

Note that this is option is not enabled by default, making normal Tripwire behavior no different than previous releases. However, running with this option enabled considerably decreases “noise” in Tripwire reports.

(Ideally, this “loose directory checking” should be offered on a per-file basis in the tw.config file. However, adding another field to the tw.config file was too extensive a change to be considered for this release. A later release of Tripwire may rectify this.)

Hooks for external services:

Tripwire now supports the “-cfd” and “-dfd” option that allows the user to specify an open file descriptor for reading the configuration file and database file, respectively. Using these options, an external program can feed Tripwire both input files through open file descriptors. This external program could supply services not provided though Tripwire, such as encryption, data compression, or a centralized network server.

This program might do the following: Open the database and configuration files, process or decode (i.e., uncompress the file), and then write out the regularly formatted file to a temporary file. Open file descriptors to these files are then passed to Tripwire by command-line arguments though excel().

An example of using a shell script to compress and encrypt your files is given in ./contrib/zcatcrypt. It is a four line Bourne shell script that encrypts and compresses the database and configuration files.

Change in tw.config preprocessor:

The tw.config preprocessor has been changed to allow the proper expansion of @@variables in filenames. The following use of @@define now works as expected:

```
@@define DOMAIN_NAME    my_main_nis_domain
/var/yp/@@DOMAIN_NAME   L
@@DOMAIN_NAME/FOO L
```

(This is the third attempt at getting this working correctly. We finally fixed this by moving the macro expansion routines into the lexical analyzer.)

Expanded test suite:

The Tripwire test suite now includes runs a more standard signature test suite. This was prompted by discovery of several implementation errors in the MD2, MD4, and MD5 signature routines that were introduced right before the official release of Tripwire. (Thanks Eugene Zaustinsky.)

Two more test suites have been added. One iterates through all the Tripwire reporting functionalities, and exercises all the database update cases. The other test suite checks for proper Tripwire preprocessor macro expansions.

CRC32 changes:

Furthermore, the CRC32 signature routine is now POSIX 1003.2 compliant. (Thanks Dan Bernstein.)

"siggen" replaces "sigfetch":

As a tester noted, “sigfetch” was a misnomer since nothing was actually being fetched. Consequently, it was easy to (incorrectly) conclude that “sigfetch” retrieved signatures from the database.

The “siggen” command is the current incarnation of “sigfetch”. The manual pages reflect this change.

Source code cleanup:

The authors went through the sources, doing generic cleanups aid in code comprehension.

Bug fixes:

This release fixes all known bugs. The TODO list, however, gives a wish list of features that may be included in future releases.

List of thanks:

Special thanks go to the testers of disappearing v1.0.3. Reports of critical bug fixes go to (in no special order): E. Clinton Arbaugh, Pat Macdonald, Eric Demerling, John Rouillard, Bob Cunningham, and Neil Todd.

Sam Gassel, Edward DeHart, Drew Gonczi, Rik Farrow, Jim Napier, Drew Jolliffe, John Rouillard, Alain Brossard, Eric Bergren, Patrick Sullivan, Nora Hermida, Juergen Schmidt, Debbie Pomerance, Michael Hines, Tim Ramsey, Georges Tomazi, Mitchell Marks, Philip Cox, Kevin Dupre’, Chris Kern, and Eugene Zaustinsky helped in getting the Tripwire v1.1 release in shape for our December 1993 release.