THE SOLARIS™ FINGERPRINT DATABASE
A SECURITY TOOL FOR SOLARIS OPERATING ENVIRONMENT FILES

Vasanthan Dasan, Sun Services Product Management
Alex Noordergraaf, Common Software Features Engineering
Lou Ordorica, Global eServices Engineering
Glenn Brunette, Client Solutions

Sun BluePrints™ OnLine — March 2006

SUN BLUEPRINTS [illegible] APPROVED

Part No 816-1148-11
Revision 2.0, 3/14/06
Edition: March 2006

Table of Contents

The Solaris™ Fingerprint Database

This Sun BluePrints™ Online article describes the Solaris Fingerprint Database (sfpDB), a security tool that enables users to verify the integrity of files distributed with the Solaris Operating Environment.
By validating that these files have not been modified, administrators can determine whether their systems have—or have not—been hacked and had trojaned malicious replacements for system files installed.

This article consists of the following sections:

Note – This is an updated version of the original Sun BluePrints publication, “The Solaris Fingerprint Database - A Security Tool for Solaris Operating Environment Files” published in May 2001 by Vasanthan Dasan (Support Services, Strategy Group), Alex Noordergraaf (Enterprise Engineering), and Lou Ordorica (Global eServices Engineering). This document has been updated by Glenn Brunette to support the Solaris 10 Operating System and includes numerous other additions, clarifications, and references.

Overview

Is the operating system that you are using genuine? Has it or have any of the tools built upon it been modified without your knowledge, accidentally or maliciously? How could you tell? It has always been a difficult task verifying whether operating system files—such as system executables, configuration files, libraries, and kernel modules—have been modified. Some organizations thankfully do attempt to validate the integrity of their file system objects. However, in most cases, the validity of such objects often goes unquestioned and unchecked. Without a well-defined and executed process for checking the integrity of these files, organizations are often completely unaware that the integrity of their operating system, applications, and even their data may have been compromised.

Attackers with sufficient access may be able to install or replace key file system objects with versions of their own choosing, leaving behind “trojan horses” (or simply “trojans”). Often these trojans are used to gain additional forms of access (for example, capturing other users’ keystrokes or passwords) or to permit the attackers to regain their privileges upon their return. To combat this threat, security tools—both commercial and open-source—have been developed that help organizations check the integrity of objects stored on their file systems. These tools usually collect attributes (such as owner, group, permissions, inode, access control lists, and so on) along with one or more cryptographic checksums (or “fingerprints”) for some set of file system objects. Each time the tool is used, a snapshot is created consisting of the current state of the collected objects. Each new snapshot can then be compared with previous ones or perhaps a standard baseline, if one exists, in order to detect changes to either attributes or fingerprints.

While such tools have been publicly available for many years, they are often not used consistently or pervasively throughout an organization, if they are used at all. To help organizations wanting to implement file integrity validation processes, starting in the Solaris 10 Operating System (Solaris 10 OS), organizations can use the integrated Basic Auditing and Reporting Tool (BART) to create and compare file system snapshots. For more information on BART, see the references at the end of this article.

A challenge that is common to all of the file integrity validation tools, however, is that they depend on the creation of a secure baseline. Because, fundamentally, such tools simply compare two snapshots (for example, a “baseline” and “current” snapshot), it is critical that the baseline be founded upon genuine file system objects. If any object in the baseline had been modified prior to its attributes and fingerprints being collected, there would be no way to detect that the object had been compromised. To cope with this risk, some organizations often create their baselines during or immediately after system or software installation or from a pristine golden image that may be reused and provisioned onto tens or hundreds of systems.

File integrity validation tools, just like other software products, must be properly used and maintained in order to remain effective. The tools themselves, along with their configuration files and snapshots, must be protected. The baseline snapshot used by these tools will necessarily change over time, as software is used and updated, and as patches are applied. It is critical, therefore, that any detected changes be carefully evaluated and reconciled before updating the baseline with new values so that future snapshots can be compared against values that are known to be correct and current. Unfortunately, in many cases, the baseline is not properly maintained. As a result, unauthorized changes to file system objects may go undetected as they hide among the mass of authorized changes resulting from normal system use and administration activities (for example, patching, upgrades, and so on).

Further, a race condition exists, whereby an object installed by a patch could be overwritten by an attacker before the administrator had the opportunity to update the baseline snapshot with the new values. As a result, the attacker’s change might not be detected because the object was expected to be changed by the patch. Depending on the processes used by the organization and how well they are followed, this exposure could exist for a long period of time.

The Solaris Fingerprint Database (sfpDB) is a free SunSolve Online^{SM} service that enables organizations to verify the integrity of files distributed with the Solaris OS. Specifically, this means that organizations can, at any time, query the sfpDB to determine whether an operating system file did indeed come from Sun.

This is a powerful tool for organizations wanting to validate the integrity of their operating systems, baseline snapshots, and patches—or even to assist during digital forensic investigations in which the integrity of objects might be called into question.

The Solaris Fingerprint Database is itself a collection of file fingerprints that have been created based on Solaris OS media kits, unbundled software, and patches. These fingerprints are created as part of the release process in order to ensure that they are up to date and reflect actual shipping versions of files provided by Sun. By submitting fingerprints to the Solaris Fingerprint Database service, organizations can then determine which, if any, of the submitted fingerprints belongs to an actual file supplied by Sun.

How Does the sfpDB Work?

As noted above, the Solaris Fingerprint Database is simply a collection of file fingerprints. These fingerprints are generated using the MD5 Message Digest Algorithm, which is defined in RFC 1321:

> The algorithm takes as input a message of arbitrary length and produces as output a 128-bit "fingerprint" or "message digest" of the input. It is conjectured that it is computationally infeasible to produce two messages having the same message digest, or to produce any message having a given prespecified target message digest.

By using this algorithm, it is expected that no two different files will share the same fingerprint. Similarly, it is virtually impossible to modify a file in such as way as to retain its original MD5 fingerprint. It should be noted that, in September 2004, it was announced that the MD5 algorithm was “broken.” That is, researchers developed a method to produce two pre-images (for example, files) with the same MD5 hash value. While this is an important announcement, this breakthrough does not impact the validity of the Solaris Fingerprint Database because it is still not feasible (as of the publication of this article) to produce a pre-image (such as a text or binary file) that matches a given hash value. That is, it is not feasible that a file could be replaced with another, different file and still retain its original MD5 hash value. For more information, see http://www.schneier.com/crypto-gram-0409.html#3.

The algorithms used by both the `sum(1)` and `cksum(1)` Solaris OS commands, however, are much easier to circumvent and re-create the hash of the unmodified file, which is why the MD5 algorithm was used instead.

Organizations wanting to use the Solaris Fingerprint Database service need only to calculate MD5 fingerprints for the files that they want to check, and then submit them to the database service for processing. The sfpDB service will evaluate the submitted fingerprints and determine if there exists a match for any of the objects.

For each fingerprint match, the following information will be provided:

Field	Description
`canonical-path`	Absolute path name of the file delivered by Sun matching the supplied fingerprint. This field is often examined in order to determine whether a valid Solaris OS program has been renamed or moved to a new location.
`package`	Solaris OS package name associated with this file. With this information, the Solaris OS packaging tools can be used to gather additional information. Further, the package name can be compared with those actually installed on the system to determine whether an unauthorized package had been installed, or a file (from that package) has been added to the system.
`version`	Version of the Solaris OS package associated with this file. This information can be used to determine whether the file was supplied as part of the initial Solaris OS distribution, or whether it was made available in a patch. This value should match the one returned by the following command: `pkgparam <package_name> VERSION`. This information is useful in detecting downgrade attacks in which a program is replaced with an older, potentially vulnerable (but otherwise valid) version of itself.
`architecture`	Hardware architecture associated with the package that delivered the file.
`source`	Original product that delivered the file. Just as with the `version` information, this value can be used to identify discrepancies between what is and what should be installed on the system. For example, this could be used to flag a Solaris 2.5.1 binary that had been installed on a Solaris 10 system.

Note that a single file fingerprint may result in several matches. This is most often the case for text files (for example, configuration files and shell scripts) that do not depend on the underlying operating system version or hardware platform.

Note – Internet connectivity does not need to be available from all systems to use the sfpDB, as the files containing the MD5 hashes can be moved to an Internet-connected machine and verified either manually or through the use of automated tools, such as the sfpDB Companion described in “Solaris Fingerprint Database Companion (sfpCompanion)” on page 16.

sfpDB Scope

The goal of the sfpDB is to provide a comprehensive collection of fingerprints for Solaris OS software. To this end, the sfpDB is updated daily and presently contains close to 3.5 million fingerprints for files used in the Solaris OS, Solaris OS patches, and unbundled products.

For example, the Solaris Fingerprint Database contains fingerprints associated with the following operating systems:

- Solaris on SPARC versions 2.0, 2.1, 2.2, 2.3, 2.4, 2.5, 2.5.1, 2.6, 7, 8, 9 and 10
- Solaris on x86 versions 2.1, 2.4, 2.5, 2.5.1, 2.6, 7, 8, 9
- Solaris on x86/x64 version 10
- Solaris on PPC version 2.5.1
- Trusted Solaris on SPARC versions 2.5, 2.5.1, 7, and 8
- Trusted Solaris on x86 versions 7 and 8
- Software on most of the additional CDs bundled with Solaris 2.6 and newer.

The list of what is contained continues to change as new products, updates, and patches are generated. For more information on what is included in the Solaris Fingerprint Database, see:
`http://sunsolve.sun.com/pub-cgi/fileFingerprints.pl?mode=coverage`

Limitations

Currently, foreign language versions of the Solaris OS and many encryption products are not included in the Solaris Fingerprint Database. To suggest a product be added to sfpDB, please send e-mail to: `fingerprints@sun.com`.

Further, the Solaris Fingerprint Database is used to validate only the content of files shipped by Sun for integrity. The Solaris Fingerprint Database cannot be used to detect changes in file ownership, group membership, permissions, access control lists, modification time, or other such attributes. The sfpDB—used in concert with tools such as BART, discussed earlier—can, however, provide a more comprehensive approach to detecting changes to files on systems.

Comparison with elfsign(1)

Starting in the Solaris 10 OS, the core operating system ELF binary objects (commands, libraries, kernel modules, etc.) have been cryptographically signed. In addition, the `elfsign(1)` program is available and can be used to query whether an object's signature is valid, and therefore whether the content of the binary is genuine. This technique provides the same level of assurance as the Solaris Fingerprint method described earlier. The Solaris Fingerprint Database goes beyond this signature verification capability, however, by allowing any operating system file—text or binary—to be evaluated. In addition, the sfpDB also contains additional attributes, such as the canonical path noted above, that can be used to determine whether files have been renamed or replaced with an older or newer version.

MD5 Download and Installation

This section applies only to the Solaris OS version 9 and older releases. Starting with the Solaris 10 OS, similar functionality is already available in the form of the `digest(1)` command.

For older versions of the operating system, it is necessary to download and install MD5 software that will be used to generate the fingerprints that are submitted to the sfpDB. The goal of this section is to illustrate where and how such a program can be downloaded and installed.

To install the MD5 program (SPARC™ and Intel architecture):

1. Download the MD5 binaries from the following URL:
   `http://sunsolve.Sun.COM/md5/md5.tar.Z`
   The MD5 programs are distributed in compressed tar file format.
2. Save the file to a directory that is not writable by other users.
3. Unpack the archive:

```
# zcat md5.tar.Z | (cd /opt; tar xvf -)
```

The archive contents are extracted into `/opt` into a newly created directory called `md5`. The programs for the SPARC and Intel Architecture hardware platforms are placed in this directory. Simply replace the `/opt` directory with one of your choosing if you do not want to store the programs in this location.

4. The file permissions on the extracted directory and files must be modified before they can be executed. The following command will assign read and execute permissions to the `md5` programs for user, group, and world. These settings are used to align with the permissions used by the `digest` command in Solaris 10.

```
# chmod -R 555 /opt/md5
# ls -l
total 94
-r-xr-xr-x 1 21782    320       23892 Apr  5  2000 md5-sparc
-r-xr-xr-x 1 21782    320       23452 Apr  5  2000 md5-x86
```

5. The owner and group of the extracted directory and files must also be modified to correspond to a system-defined user and group ID. Due to the sensitivity of the operations being performed by the `md5` programs, they should be owned by the `root` user and the `bin` group. The following example demonstrates performing this on the `md5` programs. Again, these settings are used to align with the owner and group used by the `digest` command in Solaris 10.

```
# chown -R root:bin /opt/md5
# ls -l
total 94
-r-xr-xr-x 1 root     bin 23892 Apr  5  2000 md5-sparc
-r-xr-xr-x 1 root     bin 23452 Apr  5  2000 md5-x86
```

Note – The Solaris Fingerprint Database can be used to verify the integrity of the executables included in the package itself. In addition, the MD5 fingerprints for each of these files can be found in the Notes section at `http://sunsolve.sun.com/pub-cgi/fileFingerprints.pl`.

Creating an MD5 Fingerprint

As noted in the previous section, the Solaris 10 OS contains a new program that can be used to generate MD5 fingerprints. For the remainder of this article, the Solaris 10 `digest` command will be used in the examples. If you are using an older version of the Solaris OS, then you should use the MD5 program that was downloaded and installed in the previous step.

The following example uses the `digest` program to create an MD5 fingerprint:

```
# digest -v -a md5 /usr/bin/su
md5 (/usr/bin/su) = bab18f089705c1628ccdc177b802d571
```

The `digest` program can also be used to create multiple MD5 fingerprints, as shown in the following example:

```
# digest -v -a md5 /usr/bin/su /usr/bin/ls
md5 (/usr/bin/su) = bab18f089705c1628ccdc177b802d571
md5 (/usr/bin/ls) = b526348afd2d57610dd3635e46602d2a
```

Note – The two previous examples were performed on a freshly installed Solaris 10 03/05 OS system. No patches had been installed. Do not rely on the output generated above as correct for every system. Use the following procedure, described in “Testing an MD5 Fingerprint” on page 8.

Use the `digest` program with the `find(1)` command to generate MD5 fingerprints for files that have recently changed. The next example creates MD5 fingerprints for files stored in the `/usr/bin` directory modified in the last day:

```
#find /usr/bin -type f -mtime -1 -print \
| xargs -n 1000 digest -v -a md5 > ./md5s.txt
```

The results contained in the `./md5s.txt` file can be easily reviewed and copied into the Solaris Fingerprint Database web form.

Note – A maximum of 256 entries can be submitted into the web form at one time. If you need to submit more than 256 entries, you should consider using the Solaris Fingerprint Database Companion tool, described in “Solaris Fingerprint Database Companion (sfpCompanion)” on page 16, to automate the submission process.

Starting in the Solaris 10 OS, all operating system ELF binaries have been cryptographically signed by Sun. The signatures on these files can be validated using the `elfsign(1)` command.

```
# elfsign verify -e /usr/bin/su
elfsign: verification of /usr/bin/su passed.
```

Using a combination of commands, a list of fingerprints can be created for files that do not have a valid digital signature. This situation might occur because the file has not been signed (for example, non-ELF signed objects) or because a valid file has been modified or replaced.

The following example illustrates a series of commands that will generate a list of MD5 fingerprints for files under the `/usr/bin` directory for which a valid digital signature was not found:

```
# find /usr/bin -type f -print | while read file; do
>    elfsign verify -e ${file} > /dev/null 2>&1
>    if [ $? != 0 ]; then
>       digest -v -a md5 ${file}
>    fi
> done
md5 (/usr/bin/alias) = 2fb1c3bc52d8dcc697ed739dc199887a
md5 (/usr/bin/amt) = b8ec39f1d0450712f6ebff73f1884b95
md5 (/usr/bin/arch) = ddd3c131e97692429a8de0fb9db86667
md5 (/usr/bin/basename) = e7bc3ee2a755793a2365dc5a2b4a6375
md5 (/usr/bin/clear) = 73592eaed5d9b53fafb3d04375956ef9
md5 (/usr/bin/dirname) = d63199159ea36ae8f0a8ac2e89c6e1c4
md5 (/usr/bin/hostname) = 8a72719442f8f854a596f8392e491c69
[...]
```

For each of the above commands in which an MD5 fingerprint was generated, it should be noted that the reason why they failed their digital signature check was simply because they were not signed ELF binaries. Each of the above commands corresponds to a shell script.

Note – In the Solaris 10 OS 03/05 release, the `elfsign(1)` command was delivered in the `SUNWtoo` package, which is not installed in the Reduced Networking (`SUNWCrnet`) or Core (`SUNWCreq`) software installation clusters. Be sure to install the `SUNWtoo` package if you want this functionality and are using either the Reduced Networking or the Core software installation cluster.

Testing an MD5 Fingerprint

Once the MD5 fingerprints have been computed, they can be submitted to the Solaris Fingerprint Database to determine whether they correspond to values associated with valid Sun operating system files. To test the collected fingerprints:

1. Visit the Solaris Fingerprint Database page at:

   `http://sunsolve.Sun.COM/pub-cgi/fileFingerprints.pl`

   After connecting to this URL, the Solaris Fingerprint web form will be displayed. Scroll to the bottom of the page to find the text window where the collected fingerprints can be entered.
2. Type or copy and paste one or more MD5 fingerprints into the web form. For example, to verify the `su` command fingerprint, generated in the example above, simply paste the following value into the web form:

```
bab18f089705c1628ccdc177b802d571
```

3. Click the Submit button to view the results.

Using our example, the following result should be returned:

```
Results of Last Search

bab18f089705c1628ccdc177b802d571 -   - 1 match(es)

        * canonical-path: /usr/bin/su
        * package: SUNWcsu
        * version: 11.10.0,REV=2005.01.21.15.53
        * architecture: sparc
        * source: Solaris 10/SPARC
```

In addition to submitting the actual MD5 file fingerprint, the sfpDB also accepts submissions in alternate formats. For more information on valid formats, refer to the Solaris Fingerprint Database query submission page. Depending on the format used, the results of the search might include additional information.
For example, consider a submission in which the input takes the form of a canonical MD5 output format string:

```
md5 (/usr/bin/su) = bab18f089705c1628ccdc177b802d571
```

In this case, the output generated by the Solaris Fingerprint Database will include the name of the file system object, in addition to the fingerprint on the line that indicates the number of matches found in the database.

```
 bab18f089705c1628ccdc177b802d571 - (/usr/bin/su) - 1
match(es)

        * canonical-path: /usr/bin/su
        * package: SUNWcsu
        * version: 11.10.0,REV=2005.01.21.15.53
        * architecture: sparc
        * source: Solaris 10/SPARC
```

Analyzing sfpDB Results

In the above example, the Solaris Fingerprint Database query properly identified the `/usr/bin/su` program as the object to which the fingerprint belonged. For purposes of illustration, several additional outcomes are discussed in this section, including multiple database matches, canonical path mismatches, matches indicating an upgrade or downgrade attack, and of course the failure in which a match was not found.

Multiple Matches

As mentioned previously, a single fingerprint might match multiple objects in the sfpDB. This is often the case when the object being evaluated corresponds to a text file (for example, shell script, configuration file, and so on) that does not depend on the underlying hardware architecture or operating system used. This section uses the `/usr/bin/basename` command as an example.

The MD5 fingerprint of the `/usr/bin/basename` command is:

```
# digest -v -a md5 /usr/bin/basename
md5 (/usr/bin/basename) = e7bc3ee2a755793a2365dc5a2b4a6375
```

When submitted to the sfpDB, the following results (abbreviated) are returned:

```
 e7bc3ee2a755793a2365dc5a2b4a6375 -   - 33 match(es)

        * canonical-path: /usr/bin/basename
        * package: SUNWcsu
        * version: 11.10.0,REV=2005.01.21.16.34
        * architecture: i386
        * source: Solaris 10/x86

        * canonical-path: /usr/bin/basename
        * package: SUNWcsu
        * version: 11.5.1,REV=94.07.22.14.24
        * architecture: i386
        * source: Solaris 2.4/x86

        * canonical-path: /usr/bin/basename
        * package: SUNWcsu
        * version: 11.5.1,REV=95.10.30.17.24
        * architecture: i386
        * source: Solaris 2.5/x86
[...]
```

As you can see from the returned results, there were 33 matches for this single fingerprint value, with matches coming from a variety of Solaris OS versions, including Solaris 2.4, Solaris 2.5, and Solaris 10. The actual results even returned matches for several Trusted Solaris OS versions and matches for both the x86 and SPARC platforms.

To verify that a file is indeed a text file or shell script, simply use the `file(1)` command:

```
# file /usr/bin/basename
/usr/bin/basename:      executable /usr/bin/sh script
```

Canonical Path Mismatches

When reviewing the output of an sfpDB query, it is important to look for cases in which the actual file name of a submitted object differs from the canonical path identified by the sfpDB. This can be an indication that a file has been copied or renamed. Consider the following example, where the `/sbin/su.static` program was copied and renamed as `/var/tmp/myprog`.

```
7fc315a1d00be2077cdad6d7271fa24c - (/var/tmp/myprog) - 1
match(es)

        * canonical-path: /sbin/su.static
        * package: SUNWcsr
        * version: 11.10.0,REV=2005.01.21.15.53
        * architecture: sparc
        * source: Solaris 10/SPARC
```

A canonical path mismatch is not always a problem, however. This situation can arise in cases where a symbolic link was tested rather than the target of the link itself. In this case, both the link and the link target resolve to the same file, and therefore will share the same fingerprint.

```
# digest -v -a md5 /bin/ls /usr/bin/ls
md5 (/bin/ls) = b526348afd2d57610dd3635e46602d2a
md5 (/usr/bin/ls) = b526348afd2d57610dd3635e46602d2a
```

Be careful of both symbolic and hard links. While the sfpDB does not care whether an object is a regular file or a link, be aware that you might be calculating a fingerprint of the target of a link, and therefore you might find multiple binaries with the same fingerprint. For example, in the Solaris 10 OS, the files `/usr/bin/ps` and `/usr/bin/sort` share the same fingerprint. The reason for this becomes clear when the fingerprint is submitted to the sfpDB.

In the following example, note that both of those files are hard linked to the `/usr/lib/isaexec` binary.

```
# digest -a md5 /usr/bin/sort /usr/bin/ps
(/usr/bin/sort) = 76b66713137851c1407fd9ea20785bc0
(/usr/bin/ps) = 76b66713137851c1407fd9ea20785bc0

# ls -1i /usr/bin/sort /usr/bin/ps
        677 /usr/bin/ps
        677 /usr/bin/sort
```

Submitting this fingerprint entry to the sfpDB yields the following response.

```
  76b66713137851c1407fd9ea20785bc0 -   - 1 match(es)

         * canonical-path: /usr/lib/isaexec
         * package: SUNWcsu
         * version: 11.10.0,REV=2005.01.21.15.53
         * architecture: sparc
         * source: Solaris 10/SPARC
```

Similarly, attributes—such as the canonical path for an object—might differ when symbolic links are encountered. Consider the following example in which `/bin/ls` was tested but the sfpDB returned a match whose canonical path was `/usr/bin/ls`.

```
b526348afd2d57610dd3635e46602d2a - (/bin/ls) - 1 match(es)

        * canonical-path: /usr/bin/ls
        * package: SUNWcsu
        * version: 11.10.0,REV=2005.01.21.15.53
        * architecture: sparc
        * source: Solaris 10/SPARC
```

In general, the use of links can be easily detected by considering the following cases.

The submitted file is itself a symbolic link

In this case, the submitted file can be easily tested using a variety of means in order to confirm that it is a symbolic link to the canonical path identified by the sfpDB.

The submitted file is a hard link to the one specified in the canonical path

This case can be easily detected by comparing both the file system on which both objects are stored, as well as the inode of each object. This can be confirmed by inspecting the file system on which both objects are stored, as well as the inode of the objects themselves. If the file system and inodes are respectively the same for both the submitted file and the object identified by the canonical path, then the files are actually one and the same.

```
# df -n /usr/bin/bzcat /usr/bin/bzip2
/                  : ufs
/                  : ufs

# ls -li bzcat bzip2
1876 -r-xr-xr-x   3 root    bin      33380 Jan 22  2005 bzcat
1876 -r-xr-xr-x   3 root    bin      33380 Jan 22  2005 bzip2

# digest -v -a md5 bzcat bzip2
md5 (bzcat) = 0a6bf3a8a3a38ca8b52cf17e7b3de2ab
md5 (bzip2) = 0a6bf3a8a3a38ca8b52cf17e7b3de2ab
```

The path to the submitted file contains a symbolic or hard link to the path of the canonical file

Just as in the previous case, the submitted file is actually the same as the one identified by the sfpDB, although it is recognized by a different name. Consequently, the same test should be used in order to determine whether the submitted fingerprint and the match found in the sfpDB refer to the same file.

```
 # df -n /bin/ls /usr/bin/ls
/                   : ufs
/                   : ufs
# ls -1i /bin/ls /usr/bin/ls
       336 /bin/ls
       336 /usr/bin/ls
# ls -ld /bin
lrwxrwxrwx   1 root     root           9 Sep  4 18:36 /bin ->
./usr/bin
```

In the above example, both `/bin/ls` and `/usr/bin/ls` are stored on the same file system, `/`. These programs also share the same inode number, `336`, which means that they in fact refer to the same file. This is further confirmed when `/bin` is examined and found to simply be a symbolic link to `/usr/bin`.

Upgrade or Downgrade Attacks

The sfpDB does not indicate whether a file is appropriate for the version of the operating system being evaluated. Due to Solaris OS binary compatibility guarantees, it is possible for an attacker to replace a valid Solaris binary or library, for example, with another version from an older, newer, or possibly unpatched version of the operating system.

Therefore, when evaluating file fingerprints, you must be alert to file fingerprint entries that match only versions of the operating system that are not the same as the one installed. If a fingerprint matches a version of the operating system that is either newer or older than the one installed, it is possible that the binary has been replaced, potentially by a version that is vulnerable in some way.

Consider an example in which you are running the Solaris 10 OS but you find that your `/bin/login` matches only a version that existed on the Solaris 2.5.1 OS. You might want to consider that program suspect because your `/bin/login` should at least match the Solaris 10 OS. In an attempt to avoid detection, an attacker could have “downgraded” the `/bin/login` program to a vulnerable version that existed in the Solaris 2.5.1 OS (see Sun Alert 41987). Therefore, while the `/bin/login` file will appear to be genuine (developed by Sun), it is still not appropriate for the release of the operating system being used.

Note – Remember that a given fingerprint can match multiple operating system versions. If this is the case, be certain that at least one of the entries returned matches the version of the operating system being used.

Unmatched Fingerprints

Perhaps the most disconcerting case is one in which a submitted fingerprint does not match a known value in the Solaris Fingerprint Database. Before concern gives way to panic, be sure to verify that the program being evaluated is included in the set of products currently found in the Solaris Fingerprint Database. If the program's fingerprint has not been collected, then a positive match should not be expected. Similarly, if you are running an early access version of a product or if you have applied a T-Patch, it is likely that those fingerprints have not been recorded. Finally, if the file being evaluated is a text or configuration file that may have changed as a result of normal system use, then again, a match should not be expected.

If the file being evaluated is expected to be in the sfpDB and a match was not found for the submitted fingerprint, then the file—and possibly the entire system—should be considered suspect until proven otherwise. Leverage file integrity verification programs, if they had been previously deployed, to determine whether other file attributes have changed recently. For Solaris 10 and newer versions of the operating system, leverage the `elfsign(1)` command to check the digital signatures for binaries. In addition, review change and configuration control logs to see what work might have been recently completed that could point to the cause of this problem. Depending on individual organizational requirements and policies, the system might even be temporarily decommissioned until the cause of the discrepancy can be found and corrected.

Caution: If you suspect that your system has been compromised, we strongly advise that you seek out an expert in digital forensics to assist with your incident response efforts. Remember that a compromised system is very much like a crime scene, and your first order of business should be the preservation of evidence. Even if you are not considering prosecution, if proper precautions are not taken, critical information that may help you understand how the system was breached could be lost. For more information, see the Sun BluePrint article titled "Using Computer Forensics When Investigating System Attacks" at: http://www.sun.com/blueprints/0405/819-2262.pdf.

Real World Results

The sfpDB provides an excellent mechanism for determining whether system binaries, libraries, or other files have been replaced by trojaned, or maliciously modified, versions. To demonstrate the sfpDB performance when encountering actual trojaned Solaris OE binaries, the following experiment was performed. A freshly installed Solaris 10 OS (03/05) system, used in the previous examples, had a Solaris OS rootkit installed. The term *rootkit* is used to describe a set of scripts and executables, packaged together, which allow the user to gain or keep root access on a system. The sfpDB was then used to detect that several valid operating system files had been replaced by trojaned executables.

The rootkit described in the next few examples was originally used in a successful attack on one of the HoneyNet project's systems several years ago. For the purpose of this article, the age of the rootkit is not a factor, as the Solaris Fingerprint Database would perform equally well for newer rootkits that also replace valid operating system files with their own version. For more information on the HoneyNet project, refer to the "Bibliography" on page 18.

Note – The installation of this particular rootkit on the Solaris 10 OS will cause significant harm and will cause many core OS functions to perform incorrectly unless all of the changes implemented are successfully reversed.

Note – Many Solaris OS rootkits are available on the Internet. Most search engines will find several in responding to a simple query for ‘Solaris’ and ‘rootkit.’

This rootkit, called `sun2.rootkit`, replaced several system files.

```
/bin/ls
/usr/bin/ls
/bin/ps
/bin/netstat
/usr/bin/netstat
```

Among other things, the installation script, `setup.sh`, included in the rootkit, performed the following tasks. First, the rootkit backed up some, but not all, of the files it was going to replace.

```
cp /bin/ps ./ps-back
cp /bin/ls ./ls-back
cp /bin/netstat ./netstat-back
```

Then, the rootkit installed its own version of these files with the following commands.

```
cp netstat /bin/netstat
cp netstat /usr/bin/netstat
cp ls /bin/ls
cp ls usr/bin/ls
cp ps /bin/ps
cp ps /usr/bin/ps
```

The fingerprints of these trojaned executables were calculated using the `digest` command.

```
# digest -v -a md5 /bin/netstat /usr/bin/netstat /bin/ls \
/usr/bin/ls /usr/bin/ps /bin/ps
```

The `digest` command generated the following output.

```
MD5 (/bin/netstat) = 2f4ec308b282c5c362e9fbd052b961f6
MD5 (/usr/bin/netstat) = 2f4ec308b282c5c362e9fbd052b961f6
MD5 (/bin/ls) = da2ac2fc4645ff9fb737025f2d184aeb
MD5 (/usr/bin/ls) = da2ac2fc4645ff9fb737025f2d184aeb
MD5 (/usr/bin/ps) = abd478c6597b4df1d565b9568f9e91bf
MD5 (/bin/ps) = abd478c6597b4df1d565b9568f9e91bf
```

Submitting these fingerprints as input to the Solaris Fingerprint Database Companion (described later), the following output was produced.

```
2f4ec308b282c5c362e9fbd052b961f6 - (/bin/netstat) - 0 match(es)
Not found in this database.

da2ac2fc4645ff9fb737025f2d184aeb - (/bin/ls) - 0 match(es)
Not found in this database.

abd478c6597b4df1d565b9568f9e91bf - (/usr/bin/ps) - 0 match(es)
Not found in this database.
```

The Solaris Fingerprint Database correctly identified the trojaned executables as not being part of a Solaris OS distribution or other Sun product.

Additional sfpDB Tools

Sun has developed and made freely available two tools that help to make it easier to collect MD5 fingerprints and submit them to the Solaris Fingerprint Database: the sfpDB Sidekick and sfpDB Companion. This section describes both of these tools. To download these tools from the Sun BluePrints OnLine Tools site, go to http://www.opensolaris.org/os/community/security/projects/sfpdb/.

Solaris Fingerprint Database Sidekick (sfpSidekick)

The sfpDB Sidekick (`sfpSidekick.sh`) program is a small shell script that collects MD5 fingerprints for different collections of files. Sidekick was designed to help simplify the process of collecting and processing fingerprints (using the sfpDB Companion discussed below). In particular, the Sidekick program can be used to collect fingerprints for the following classes of files:

- Files commonly found in rootkits available on the Internet;
- Files with the set-user-id bit set;
- Files with the set-group-id bit set;
- Files with the sticky-bit set;
- Files not currently assigned to a valid user on the system;
- Files not currently assigned to a valid group on the system;
- Files with world writable permissions;
- Files with extended file attributes (Solaris 9 and newer only)
- Files that do not have a valid digital signature (Solaris 10 and newer only)

In addition, the Sidekick program can be configured to collect fingerprints associated with every file on the system (local file systems only).

Solaris Fingerprint Database Companion (sfpCompanion)

The Solaris Fingerprint Database Companion (`sfpCompanion.pl`) program is a Perl script that automates the process of submitting MD5 fingerprints to the sfpDB. Accepting as input a list of MD5 fingerprints (from either a file or standard input), the Companion breaks up the list into manageable chunks (less than the maximum allowable by the sfpDB) and sends them to the sfpDB for processing. The returned results are then collected and parsed as text based on the HTML output. The Companion tool makes it much easier to check large lists of Solaris OS files by automating the submission of the files to the sfpDB web site. The Sidekick program contains an option to automatically call the Companion tool in order to automate both the collection and processing of file fingerprints.

The Solaris Fingerprint Database Companion tool can be used as follows:

```
# digest -v -a md5 /usr/bin/ls | ./sfpC.pl -

 b526348afd2d57610dd3635e46602d2a - (/usr/bin/ls) - 1 match(es)

 canonical-path: /usr/bin/ls
 package: SUNWcsu
 version: 11.10.0,REV=2005.01.21.15.53
 architecture: sparc
 source: Solaris 10/SPARC
```

Similarly, refer to the Sun BluePrint titled “Integrating BART and the Solaris Fingerprint Database in the Solaris 10 Operating System” (see http://www.sun.com/blueprints/0405/819-2260.pdf) for additional ways in which the Solaris Fingerprint Database Companion can be used.

sfpDB Frequently Asked Questions

Why do some of the returned entries contain odd path names?
In the process of gathering MD5 file fingerprint data, it was discovered that many Solaris packages are not properly structured. Some path names may not be decided until installation. For these path names, it is not possible to derive the file name as found installed on the system—some path names are wrong, and some will contain `$SOMEVAR` values to be expanded during installation. For the vast majority of files, however, they should be able to be resolved to a specific path name. In any case, if a file was positively identified, it was shipped by Sun. The pathname does not need to match, although in cases where a path name match is not found, further investigation might be warranted to determine whether the file was not copied from another location.

Will Sun publish the full content of the database?
Sun is currently studying how best to publish the full content of the database, as for some applications, the Web interface to a CGI program is too limiting.

Conclusion

The Solaris Fingerprint Database is a tool available to all of Sun’s customers that allows them to check the validity of operating system files. The Solaris Fingerprint Database enables organizations to verify the integrity of system files through the use of MD5 file fingerprints. Using this tool, administrators can quickly and easily determine whether system binaries, libraries, or other files have been modified from the way in which they had originally been distributed by Sun.

It should also be noted that, in the first six months of availability, several customers reported finding unexpected rootkits on their systems using the Solaris Fingerprint Database.

Bibliography

- The Solaris Fingerprint Database

  `http://sunsolve.Sun.COM/pub-cgi/show.pl?target=content/content7`

- The Solaris Fingerprint Database Sidekick and Companion

  `http://www.sun.com/blueprints/tools/`

- MD5 Message Digest Algorithm (RFC 1321)

  `ftp://ftp.rfc-editor.org/in-notes/rfc1321.txt`

- Basic Auditing and Reporting Tool

  `http://docs.sun.com/app/docs/doc/816-4557/6maosrjds?a=view`

- Integrating BART and the Solaris Fingerprint Database in the Solaris 10 Operating System

  `http://www.sun.com/blueprints/0405/819-2260.pdf`

- Collisions for Hash Functions MD4, MD5, HAVAL-128, and RIPEMD

  `http://eprint.iacr.org/2004/199.pdf`

- HoneyNet project

  `http://www.honeynet.org`

- HoneyNet Project—Know Your Enemy: Motives

  `http://project.honeynet.org/papers/motives/`

- `cksum(1)` manual page

  `http://docs.sun.com/app/docs/doc/816-5165/6mbb0m9d3?a=view`

- `digest` manual page

  `http://docs.sun.com/app/docs/doc/816-5165/6mbb0m9ei?a=view`

- `find(1)` manual page

  `http://docs.sun.com/app/docs/doc/816-5165/6mbb0m9gn?a=view`

- `pkgparam` manual page

  `http://docs.sun.com/app/docs/doc/816-5165/6mbb0m9oe?a=view`

- `sum(1)` manual page

`http://docs.sun.com/app/docs/doc/816-5165/6mbb0m9tc?a=view`

About the Authors

Vasanthan Dasan
Vasanthan Dasan is an Sun Distinguished Engineer based in Colorado, USA. Vasanthan joined Sun Microsystems in 1992 and is currently the Chief Technologist for Sun Services Product Management group. He works on remote servicing, service automation and technology use in service delivery. As Chief Architect for Services Engineering from 1997-2002, Vasanthan led Sun's effort to deliver services via technology. Prior to that, he worked on Solaris products such as CacheFS, AutoClient, Solstice PC Products, and Jumpstart as part of the Solaris engineering team. Vasanthan co-authored Hands-On Intranet, published by Prentice Hall, and has written numerous Sun white papers. He was largely responsible for Sun's early adoption of the Web in 1994, and holds one of the industry's first Web patents, awarded for the invention of Web-based personal newspapers.

Alex Noodergraaf
Alex Noordergraaf, Lead Security Architect for Common Software Features Engineering, has over ten years' experience in the areas of computer and network security. As the Lead Security Architect of the Common Software Features Engineering (CSW) group at Sun Microsystems, he is responsible for providing technical leadership to define the security of Sun's next generation servers, while addressing security for current products. He is the driving force behind the very popular freeware Solaris Security Toolkit. He is author of a number of Sun BluePrints security articles, and he has co-authored three Sun BluePrints books.

Lou Ordorica
Lou Ordorica worked for several years as a system administrator at Sun Microsystems. He went on to teach and write about system administration for Sun's employees and customers, and is currently providing online support to customers using the Web.

Glenn Brunette
Glenn Brunette is a Sun Distinguished Engineer with nearly 15 years' experience in information security. Glenn currently works in Sun's Client Solutions CTO as the Director and Chief Architect of the CSO Security Office. In this role, Glenn is responsible for global security strategy and architecture, security-focused collaboration and knowledge sharing, as well as improving the quality and security of products and services delivered to Sun's customers.

Glenn is the driving force behind Sun's Systemic Security approach and is also an OpenSolaris Operating System Security Community, the co-founder of the Solaris Security Toolkit software, and a frequent author, contributor, and speaker at both Sun and industry events. Externally, Glenn has served as the Vice-Chair of the Enterprise Grid Alliance Grid Security Working Group, and Working Group Champion for the National Cyber Security Partnership's Technical Standards and Common Criteria Task Force. Finally, Glenn is an active contributor to the Center for Internet Security's Unix Benchmark team. Glenn is a

Certified Information Systems Security Professional (CISSP) and has been trained in the National Security Agency's INFOSEC Assessment Methodology (IAM).

Acknowledgments

The authors would like to thank Scott Rotondo and Mark Thacker for their inspiration, technical feedback, and overall support in the development of this article.

Ordering Sun Documents

The SunDocs℠ program provides more than 250 manuals from Sun Microsystems, Inc. If you live in the United States, Canada, Europe, or Japan, you can purchase documentation sets or individual manuals through this program.

Accessing Sun Documentation Online

The `docs.sun.com` web site enables you to access Sun technical documentation online. You can browse the `docs.sun.com` archive or search for a specific book title or subject at `http://docs.sun.com/`.

To reference Sun BluePrints OnLine articles, visit the Sun BluePrints OnLine Web site at:
`http://www.sun.com/blueprints/online.html`