Global Information Assurance Certification Paper

GIAC Certified UNIX Security Administrator (GCUX) Practical Assignment Version 3.0 (revised July 15, 2004) Security Aspects of a Samhain Client/Server Installation to Protect a Solaris Web Server

Winston Holmes

September 19, 2004

Abstract

GIAC Enterprises runs a document repository which is accessible via the web. As part of securing their Unix servers, GIAC has historically relied on the Tripwire Academic Source Release for file monitoring software. GIAC has evaluated several alternatives and has chosen the Samhain File Integrity / Intrusion Detection System as a replacement for Tripwire ASR.

The goal of the installation is to improve file integrity monitoring by installing a central monitoring system that is resistant to tampering by attackers. The sytem will be installed with comprehensive installation documentation and operating procedures such that any system administrator may reinstall or maintain the system. The vulnerabilities and risks associated with this Samhain installation are also addressed.

Introduction

One aspect of securing a system is monitoring the system for unauthorized file modifications such as changing configuration files or the installation of backdoors. Most of this monitoring occurs at the host level and is accomplished with the use of file integrity checkers. The concept of file integrity monitoring is fairly simple. A database of file attributes such as ownership, permissions, modification times, and digital signatures is created for the files and directories of interest. The files are then periodically checked against the database for signs of tampering.

The implementation becomes more complex since an intruder can change the database, replace the integrity monitoring program, or alter the configuration of the integrity checker. With programs such as Tripwire ASR, the defense against such tampering has been to keep copies of the programs and associated files on removable media or read-only media. This becomes unwieldy as the number of machines increases, or frequent changes to the integrity database are required.

The commercial version of Tripwire makes integrity monitoring more tamper resistant by digitally signing and encrypting its configuration and database files. The commercial software also has an option for centralized monitoring and configuration.

While the commercial software is much more capable than the ASR release, the software is also rather expensive. As an alternative, GIAC has chosen to implement Samhain. Samhain provides the nearly the same monitoring facilities and can be configured for centralized monitoring and configuration. Samhain also uses digitally signed configuration and database files. The downside to using Samhain is the complexity of the installation.

Description of Environment

GIAC Enterprises runs a web-accessible document repository. A simplified diagram of the GIAC network is given in Figure 1.

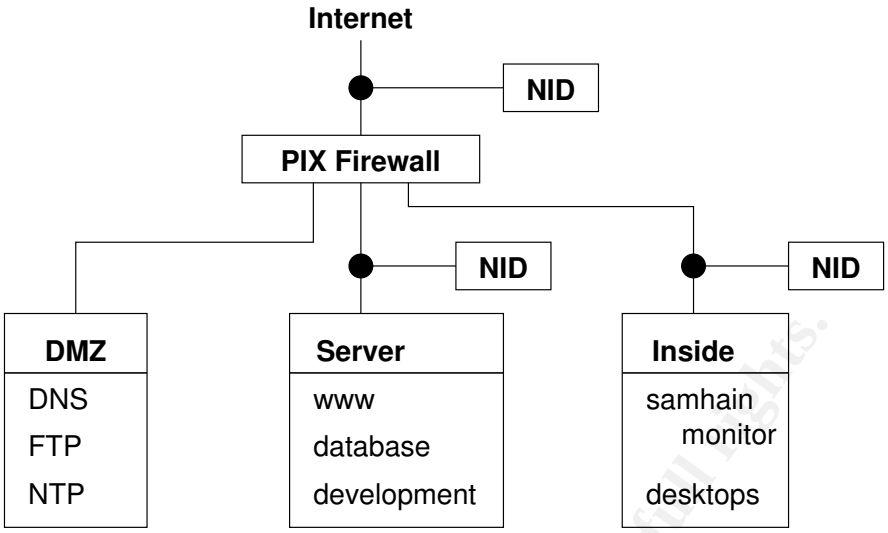

Figure 1: Network Layout

Internally, the network is divided into three subnets: inside, server, and dmz. The **dmz** network contains DNS servers which satisfy external requests, an FTP server, and NTP services. The **inside** network contains user desktops and Windows servers. Initially, the centralized monitor for Samhain will also be located in the inside network. Finally, the **server** network contains a production web server, production database server, and a development server.

A Samhain client will be installed on the production web server. The system is a Sun V440 with 16 GB RAM, four internal 36 GB disks, and two external S3310 arrays. The system is running the Solaris 9 4/04 operating system. The web server software consists of Apache 2.0.50 and Tomcat 4.1.30 with Java 1.4.2_04.

The samhain monitor is a Sun V100 with 512 MB RAM and two internal 40 GB disks. This system was originally setup to provide internal DNS and NTP services. The operating system on this machine is Solaris 9 5/02. Gcc 3.3.2 is used for compiling the software. Both systems receive backups using Legato Networker.

Network Intrusion Detection (NID) systems are located on the external network link and links into the server and inside networks.

The Samhain Integrity Checker

Samhain is a file integrity monitoring system. Samhain monitors the files by comparing the digital signature, ownership, permissions, timestamps, inode, size, minor and major device numbers (devices only), and filename to values stored in a database. Any discrepancies can be reported through many different logging methods. These methods include logging to local and remote files, syslog, email, SQL databases, the Prelude IDS, and user configurable external programs.

Samhain can be setup as standalone software on individual machines, or it can be installed in a client-server configuration to allow centralized monitoring, configuring, and logging. Samhain has the option of digitally signing its configuration and database files in order to eliminate tampering. Samhain can also monitor login and logout events, and Samhain can monitor kernel modules on Linux. As a further deterrent to a compromise of the Samhain software, the package can be setup

to run in stealth mode. In stealth mode, Samhain supports the following countermeasures[1]:

1. Embedded strings in the executable are obfuscated by XORing the strings with a value, *xor_val*, chosen at compile time.
2. The messages in the log file can be obfuscated as well.
3. The log file can be appended to a binary file such as an executable or JPEG file.
4. Strings in the database are obfuscated by XORing with *xor_val*.
5. The configuration file is steganographically hidden in a postscript image file (the image data must be uncompressed).
6. Command line options can be read from stdin rather than the command line in order to further hide the process. Command line input can also be ignored until a password is entered.
7. The executables, scripts, and configuration files can all use alternative names rather than "samhain" or "yule."

Utilities are provided in order to validate and extract content from the obfuscated files. In order of increasing complexity, Samhain can be installed in the following configurations.

1. Samhain standalone.
2. Samhain standalone with signed configuration and database files.
3. Samhain client with Yule server.
4. Samhain client and Yule servers using signed configuration and database files.
5. Samhain client/server installation using the stealth options.

Since this is a new installation, the client/server setup using signed files will be used. The installation is already quite complex and does not need the added complexity of the stealth configuration. Once we are sufficiently comfortable with Samhain, the stealth option may be implemented in the future.

Security Issues Concerning Samhain

The installation of Samhain is being done as an improvement to existing security software. Therefore, this software should certainly not reduce the security of the systems on which it resides. There are several issues that need consideration before Samhain can be installed for production use.

- Samhain software complexity.

[1] Wichmann, Rainer. *Samhain User Manual*. 2004. http://www.la-samhna.de/samhain/manual/stealthmode.html (02 Aug 2004).

- Trust issues with open source software.
- Yule server is network accessible.
- Management host must be secured.
- Network layout and firewall issues.
- Stealth options.
- Passwords and Pass Phrases.
- Adverse impact on systems on which Samhain runs.
- Effective monitoring.
- Initial system integrity.

Samhain Software Complexity

As mentioned before, the installation of Samhain is complex. Good documentation is available, but there are no complete examples detailing the different ways in which Samhain can be installed. A working installation can be setup and tested for the desired requirements. However, the system cannot be maintained by multiple system administrators without good local documentation. This documentation includes the installation details and maintenance procedures. There are numerous small details that affect the success of the Samhain installation. Therefore, it is recommended that the system be installed and documented, followed by removal of the software and a complete reinstallation according to the local documentation. The reinstallation only needs to be done for Samhain. Other prerequisite software such as the GMP Library and GnuPG does not need to be reinstalled as the installation of those two packages is much simpler.

Vulnerability Software complexity.

Risk Complexity increases the risk of a misconfiguration. The software may fall into disuse or be replaced by a simpler, yet less capable package if complexity is frustrating to the system administrators.

Mitigation Document the installation thoroughly and develop procedures for common operation.

Trust Issues with Open Source Software

If a policy on the use of open source software does not exist, management approval for this installation should be obtained. Some companies have strict policies regarding the use of open source or public domain software. Explicit approval must be solicited in this case. Other trust considerations are:

- Does the software work as advertised?
- Does the software do anything suspicious such as "phoning home?"
- Can you trust the source provided from the Samhain web site? Are digital signatures available for verifying the source integrity?

Samhain has been featured in numerous articles on the Internet, and discussed in mailing lists such as Bugtraq. Samhain is actively maintained. Notice of new releases and security issues can be obtained through the Samhain mailing list or by monitoring Freshmeat [2]. Subscriptions to the Samhain-users forum can be obtained at http://lists.la-samhna.de/listinfo/samhain-users.

The Samhain source is digitally signed. Instructions for verifying the integrity are available on the web site and with the program documentation.

Vulnerability Trusting open source software.

Risk The software may not function as designed or may function in an unauthorized and undocumented manner. The source of the software may have been compromised by crackers.

Mitigation The source is digitally signed in order to make it very difficult for crackers to install trojans. Wide-spread use provides many eyes examining the source and getting familiar with the behavior of the program. Acceptance testing will document the expected behavior of the programs.

Network Service Provide by Yule

Adding a network service adds another potential vulnerability to the system running the Samhain monitoring agent yule. The yule program listens on a single port number and does not initiate connections to the clients. The yule program does not need to be run as root which reduces the damage that can be done by compromising the yule program. For further risk reduction, yule can be run in a chroot jail.

Since our initially setup will have the clients and server in different networks, the communication mechanisms of the samhain client and yule server can be verified by looking at the log files on the firewall and examination of the raw data contained on the network intrusion detection systems. Yule can be compiled with support for TCP wrappers and this should be done in order to restrict network access.

Vulnerability Network access to Yule server.

Risk The yule server may be attacked. A successful attack may defeat the the integrity monitoring system.

Mitigation Verify that samhain and yule access the network in the manner documented. Compile yule with TCP Wrappers to minimize the number of machines which can conduct a network attack. Install yule in a chroot jail.

Securing the Yule Server

All configuration and integrity information will be located on the yule server. Therefore, this system is a high value target for an attacker who wishes to hide their tracks. Samhain will also be used to monitor the integrity of the yule server.

This server has only two functions: yule and NTP services. In its past life as a DNS server, the system was already hardened and is accessible only to the system administrators.

Vulnerability The yule server represents a single point of failure with integrity monitoring.

[2] http://freshmeat.net.

Risk Successful compromise of the yule server may defeat the integrity monitoring system.

Mitigation Harden the yule server. Minimize the services run on the system running yule. Examine the running processes and open ports using tools such as **ps**, **netstat**, and **lsof**. Restrict access to the yule server to a very small set of users.

Location of Yule Server on Network

In general, it is best to locate the yule server on the same subnet as the clients that are being monitored. With the server placed in the same subnet as its clients, additional holes in the firewall do not need to be added. Eventually, our yule server will be moved into the server network since all of the monitored systems are in that network. The current location of the yule server does have one benefit. The PIX firewall and two network intrusion detection systems will be able to monitor all traffic to and from the yule server. This will allow verification of the network behavior of samhain and yule. The expected behavior is characterized as follows.

- The yule server listens on its default port of 49777.
- Samhain clients initiate connections to the yule server rather than the server contacting the clients.
- The traffic between the clients and server is encrypted.

Vulnerability Location of the yule server on the network.

Risk The network may become a single point of failure in integrity monitoring. Traffic between samhain clients and the yule server may be sniffed. The yule server may be in a subnet that has a lower security level, ie. is less defended, and may be subject to more avenues of attack.

Mitigation Move the yule server to the subnet of the clients in which it is monitoring after acceptance testing is completed. The traffic between clients and the server should be encrypted by design.

Stealth Configuration Options

The Samhain manual has a section on the security design of the program. [3] Specific recommendations include

- Compile a static binary.
- Strip the binary.
- Use signed database and configuration files.
- Take a look at the stealth options.

[3] Wichmann, Rainer. *Samhain Manual*, page 64-65. A PostScript version of the manual ships with the source code. The online manual is in HTML and divided into sections.

Compiling a static binary for Solaris is difficult, but it can be done for some programs. Hal Pomeranz has instructions for creating static Solaris binaries on his web site[4]. The use of static libraries reduces the dependencies on shared libraries and removes one possible avenue of attack. Unfortunately, Samhain uses `-lresolv` when linking, and there is no libresolv.a. The name resolver routines are only available in a shared library.

The executable is automatically stripped by **make install**, and this installation will used signed database and configuration files.

The remaining method for strengthening the security of Samhain is use use the stealth installation options. The stealth options change the name of the executable, obfuscate the database, and log files, and hide the configuration file. The log file can even be hidden by appending the log file to an existing binary file. This option will not be selected due to the added complexity.

Vulnerability Visibility of Samhain programs.

Risk The visibility of the Samhain processes may alert intruders to the fact that monitoring systems are present. Intruders may choose to attack the monitoring software.

Mitigation The mitigation of using a stealth installation will not be implemented due to the complexity. Note that visible monitoring processes may serve as a deterrent.

Protecting Passwords, Keys, and Pass Phrases

The samhain executables contain a compiled-in key that is either selected randomly by the configure script or specified as an option to the configure command. This key is used to verify the integrity of email messages and log file entries. Consequently, all executables within a Samhain installation should use the same key so that all executables can verify messages and log entries. An intruder could potentially discover the keys by decompiling and analyzing the samhain binary. But, this would take time and skill. The intruder will not have to exert this effort if the key can be easily found. Therefore, the key used in compiling should be treated with the same security as a password. The key is valuable and should be stored in a secure manner such as in a safe.

Samhain uses GnuPG to verify the signed database and configuration files. The root user will sign the database and configuration file for the samhain client. The userid responsible for running the yule server will sign its configuration file. The pass phrases which encrypt the private keys of these two users must be protected. If an intruder can gain access to either of these keys, there is potential for defeating the integrity monitoring.

Samhain uses the Secure Remote Password (SRP)[5] protocol to enable trust between the client and server. Typically, yule is used to generate a random password which is then embedded in the samhain client program. Yule also uses this same password in order to generate an entry for its server configuration file. An intruder with this password can replace the samhain client and successfully communicate with the server. Note that the compiled-in key must also be discovered in order to defeat signature verification on email messages and log file entries. Each client should have its own password, and the yule server will have one authorization entry for each client.

Vulnerability Passwords, keys, and pass phrases are used for signing of configuration files, signing of database files, and authorization between clients and the server.

[4] Pomeranz, Hal. http://www.deer-run.com/~hal/sol-static.txt.

[5] Wu, Tom. *The Stanford SRP Authentication Project*. http://srp.stanford.edu/ (04 Sep 2004).

Risk The system may be defeated if signed files can be modified and resigned by an intruder. A rogue client may be substituted for a host that has been compromised.

Mitigation Protect passwords, keys, and pass phrases with the same level of protection used for the root password. Use samhain to monitor the public and private keys that are used for signing the configuration and database files. The base keys that are used in compiling samhain should not be stored on disk any longer than is necessary to complete the installation. Passwords can be stored in encrypted files if your local policy approves the use of specific programs to manage passwords. Examples of such programs include gpasman [6] and Secure Data Manager [7].

Samhain Impact on Client Systems

The samhain client is typically run as a daemon and continuously monitors files for changes. This monitoring has the potential to adversely impact the local system by stealing CPU cycles and increasing the I/O load. Options exist in samhain to throttle the number of files checked per second, adjust the frequency of checks, and restrict the I/O load. The systems on which samhain runs will be monitored for adverse impacts and configure appropriately if adjustments need to be made.

Vulnerability Samhain is an added CPU and I/O load on client systems.

Risk The primary purpose of the client systems may suffer due to integrity monitoring.

Mitigation Monitor the client systems. Adjust the samhain configuration if necessary in order to reduce the load on the client system.

Monitoring Effectively

Samhain has many configuration directives for logging. Directives must be set properly so that important alerts are not lost in a flood of noise. Following generation of a significant alert, this alert must trigger a notification such as an email or page to the system administrators. The following criteria should be met during the test rollout of Samhain.

- A system which is not experiencing any file changes should not burden the system administrators with emails and/or pages.
- Changes to a monitored file should result in the proper alert appearing within the Samhain log file within some reasonable time limit, ie. one hour.
- Changes to a monitored file should appear in the log file of the yule server within some reasonable time limit.
- Critical alerts should result in an email or page to the system administrators.

The correct files must also be monitored. Some sites choose to monitor nearly every file in /etc, /sbin, /bin, /usr/bin, /usr/lib, etc. Our site will choose to monitor substantially fewer files and choose from:

[6] http://gpasman.sourceforge.net. Gpasman. (04 Sep 2004).
[7] http://sdm.sourceforge.net. SDM: Secure Data Manager. (04 Sep 2004).

- Most files in /etc.
- /sbin.
- All SUID programs.
- All programs that can run as daemons.
- All programs that are useful in examining a system: df, du, ls, ps, lsof, netstat, etc.
- Select libraries in /usr/lib.
- Public and private keys of root and yule accounts.
- Apache executables and modules.

An attacker can defeat the monitoring if the samhain client process is shutdown. The shutdown should be noticed on the yule server. However, an external monitoring facility such as Nagios[8] or Big Brother[9] should also be used to ensure that the samhain client and server processes are running.

Vulnerability Samhain has many logging and alerting options. The messages are verbose, and the samhain clients can be chatty.

Risk Important messages may not be seen in the log files. Misconfiguration may keep important messages from even being logged.

Mitigation Test the configuration options. If necessary, use log monitoring software to improve upon the notification abilities of samhain and yule. Use host monitoring software to ensure that the samhain client and yule server are running.

Initial System Integrity

Adding an integrity monitoring system to a machine that is already compromised will do very little good. The integrity of a system should be good if the system is initially installed, hardened, and patched prior to be being connected to any network. Many people like to automate system installs, and this usually requires that new system be connected to a network during the install. With sufficient resources, a standalone network can be created and dedicated to installation tasks. Naturally, the usual caveats about maintaining physical security also apply.

If these standards were not in place during the initial installation, and the current maintenance procedures were sub-optimal, there are still other methods for verifying initial system integrity. Sun maintains a web site that allows submission of the MD5 signatures of files. After the signatures are submitted, the web server will respond with a list of the files and patch levels that match known signatures. Tools also exist for the purpose of automating the signature submission.

Vulnerability Initial system integrity.

Risk Monitoring a compromised system may lead to a false sense of security.

[8] Nagios. http://www.nagios.com. (30 Aug 2004)
[9] Big Brother. http://bb4.com. (30 Aug 2004)

Mitigation Perform initial system installations in a clean and secure environment. Systems already in place should be qualified by submitting MD5 signatures to the Sun Fingerprint Database. [10]

Installation of Samhain

Samhain can be configured to run standalone, as a client, or as a server. Each configuration and compilation must be done independently.

Two prerequisite packages are the GNU Multiple Precision Arithmetic Library (GMP) and GNU Privacy Guard (GnuPG). GMP is not actually required, but the yule server will run faster by using GMP rather than its own multiple precision math library. GnuPG is used by Samhain for signing the configuration and database files.

GMP Library Installation

The GMP sources are available at www.swox.com/gmp/. The GMP source is signed with a key having the following properties.

```
Name: Swox AB (Software signing key 2004)
Key ID: 0xDB899F46
Key type: 1024 bit DSA
Fingerprint: 73D4 6C36 6746 1E4B D939 7249 5D6D 47DF DB89 9F46
```

The public key server at wwwkeys.eu.pgp.net was searched to locate the public key for key ID 0xDB899F46. The key was copied and pasted into a file and imported into the gpg keyring. The signature for the GMP source tarball, gmp-4.1.3.tar.bz2, is given on the GMP web site.

```
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.2.4 (FreeBSD)

iD8DBQBAkAAdXW1H39uJn0YRAqwDAJ9D9qa6yvh9v5KVqOLyBniC/wxiVQCeP/ge
XYTFoEe8YtCH3oH9a9r4r1s=
=3T2K
-----END PGP SIGNATURE-----
```

This signature was put into a file named `gmp.asc`. The integrity verification of the source can now be done.

```
% gpg --import gmp.key
% gpg --verify gmp.asc gmp-4.1.3.tar.bz2
gpg: Signature made Wed 28 Apr 2004 03:03:57 PM EDT using DSA key ID DB899F46
gpg: Good signature from "Swox AB (Software signing key 2004) <info@swox.com>"
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: 73D4 6C36 6746 1E4B D939  7249 5D6D 47DF DB89 9F46
```

[10] *Solaris Fingerprint Database: An Identification Tool for Solaris Software and Files*. http://sunsolve.sun.com/pub-cgi/show.pl?target=content/content7 (05 Sep 2004).

The key fingerprint from the verify operation must match the fingerprint of the key that is given on the web site. Compilation and installation of GMP can now proceed. GMP will be installed beneath the /opt/gnu directory. Our site manages this directory as a stow software repository.

Stow [11] is a package installation tool that is much easier to use than System V or RPM packages. Stow allows some configuration control over packages by making it easy to add or remove software packages without the effort of actually building a package. Each software package is installed in its own stow directory. In the case of GMP, the prefix option to the configure command is /opt/gnu/stow/gmp-4.1.3. Stow will create or modify symbolic links so that user believes the package is actually installed in /opt/gnu/bin, /opt/gnu/lib, etc.

In the top directory of the GMP source, create a file named runconfigure with the following contents.

```
#!/bin/sh
#
./configure --prefix=/opt/gnu/stow/gmp-4.1.3 ABI=32
```

The `runconfigure` scripts are preferred over entering a command line since the script clearly documents how the configure command was run. The compilation was done as an unprivileged user. Sudo[12] was used for the steps requiring root access.

```
% sh runconfigure
% make
% make check
% sudo make install
% cd /opt/gnu/stow
% sudo stow --verbose gmp-4.1.3
```

For GMP, the `make check` step is very important. This step will compile and run several tests of the library. If any of the tests fail, the problems should be fixed prior to installation.

While GMP is only needed for the yule server, GMP was installed on all systems. The presence of GMP does not present a security risk, and having systems look as similar as possible is generally a good thing. The installation is followed by good configuration management practices which include updating or creating the documentation that goes on the intranet server, and noting the change made to each machine. For our site, a simple note is added to the /diary file noting the installation. Example:

```
* 08/24/2004 holmesw
Installed gmp-4.1.3 in /opt/gnu.
```

GnuPG Installation

GnuPG was not an actively maintained package on our systems. Version 1.0.6 was available beneath /opt/gnu, and version 1.2.1 was installed in /usr/local as a package obtained from www.sunfreeware.com. A new installation of version 1.2.6 will be done for all systems.

Verifying the integrity of the source for this software is particularly important since gnupg is security software and is often used in verifying the integrity of other packages.

[11] GNU Stow. http://www.gnu.org/software/stow/stow.html. (05 Sep 2004).
[12] Sudo. http://www.courtesan.com/sudo. (05 Sep 2004).

```
% md5 gnupg-1.2.6.tar.bz2
MD5 (gnupg-1.2.6.tar.bz2) = b1890f5dfacd2ba7ab15448c5ff08a4e
```

This matches the signature on http://www.gnupg.org/(en)/download/integrity_check.html (08/29/2004).

The README file in the source has more information on verifying the integrity of the source.

```
% gpg --import doc/samplekeys.asc
% gpg --verify gnupg-1.2.6.tar.bz2.sig
gpg: Signature made Wed Aug 25 11:29:58 2004 EDT using DSA key ID 57548DCD
gpg: Good signature from "Werner Koch (gnupg sig) <dd9jn@gnu.org>"
gpg: checking the trustdb
gpg: no ultimately trusted keys found
gpg: Note: This key has expired!
Primary key fingerprint: 6BD9 050F D8FC 941B 4341  2DCC 68B7 AB89 5754 8DCD
```

Verify this is the real key by comparing the output of `gpg --fingerprint` with the fingerprint published elsewhere.

```
% gpg --fingerprint 0x57548DCD
pub  1024D/57548DCD 1998-07-07 Werner Koch (gnupg sig) <dd9jn@gnu.org>
     Key fingerprint = 6BD9 050F D8FC 941B 4341  2DCC 68B7 AB89 5754 8DCD
```

One source is www.gnupg.org/(en)/signature_key.html, which gives the same fingerprint.

GnuPG will be installed in our /opt/gnu stow repository. Samhain will use the TIGER192 algorithm for signing the configuration and database files. By default, this algorithm is not used in the compilation of GnuPG. The option to use TIGER192 is "`--enable-new-tiger`" when configuring GnuPG.

The runconfigure script is created with the following contents.

```
#!/bin/sh
#
./configure --prefix=/opt/gnu/stow/gnupg-1.2.6 --enable-new-tiger
```

GNU make must be used for the installation of GnuPG. Existing versions of GnuPG must be removed prior to the installation.

```
% sh runconfigure
% gmake
% sudo gmake install
% cd /opt/gnu/stow
% sudo pkgrm SMCgnupg
% sudo stow --verbose --delete gnupg-1.0.6
% sudo stow --verbose gnupg-1.2.6
% sudo chmod 4755 /opt/gnu/stow/gnupg-1.2.6/bin/gpg
```

The final chmod command will make gpg a setuid root program. This command is necessary on Solaris in order to allow gpg to lock pages in memory. If pages are not locked in memory, there is the possibility of allowing pass phrases to be written to the swap space on disk. Without the setuid bit, the gpg program will give an error message about "insecure memory."

```
gpg: WARNING: using insecure memory!
gpg: please see http://www.gnupg.org/faq.html for more information
```

The program will still continue to work even if it is not setuid root. For sites that do not install gpg as setuid root, the warning message can be disabled by adding `no-secmem-warning` to either the `~/.gnupg/option` or `~/.gnupg/gpg.conf` files. Normally, the permissions on an suid program should be execute-only for group and other. But, the samhain configuration will use gpg to generate a checksum of the gpg program. This operation will fail if the gpg binary is not readable. The permissions can be strengthened following the successful installation of samhain.

Samhain Client Installation

GnuPG and the creation of public and private keys for the root user are prerequisites for installing the samhain client.

The source for Samhain 1.8.10b was downloaded from the Samhain web site. The first step is to verify the source integrity. Untarring the source tarball will extract another source tarball and an ASCII signature file for that source tarball. The public key for key id 0F571F6C can be downloaded from the public key server blackhole.pca.dfn.de.

```
% gpg --keyserver blackhole.pca.dfn.de --recv-keys 0F571F6C
% gpg --verify samhain-1.8.10b.tar.gz.asc samhain-1.8.10b.tar.gz
gpg: Signature made Tue Jul 13 12:07:48 2004 EDT using DSA key ID 0F571F6C
gpg: Good signature from "Rainer Wichmann <rwichmann@la-samhna.de>"
gpg:                  aka "Rainer Wichmann <rwichmann@hs.uni-hamburg.de>"
gpg: checking the trustdb
gpg: checking at depth 0 signed=0 ot(-/q/n/m/f/u)=0/0/0/0/0/1
gpg: next trustdb check due at 2009-08-29
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: EF6C EF54 701A 0AFD B86A  F4C3 1AAD 26C8 0F57 1F6C
```

The key fingerprint matches the fingerprint in the documentation and on the website.[13]

The checksum of the gpg executable should be compiled into the samhain client. The checksum can be obtained with:

```
% gpg --load-extension tiger --print-md TIGER192 /opt/gnu/bin/gpg
gpg: WARNING: digest 'TIGER192' is not part of OpenPGP.
  Use at your own risk!
/opt/gnu/bin/gpg: 7D0A5014 15A7E093 B1BB0F3A 57B7C789 DD77B74A 9BE7B4F8
```

Like the earlier installations, a runconfigure script will be created to handle the many options required for samhain.

–enable-network=client This option builds the samhain client as opposed to the yule server. If this option is not given, a standalone version of samhain is build.

[13] Samhain Labs PGP Key. http://www.la-samhna.de/samhain/s_rkey.html. (04 Sep 2004).

–prefix=OPT Samhain's directory layout does not use `--prefix` in the manner as most Open Source programs. With this option, pieces of Samhain will be installed in /opt/samhain/bin, /etc/opt, and /var/opt/samhain. As this layout was not desired, additional configuration options were specified in order to confine samhain to /opt/samhain and /var/samhain. Similar options are used in the yule server compilation.

–with-gpg=/opt/gnu/bin/gpg Specify the location of the **gpg** program.

–with-checksum="value" Specify the TIGER192 checksum of the gpg program.

–enable-mounts-check Compile in a module to check for correct mount options.

–enable-base=1234567,7654321 Specify the base keys that are used for auditing log and email messages. The same keys must be specified in different program compilations if those programs need the ability to verify audit trails of each other. The keys must be in the range 0 to 2,147,483,647. The actual configuration used base keys that are more random than the ones listed in this documentation.

–with-logserver=IP_Address Specify the IP address of the log server. The option must be specified if configuration files are to be fetched from the log server.

–with-log-file=/var/samhain/samhain.log Specify the location of the log file.

–with-config-file=REQ_FROM_SERVER/var/samhain/samhain.conf Specify the location of the configuration file. When initializing the database, the path that is specified will be used as a fallback if the connection to the server fails. The server will provide a configuration file using the following names in order of priority:

1. *localstatedir*/lib/yule/rc.*CLIENTNAME*
2. *localstatedir*/lib/yule/rc

At least, this is what the manual stated. With our configuration options and the chroot installation of yule, the configuration file for the client web.giac.org ended up as /yule/var/yule/rc.web.giac.org on the yule server.

–with-data-file=REQ_FROM_SERVER/var/samhain/samhain.db Specify that the client database file should be downloaded from the server when checking. Initialization of the database will be done using the local file /var/samhain/samhain.db. The server will provide a database file using the following names in order of priority:

1. *localstatedir*/lib/yule/file.*CLIENTNAME*
2. *localstatedir*/lib/yule/file

Again, the manual did not match our configuration. The database for the client web.giac.org ended up as /yule/var/yule/file.web.giac.org. The samhain client will not start if it cannot find both the configuration file and its database. If the samhain client can make a successful connection to the yule server, the error message in the yule log will give the expected name of the configuration or database file.

–with-pid-file=/var/run/samhain.pid Specify the location of the file containing the process ID of the samhain program.

–with-state-dir=/var/samhain Specify the state data directory.

With these options, our runconfigure script is shown below.

```
#!/bin/sh
# Samhain client configuration
#
./configure --prefix=OPT \
  --enable-network=client \
  --with-gpg=/opt/gnu/bin/gpg \
  --with-checksum=\
"/opt/gnu/bin/gpg: 54ED11BE D08CAD2D 18BA53A0 66AE573F 5053C27F D86D213E" \
  --enable-mounts-check \
  --enable-base=1234567,7654321 \
  --with-logserver=10.0.0.6 \
  --with-log-file=/var/samhain/samhain.log \
  --with-config-file=REQ_FROM_SERVER/var/samhain/samhain.conf \
  --with-data-file=REQ_FROM_SERVER/var/samhain/samhain.db \
  --with-pid-file=/var/run/samhain.pid \
  --with-state-dir=/var/samhain
```

The software is then configured and compiled with the normal sequence of commands.

```
% sh runconfigure
% make
```

The `make install` procedure will install a configuration file for samhain and attempt to sign that file. If this command is run with **sudo**, the environment settings will end up signing the configuration file with the private key of the user running sudo. This is not what is desired as the root user private key should be used. The work around is to actually become the root user with a command such as `su - root`. Alternatively, sudo can still be used to obtain a root shell with environment settings that are correct for performing the installation. The scripts for running samhain at system boot will be installed by running `make install-boot`.

```
% sudo -u root sh -c "HOME=/ ENV=/.kshrc /bin/ksh"
# make install
# make install-boot
```

Modifying, removing signatures, and adding signatures to the database and configuration files can be cumbersome. Samhain has a utility script to simplify these operations. This script should be copied to /opt/samhain/bin. For version 1.8.10b, there is an error in line 453; `create-cfgfile` should be changed to `create-datafile`. The script **samhainadmin.pl** is located in the scripts directory.

This is as far as we can procede with the samhain installation until the yule server is installed and running.

Samhain Server (Yule) Installation

The prerequisites for installing the Samhain server yule are the installation of GnuPG and the GMP library. Additionally, public and private keys already be generated for the root and yule users. Naturally, the yule user must already exist. TCP Wrappers should also be installed if TCP Wrappers will be used to restrict client connections.

The following configuration options are used in building the yule server. The chroot configuration is done at installation time. This is convenient as a standard installation can be done and debugged prior to attempting the chroot installation.

–prefix=OPT This will put the binaries in /opt/yule/bin. Configuration and log files will be put into other locations, but additional configuration options will be specified to place files into /var/yule.

–enable-network=server The yule server will be built.

–enable-identity=yule The yule account will be used when root privileges are dropped.

–with-sender=yule The yule username will be used when sending emails.

–with-gpg=/opt/gnu/bin/gpg GnuPG will be used to verify the database and configuration files.

–with-checksum="value" The TIGER checksum of the gpg executable is compiled into yule.

–with-libwrap=/usr/local/lib TCP Wrappers will be used to restrict access to the yule server.

–enable-base=1234567,7654321 Specify the base key for one-time pads. Binaries compiled with different base keys will not be able to produce audit trails of each other.

–with-config-file=/var/yule/yule.conf Specify the configuration file.

–with-log-file=/var/yule/yule.log Specify the log file.

–with-pid-file=/var/yule/run/yule.pid Specify the file containing the process ID of the yule server.

–with-state-dir=/var/yule Specify the state data directory.

The runconfigure script is given below.

```
#!/bin/sh
#
# Samhain-1.8.10b server (yule) configuration
#
./configure --prefix=OPT \
  --enable-network=server \
  --enable-identity=yule \
  --with-sender=yule \
  --with-gpg=/opt/gnu/bin/gpg \
  --with-checksum=\
```

```
"/opt/gnu/bin/gpg: A7DB62AB A4110B51 6D77FC41 0EFBD95A D07538DA 7F09EBBF" \
  --with-libwrap=/usr/local/lib \
  --enable-base=1234567,7654321 \
  --with-config-file=/var/yule/yule.conf \
  --with-log-file=/var/yule/yule.log \
  --with-pid-file=/var/yule/run/yule.pid \
  --with-state-dir=/var/yule
```

Yule will be installed using /yule as the directory of its chroot jail. [14]

```
% sh runconfigure
% make
% sudo -u root sh -c "HOME=/ ENV=/.kshrc /bin/ksh"
# mkdir /yule
# make DESTDIR=/yule install
# make DESTDIR=/yule install-user
# make install-boot
```

Errors occur during the "make install" step as the root user cannot sign the configuration file using the yule user's private key. The template file for the yule configuration is **yulerc**, found in the top level of the Samhain source directory. This file should be copied as **yule.conf** to the configuration directory, edited, and then signed.

The samhainadmin.pl script should be copied to the yule bin directory, and the error in the script on line 453 should be fixed.

```
# cp scripts/samhainadmin.pl /yule/opt/yule/bin
  edit the script and fix line 453
  s/create-cfgfile/create-datafile/
# cp yulerc /yule/var/yule/yule.conf
# chown -R yule:yule /yule/var/yule
# chmod 750 /yule/var/yule
```

A samhain client installation will also be done on the yule server. This will result in two samhainadmin.pl scripts being installed. Symlinks, aliases, or renaming the script is suggested if full pathes are not used when accessing the commands. Example:

```
# cd /yule/opt/yule/bin
# ln -s samhainadmin.pl yadm
```

To avoid confusion, the full path to samhainadmin.pl will be used in the examples.

The yule user will need a home directory in the chroot environment, and several files need to be present in the /etc directory of the chroot directory.

```
# mkdir -p /yule/home/yule /yule/etc
# cd /etc
# egrep 'yule|root' passwd > /yule/etc/passwd
# cp nsswitch.conf hosts resolv.conf services protocols /yule/etc
# cp hosts.allow hosts.deny /yule/etc
```

[14] *Samhain Manual*, page 46.

Yule will need a source of entropy for its cryptographic functions. Find the major and minor nodes of the /dev/random and /dev/urandom devices and create the devices in /dev/of the chroot environment.

```
# cd /yule
# mkdir dev
# cd dev
# ls -l /devices/pseudo/random*
# mknod random c 190 0
# mknod urandom c 190 1
```

Edit the /yule/etc/passwd file and insert an asterisk for the passwords. Since we are using the TCP wrappers option, hosts.deny and hosts.allow files must be present in the chroot environment. The hosts.deny configuration should block all connections. For hosts.allow, the name of the service is "yule."

GnuPG must be available in the chroot environment.

```
# mkdir -p /yule/opt/gnu/bin
# cp /opt/gnu/bin/gpg /yule/opt/gnu/bin
```

The startup script, /etc/init.d/yule, must be modified to have the correct location of yule binary. According to the manual, the chroot option is specified either as a command line option (`--chroot=`*/chrootdir*) or as a directive in the `Misc` section of the configuration file. In practice, the configuration directive did not seem to take effect, and the option had to be specified in the startup script. The modified script is given in an appendix.

At this point, yule can be started if the configuration file has been edited and signed. However, no clients have been configured.

Connecting the Samhain Client to the Yule Server

Clients must be properly registered with the yule server in order to send log messages and download configurations or databases. A password is embedded in the samhain client executable. A hash of this password is placed in the yule configuration file in order to grant access. The password is 16 hexadecimal digits and can be randomly generated by yule. Yule also has a facility for generating the salt and hash that are placed in the configuration file.

```
% yule --gen-password
92C272EB4589AFC8
yule % /yule/opt/yule/bin/samhainadmin.pl --remove-signature yule.conf
yule % yule --password=92C272EB4589AFC8 | \
 > sed -e 's/HOSTNAME/web.giac.org/' > yule.conf
yule % /yule/opt/yule/bin/samhainadmin.pl --sign yule.conf
 enter the passphrase for the yule private key.
yule # /etc/init.d/yule restart
```

On the client, embed the password within the samhain executable.

```
client # cd /opt/samhain/bin
client # ./samhain_setpwd samhain new 92C272EB4589AFC8
INFO   old password found
INFO   replaced:  f7c312aaaa12c3f7  by:  92c272eb4589afc8
INFO   finished
client # mv samhain samhain.orig
client # mv samhain.new samhain
```

The samhain.orig file should be removed after things are working. Otherwise, an attacker may easily create a new samhain binary with a password of their choosing.

Since TCP Wrappers is used, edit /yule/etc/hosts.allow and add in the client's IP address. The connectivity can be verified by starting samhain and checking the log messages on the client and server.

Maintenance and Operation Procedures

Samhain Updates

Update notices to Samhain are posted to the Samhain web site [15] and Freshmeat [16]. But, the most reliable method for receiving notices of updates is to subscribe to the samhain-users e-mail list.[17] Significant security issues with Samhain are also likely to be posted to the Bugtraq full-disclosure list.[18]

Appropriate Permissions on Files and Directories

The files of significance for the samhain client are in /var/samhain. This directory should be owned by root and readable only by root.

The yule server is chrooted to the directory /yule. The /yule/var/yule subdirectory should be read/write for the yule user. This subdirectory will contain the yule configuration, yule log file, and the configuration and database files for the clients. Since the root user signs the client files, these files will consequently be owned by root. The client files must still be readable for the yule user.

File/Dir	Name	Owner	Mode
Dir	/var/samhain	root	700
File	/opt/samhain/bin/samhain	root	700
File	/opt/samhain/bin/samhain_setpwd	root	700
Dir	/yule/var/yule	yule	700
File	/yule/var/yule/rc.*client_fqdn*	root	644
File	/yule/var/yule/file.*client_fqdn*	root	644
File	/yule/var/yule/yule.conf	yule	600
File	/yule/var/yule/yule.log	yule	600

[15] http://www.la-samhna.de/samhain/. (02 Aug 2004).
[16] http://freshmeat.net.
[17] Samhain-users Forum. http://lists.la-samhna.de/listinfo/samhain-users. (05 Sep 2004).
[18] Bugtraq. http://www.securityfocus.com/archive/1. (05 Sep 2004).

Initializing the Database

Initialization is easiest to do if samhain is not already running as a daemon.

1. Stop samhain if it is running and then run samhain to create the initial database. The database file /var/samhain/samhain.db should not already exist, otherwise the new data will just get appended to the database file.

   ```
   client # /etc/init.d/samhain stop
   client # samhain -t init
   ```

2. Messages should indicate a successful download of the configuration file. Subsequent messages list the files that are being checked. This should create /var/samhain/samhain.db. This database must be signed. `/opt/samhain/bin/samhainadmin.pl --create-datafile` will sign the file, but tries to move it to REQ_FROM_SERVER/var/samhain/samhain.db which causes an error. Therefore, sign the database with

   ```
   client # /opt/samhain/bin/samhainadmin.pl --sign samhain.db
   ```

   An alternative is to sign the database after copying it to the yule server.

3. Copy the file to the yule server.

   ```
   client # scp samhain.db yule@dns1:/yule/var/yule/file.web.giac.org
   ```

4. Start samhain in daemon mode on the client.

   ```
   client # /etc/init.d/samhain start
   ```

If the initialization is to be done with samhain already running, the options should be specified to prevent logging to the server and the local log file. After the database is available on the server, the samhain client should reload the database.

```
# samhain -l none -e none -t init
# /opt/samhain/bin/samhainadmin.pl --sign samhain.db
# scp samhain.db yule@dns1:/yule/var/yule/file.web.giac.org
# /etc/init.d/samhain reload
```

Updating the Database

Samhain will not successfully download the remote database and perform an update. The database must first be copied to the client, the signature must be removed, and then the update can be run.

The samhain client was shutdown during this update. If the client is not shutdown, use the "-l none" option to prevent simultaneous access to the log file. Also use "-e none" to avoid concurrent access to the server.

```
# cd /var/samhain
# scp yule@dns1:/yule/var/yule/file.web.giac.org samhain.db
# /opt/samhain/bin/samhainadmin.pl --remove-signature samhain.db
```

Update the database with:

```
# samhain -t update
```

An update can also be run with interactive questions regarding whether file parameters should be updated.

```
# samhain -t update --interactive
# /opt/samhain/bin/samhainadmin.pl --sign samhain.db
# scp samhain.db yule@dns1:/yule/var/yule/file.web.giac.org
```

Modifying the Samhain Configuration

1. Find the desired configuration file on the server. For our configuration, this file should be named /yule/var/yule/rc.*client_fqdn*, ie. rc.web.giac.org.
2. Remove the signature.

   ```
   # /yule/opt/yule/bin/samhainadmin.pl --remove-signature rc.web.giac.org
   ```

3. Edit and save the file. Then, sign the configuration with the root key.

   ```
   # /yule/opt/yule/bin/samhainadmin.pl --sign rc.web.giac.org
   ```

4. Restart the samhain client

   ```
   # /etc/init.d/samhain restart
   ```

If the changes concern the files that need to be monitored, then the database needs to be updated. It is usually easiest to just initialize a new database.

Modifying the Yule Configuration

1. Find the desired configuration file on the server. For our configuration, this file should be named /yule/var/yule/file.*client_fqdn*, ie. file.web.giac.org.
2. Remove the signature from the file.

   ```
   % cd /yule/var/yule
   % /yule/opt/yule/bin/samhainadmin.pl --remove-signature yule.conf
   ```

3. Edit and save the file. Sign the configuration with the yule key.

   ```
   % /yule/opt/yule/bin/samhainadmin.pl --sign yule.conf
   ```

   Restart the server.

   ```
   # /etc/init.d/yule restart
   ```

Adding a New Client

Creating a new client is a lengthy procedure that is done by following the instructions for several different operations.

1. Follow the instructions for compiling and installing samhain. Or, copy an existing installation of /opt/samhain from one client to another if the systems are of similar architecture and OS. The prerequisites of the GMP library and GnuPG must also be satisfied.
2. Connect the client to the server as shown in the next section.
3. On the yule server, create a samhain configuration file for that client. This is easiest to do by copying an existing configuration file. The configuration file should be named rc.*client_fqdn*, ie. rc.medusa.giac.org for the host medusa. The configuration file must be signed using the root key.
4. Initialize the client database, sign the database, and store the database on the yule server.
5. Start the samhain daemon to begin monitoring.

Rotating Log Files

Samhain has a mechanism for dropping the lock on its log file and pausing to allow the movement of the current log file. Sending SIGABRT to the samhain process will tell samhain to finish its current task and unlock the log file (remove /var/samhain/samhain.log.lock). Samhain will then pause for three seconds before resuming operation. The following script is a ksh implementation of the bash script on page 12 of the manual. /opt/samhain/bin/rotatelog:

```
#!/bin/ksh
# Rotate the Samhain log file
#
PIDFILE=/var/run/samhain.pid
LOCKFILE=/var/samhain/samhain.log.lock
LOGFILE=/var/samhain/samhain.log
OLDLOG=/var/samhain/samhain.log.old
#
if [ -f "$PIDFILE" ]; then
    PID=‘/bin/cat $PIDFILE‘
    /bin/kill -ABRT $PID
    sleep 1
    AA=0
    while [ "x$AA" != "x120" ]; do
        AA=$(($AA+1))
        if [ -f "$LOCKFILE" ]; then
            sleep 1
        else
            break
        fi
    done
```

```
fi
if [ -f "$LOGFILE" ]; then
    ### echo "Rotate Samhain log file"
    /bin/mv $LOGFILE $OLDLOG
else
    echo "$LOGFILE not found"
fi
```

A similar script is available for rotating the yule log file.

Verifying Log File Messages and Emails

The verification of email messages and log file messages relies on hashing. A random key is generated and sent by email. Additional keys are generated by a hash chain. Therefore, all messages can only be verified by knowing this initial key (generated at startup). Our use of Outlook makes this more difficult, but the emails can be verified. Forward the email messages from Outlook to a Unix system with Samhain installed. Start with the initial key message (the one with "BEGIN LOGKEY").

```
# samhain -M /var/mail/holmesw
Message 000001  Trail 1095516726::dns1.giac.org
(passed)
```

The mailbox was intentionally modified by changing "service" to "SERVICE", and the integrity check was rerun.

```
# samhain -M /var/mail/holmesw
Message 000001  Trail 1095516726::dns1.giac.org
(FAILED)
```

Verifying a log file also requires access to the LOGKEY. The email message with the key looks similar to the following. Line wrapping is courtesy of Outlook.

```
-----BEGIN MESSAGE-----
[2004-09-18T10:12:06-0400] dns1.giac.org
ALERT  :  [2004-09-18T10:12:06-0400] msg=<LOGKEY>, program=<Yule>,
hash=<1FE7E1E7F8456F67753EECE3A75BDB22DD829A712BBBDEB7>
-----BEGIN LOGKEY-----
1FE7E1E7F8456F67753EECE3A75BDB22DD829A712BBBDEB7[2004-09-18T10:12:06-0400]
ERROR  :  [2004-09-18T10:12:06-0400] msg=<Service failure>,
service=<console>, obj=</dev/console>
-----BEGIN SIGNATURE-----
927C4F553B5AC994BF36E6DFFA66DA87A8204F324CB9844B
000001 1095516726::dns1.giac.org
-----END MESSAGE-----
```

The log file or a section of the log file can also be verified. For this example, the last few lines of the yule log file were put into a separate file just after the startup of yule. Lines have been wrapped for clarity. The key was obtained from an email sent just after the startup of yule.

```
# yule -L /tmp/yule.log
New audit trail ([2004-09-18T10:12:06-0400]), enter key|keyfile:
 1FE7E1E7F8456F67753EECE3A75BDB22DD829A712BBBDEB7
PASS:  line=    1 ERROR  :  [2004-09-18T10:12:06-0400]
 msg=<Service failure>, service=<console>, obj=</dev/console>
PASS:  line=    3 MARK   :  [2004-09-18T10:12:06-0400] msg=<Server up,
 simultaneous connections: 249>, socket_id=<3>
PASS:  line=    5 <TCP>  :  [2004-09-18T10:14:46-0400] client=<web>,
 msg=<MARK   :  [2004-09-18T10:14:45-0400] msg=<---- TIMESTAMP ---->>
PASS:  line=    7 <TCP>  :  [2004-09-18T10:21:29-0400] client=<dev>,
 msg=<MARK   :  [2004-09-18T10:21:28-0400] msg=<---- TIMESTAMP ---->>
PASS:  line=    9 MARK   :  [2004-09-18T10:22:14-0400]
 msg=<---- TIMESTAMP ---->
PASS:  line=   11 <TCP>  :  [2004-09-18T10:24:45-0400] client=<web>,
 msg=<MARK   :  [2004-09-18T10:24:45-0400] msg=<---- TIMESTAMP ---->>
```

Log File Review

The most important messages in the log files and email are those concerning the unauthorized modification of files. Theses messages are clearly indicated by the presence of "CRIT" in the log file line. The following log entry occurred as a result of modifying the rotatelog script.

```
<TCP>  :  [2004-09-18T13:05:09-0400] client=<web>, msg=<CRIT   :
  [2004-09-18T13:05:09-0400] msg=<POLICY [ReadOnly] C-------TS>,
  path=</opt/samhain/bin/rotatelog>, size_old=<609>, size_new=<584>,
  ctime_old=<[2004-09-18T15:01:40]>, ctime_new=<[2004-09-18T15:06:31]>,
  mtime_old=<[2004-09-18T15:01:40]>, mtime_new=<[2004-09-18T15:06:31]>,
  chksum_old=<3144A2A905CB590B1B010B2EFCA37C97EEA60EB6FF3054DD>,
  chksum_new=<4A9E628A9059686898A182FA4B7A228173A273EBA56D84B6>, >
```

This entry has been conveniently split into multiple lines in order to be more readable. The line specifies the time of detection (September 18 at 13:05), the host (web), the policy violation (ReadOnly), and the file qualities that have changed. The messages have a short code which indicates which properties have been modified. In this example, the code is C-------TS. The meaning of codes is given in the table below.

Code	Property
C	checksum
L	soft link
D	device number
I	inode
H	number of hardlinks
M	mode
U	user
G	group
T	time
S	size

All critical messages are sent via email to the system administrators.

Verifying Initial System Integrity

The Solaris Fingerprint Database (sfpDB)[19] is a free SunSolve Online service that enables users to verify the integrity of files distributed with the Solaris OS.

The sfpDB is accessed online at http://sunsolve.sun.com/pub-cgi/fileFingerprints.pl. Md5 signatures for key files can be pasted into the form. Submitting the form should return the exact distribution and patch level for each signature. Unmatched signatures may indicated a trojaned binary.

The Solaris Fingerprint Database Companion (sfpC) is an alternative to the web form and automates the process of collecting and checking Md5 signatures against the sfpDB. The sfpC makes it much easier to check large lists of Solaris OS files by automating the submission of the files to the sfpDB website.

The Solaris Fingerprint Database Sidekick (sfpS) is a script, `sidekick.sh`, that works in conjunction with sfpDB and spfC by simplifying the process of checking a system for rootkits. Sidekick does this by maintaining a list of commonly trojaned Solaris executables. An alternative is to use the list of programs and daemons from our Samhain configuration.

SfpC requires the following Perl modules: HTTP::Request, LWP::UserAgent, and HTML::Parser. Naturally, these modules require other modules. The SfpC tool does not have to be installed on the system that is being checked. This allows checking to be carried out without having to modify the Perl installation on our production servers. Lists of signatures from remote hosts can be copied to the system with SfpC and then checked.

The following discussion shows how this was used on web.giac.org to get the md5 signatures and then run the check from another system. On the web server, use the samhain configuration to get a list of important system binaries. The file "**files**" contains a list of binaries which should be checked, and **web.md5** will contain the Md5 signatures of these programs.

```
# touch web.md5
# for f in ‘cat files‘; do
> echo $f
> md5 $f >> web.md5
> done
```

Copy the web.md5 file to a system which has the sidekick tool installed.

```
% ./sfpC.pl web.md5 > web.md5.out
% grep ’0 match’ web.md5.out
```

There should be no output as a result of the grep command. The web.md5.out file contains the digital signatures and the match from the Sun database. Example:

```
 1ce0da7fb7ad3d63fbfbd2ce87267437 - (/usr/lib/sendmail) - 1 match(es)

 canonical-path: /usr/lib/sendmail
 package: SUNWsndmu
 version: 11.9.0,REV=2002.04.06.15.27
```

[19] Dasan, Vasanthan., Noordergraaf, Alex., and Ordorica, Lou. *The Solaris Fingerprint Database - A Security Tool for Solaris Operating Environment Files*. http://www.sun.com/blueprints/0501/Fingerprint.pdf. (12 Sep 2004).

```
architecture: sparc
source: Solaris 9/SPARC
patch: 113575-05
```

Simple checks for the installation of rootkits can be done using **chkrootkit**, available from www.chkrootkit.org. This software looks for strings in system binaries that may have been trojaned, and also looks for signs of compromise based on behavior of known rootkits. The software is easily compiled or is available as a package from the Sun Freeware site. The results of this tool are not absolute guarantee that a system has not been compromised, but it does provide further assurance.

Testing and Qualifying the Installation

Samhain has numerous features that are supposed to make it difficult for intruders to tamper with the configuration and database files. This behavior will be tested. Additionally, the use of Samhain should burden the host system with excessive CPU and IO load. Tests and expected results are given in the following table.

Test	Expected Results
CPU and IO load	No adverse impacts
Change recognition	Email is received in a timely manner.
Tampered configuration	Samhain alerts and refuses to run.
Network connections	Only from client to server
Connection authentication	Authorized clients only
Open ports	Port 49777 on server

CPU and IO Load on the Client System

The CPU time used by the samhain and yule processes can be obtained with the **ps** command. Our lean samhain configuration results in about one minute of CPU time per day. Unless the CPU load increases substantially, there is no point in checking for excessive IO.

If the configuration is changed, and the CPU load increases, samhain has several options for improving performance.

- Change the checksum algorithm from TIGER to MD5 by setting `DigestAlgo=MD5` in the configuration file. This is reported to be about 20% faster. There is a trade off in that the MD5 checksum may not be as secure as TIGER.
- Use the Sun compiler, or optimize more aggressively.
- Reduce the priority of the samhain process with a configuration directive such as `SetNiceLevel=19`.
- Limit the I/O rate with the SetIOLimit directive. For example, `SetIOLimit=1000` sets the rate to 1000 kilobytes per second.

Change Recognition

Test the following changes to files monitored by samhain to verify that the changes are recognized and notifications are sent.

- Edit a file such as /etc/motd.
- Perform a touch on a ReadOnly file, ie. `touch /bin/ps`.
- Add a file into a monitored directory, ie. `touch /opt/samhain/bin/junk`.

These tests were conducted. In all cases, email notifications were sent in less than 1.5 hours.

Recognizing a Tampered Configuration File

Run the test by shutting down samhain on a client. Add a single character to the configuration file without updating the signature and restart samhain. Samhain should exit soon after startup and put a notice about the bad signature in its log file.

```
ERROR  :  [2004-09-19T11:37:58-0400] msg=<No good signature>,
 subroutine=<gpg_check_file_sign>
A7D91DFC492D2FCA86055BFFE2507D5A591DC89728D2D915[2004-09-19T11:37:58-0400]
ALERT  :  [2004-09-19T11:37:59-0400] msg=<PANIC Error initializing
 the application>, program=<Samhain>
30C8D4D05276331E0D43DBAA8D1572412122CC3D860D370E
ALERT  :  [2004-09-19T11:37:59-0400] msg=<EXIT>, program=<Samhain>,
 status=<None>
8020D505C1EC930834377C5264C958051D38610C38C8BBD8
```

The yule server will also log the failed startup, and emails are sent regarding the shutdown and aborted startup.

Network Connections

The Samhain documentation states that the yule server only listens for connections from the clients. The yule server does not initiate contact with the client systems. Since a firewall is placed between our server and the clients, the firewall logs can be examined to see the nature of the connections.

A Cisco PIX firewall is used. Firewall messages are collected into daily log files for post-processing and reporting. This examination looked at the log files for September 10 - 16. First, extract all of the log messages related to the yule server, 10.0.0.6.

```
# for f in 10 11 12 13 14 15 16; do
> egrep '10\.0\.0\.6' 200409${f}.pix > yule.$f
> done
# ls -l yule.*
-rw-r--r--   1 root     other    1200701 Sep 17 11:29 yule.10
-rw-r--r--   1 root     other    1376474 Sep 17 11:30 yule.11
-rw-r--r--   1 root     other    1346647 Sep 17 11:30 yule.12
```

```
-rw-r--r--   1 root     other    1150117 Sep 17 11:31 yule.13
-rw-r--r--   1 root     other    1082290 Sep 17 11:32 yule.14
-rw-r--r--   1 root     other    1363149 Sep 17 11:33 yule.15
-rw-r--r--   1 root     other    1125343 Sep 17 11:35 yule.16
```

Since this system provides NTP services and uses DNS, the log files can be further reduced by removing the lines associated with these two services.

```
# for d in 10 11 12 13 14 15 16; do
> echo $d
> egrep -v '/53 |/123 ' yule.$d > sam.$d
> done
```

Messages left in the file look similar to:

```
Sep 10 10:36:05 [10.0.1.157.2.2] %PIX-6-302013: Built inbound TCP
  connection 285668576 for Server-net:10.0.1.133/39208
  (10.0.1.133/39208) to inside:10.0.0.6/49777 (10.0.0.6/49777)
```

This particular message is for the client making a connection to the yule server on port 49777.

The command `egrep -v 'to inside'` will display all of the messages which are not related to inbound connections to the yule server. This left only a few messages for SSH connections on September 11. All other days did not have any outbound connections.

Therefore, the conclusion is that the yule server only listens on port 49777 and does not initiate connections to clients or any other systems. The truly paranoid may wish to establish firewall rules that prevent the yule server from establishing any connections to systems outside the organization.

Connection Authentication

Connections are authenticated using a password mechanism and TCP Wrappers. The password mechanism can be easily tested by removing or altering a valid client entry at the end of the yule configuration. The use of libwrap is tested by not listing the client's IP address for the yule service in /etc/hosts.allow.

The client will attempt to connect and have errors.

```
ERROR  :  [2004-09-10T10:36:05-0400] msg=<Invalid connection state>,
 expect=<INIT>, received=<\000\007\054\000>
A877D8401A3B1D8B33F81E3D4358C7AD50E1CAD2145C7056[2004-09-10T10:36:05-0400]
ERROR  :  [2004-09-10T10:36:06-0400] msg=<Session key negotiation failed>
6AB016CEFF553FF0C30235E8D4569AC6025F70C343821DFA
ERROR  :  [2004-09-10T10:36:06-0400] msg=<Service failure>,
 service=<log server>, obj=<10.0.0.6>
4DBCB065559B0254B8F9FA087621E01410B72C53F94F9A3E
```

The yule server will also log errors.

```
ERROR  :  [2004-09-10T10:36:05-0400] msg=<Refused connection from
 dev.giac.org>, subroutine=<libwrap>
C5B59E7C627DFCDC798BA65B95896626596B7F5598695D17
ERROR  :  [2004-09-10T10:36:10-0400] msg=<Refused connection from
 dev.giac.org>, subroutine=<libwrap>
0D98ECE297C679B25462F0E85D444112C6F3111A1EFFFFBF
ERROR  :  [2004-09-10T10:36:14-0400] msg=<Refused connection from
 dev.giac.org>, subroutine=<libwrap>
54CD3E5BE1E1534C37CFB7C54B1C4DC488AE04CC59C22DD6
```

Add the correct yule service line to /etc/hosts.allow and restart samhain. No errors will in the client log file (/var/samhain/samhain.log) or the server log file (/yule/var/yule/yule.log). The server will now be logging messages from the client.

Open Ports on the Yule Server

The yule server should only listen on port 49777. The analysis of firewall logs seems to verify this, but this can also be examined on the yule server.

```
# netstat -P udp
# netstat -P tcp

TCP: IPv4
   Local Address     Remote Address    Swind Send-Q Rwind Recv-Q  State
---------------- ------------------- ----- ------ ----- ------ -------
dns1.ssh          holmesw4.4693       64671      0 48340      0 ESTABLISHED
dns1.ssh          holmesw4.4701       65467      0 48952      0 ESTABLISHED
dns1.ssh          dogbert.35020       49680      0 49152      0 ESTABLISHED
dns1.ssh          dogbert.35021       49680      0 49088      0 ESTABLISHED
dns1.ssh          medusa.44964        49680     51 48560      0 ESTABLISHED
```

This output reveals that there are no udp connections in progress, and the only active tcp connections are for ssh. Using `netstat -a` does reveal that the system is listening on port 49777.

```
TCP: IPv4
   Local Address        Remote Address    Swind Send-Q Rwind Recv-Q  State
-------------------- -------------------- ----- ------ ----- ------ -------
      *.49777              *.*               0      0 49152      0 LISTEN
```

Lsof can be used to find the open files and ports that are used by the yule server. Find the process ID of yule and use the PID as an argument to `lsof -p`.

```
# lsof -p 12979
COMMAND   PID USER   FD   TYPE        DEVICE SIZE/OFF   NODE NAME
yule    12979 yule  cwd   VDIR          85,2      512 103729 /yule
yule    12979 yule  rtd   VDIR          85,2      512 103729 /yule
yule    12979 yule  txt   VREG          85,2   301984 103738
 /yule/opt/yule/bin/yule
```

```
yule    12979 yule  txt   VREG          85,2   866456  25287
 /usr/lib/libc.so.1
yule    12979 yule  txt   VREG          85,2    22552  25072
 /usr/lib/nss_dns.so.1
yule    12979 yule  txt   VREG          85,2    35060  25073
 /usr/lib/nss_files.so.1
yule    12979 yule  txt   VREG          85,2    16768  27833
 /usr/platform/sun4u/lib/libc_psr.so.1
yule    12979 yule  txt   VREG          85,2   316436  25358
 /usr/lib/libresolv.so.2
yule    12979 yule  txt   VREG          85,2   743856  25039
 /usr/lib/libnsl.so.1
yule    12979 yule  txt   VREG          85,2    21676  25036
 /usr/lib/libmp.so.2
yule    12979 yule  txt   VREG          85,2    58504  25060
 /usr/lib/libsocket.so.1
yule    12979 yule  txt   VREG          85,2     3984  25013
 /usr/lib/libdl.so.1
yule    12979 yule  txt   VREG          85,2      104   4424
 /var/ld/ld.config
yule    12979 yule  txt   VREG          85,2   192000  24897
 /usr/lib/ld.so.1
yule    12979 yule    0u  VCHR          13,2      0t0  68261
 /devices/pseudo/mm@0:null
yule    12979 yule    1u  VCHR          13,2      0t0  68261
 /devices/pseudo/mm@0:null
yule    12979 yule    2u  VCHR          13,2      0t0  68261
 /devices/pseudo/mm@0:null
yule    12979 yule    3u  IPv4 0x30002542f38      0t0    TCP *:49777 (LISTEN)
```

Summary

A client/server installation was done to provide file integrity checking for a web server. Samhain improves on other file integrity checkers by signing its database and configuration files in order to prevent unauthorized modification. Samhain also provides the ability to log to a central server that can also provide the client configuration and database files. Samhain even goes beyond traditional integrity checkers by including the ability to run in stealth mode, check file system mount options, and detect unauthorized SUID/SGID files.

Unfortunately, all of these features have substantial complexity. While the complexity probably will not be a security risk, there is the chance that the programs will not be fully utilized without sufficient local documentation. Features such as the stealth installation were not implemented due to the complexity.

Other security risks and mitigations were examined and discussed. Following a detailed discussion on the installation, the maintenance procedures necessary for running Samhain were discussed. Testing and qualification of the installation was done to see that the system works as intended. This testing concluded that Samhain does work as advertised. The effort required for

the installation and configuration, plus this testing, made it very clear that a successful attack on the Samhain software would be very difficult. A successful attack would require compromising all systems running samhain and the yule server, plus discovering the passwords and pass phrases that are using for authentication and signing.

Further Research and Work

This installation concerned protecting a single web server using the samhain client while utilizing a yule server for logging and storage of the configuration and database files. Within our environment, the following work efforts should build on this Samhain installation.

- Other Samhain clients should be installed. This can be done with little effort since new client installations will closely mirror the installation done for the web server.
- Logging can still be improved. The current system emails critical alerts to the system administrator with only a few additional messages of limited value. Further tuning of the logging options can yield slight improvements. Samhain also allows an external program to be called to facilitate logging and formatting. Programs like Logwatch or Swatch could also be used to better control logging. For example, Logwatch could email all administrators only when a file modification occurs. But, all of the other alerts from Samhain could be sent to just one primary Samhain administrator.
- The Samhain log file is verbose. Every other line is a signature used to verify log file entries. The file also contains timestamps just to determine that clients are alive. It should be a fairly simple effort to create a Perl program that reformats the messages of primary interest.
- Samhain does have a web-based management tool named Beltane. Implementation might improve management and reviewing of alerts. However, the installation does require a web server and SQL server to be running on the management host. This presents a possible reduction in security in order to improvement the ease of management.

Samhain could be further improved by simplifying the installation. Numerous options must be specified a compile time. If you get it wrong, then you start over. Moving more of these options into configuration files would make installing Samhain much easier.

Linux and FreeBSD installations of Samhain are capable of checking kernel modules. Extending that ability to Solaris would be nice and might improve the popularity of Samhain.

Samhain will search for new SUID and SGID files. This can detection the installation of new backdoors since samhain will know about all authorized SUID/SGID files. However, the I/O required for checking all file systems is substantial. Our systems have several terabytes of documents. All of these file systems are mounted with the *nosuid* option, and samhain should know that these file systems do not need checking.

Since our site hosts numerous web sites, there is a concern about web site defacement. Simple Perl scripts are currently used to check the home pages and popular pages for defacement by looking for foul language and other key words. Extending Samhain to have this ability would be an improvement over the Perl scripts which can be easily defeated once found. Samhain already supports an interface that allows for implementing modules written in C. Therefore, it should not require major changes to the Samhain source in order to implement checking for web site defacement.

Literature Review

The author of Samhain, Rainer Wichmann, has written a comprehensive user manual[20] This manual is extremely thorough for open source software. However, samhain is a very complex piece of software. Even though the manual has good coverage of every configuration option for compilation and the configuration file format, the manual lacks complete examples. Extensive examples would be a great help given that the software can be configured in so many different ways. These deficiencies are offset by several of the HOWTO guides that are available in the documentation section[21] of the Samhain web site.

A search of GIAC posted practicals was done for Samhain as there was no desire to duplicate the work of others. Seven practicals mention Samhain. Six of the references are generic references to Samhain while discussing either file integrity checkers or Samhain's ability to check for Linux rootkits. A single paper by Del Armstrong uses Samhain as an example in a paper titled "An Introduction to File Integrity Checking On Unix Systems" [22]. That paper is roughly divided into ten pages of discussion of file integrity checkers as a whole, twenty pages on the installation of a client-server setup, and 35 pages output and configuration files from the installation. This paper treats Samhain as a service added to a system and discusses the security impact of adding such a service.

An overwhelming number of references can be found in either Yahoo or Google by searching for "file integrity checkers."

Websites for alternative integrity checkers include the commercial products GFI LANguard [23], Data Sentinel [24], Sentinel [25], Veracity [26], Tripwire [27], and free products such as AIDE [28], FCHECK [29], SigTree [30], ViperDB [31], Triplight (also known as Claymore) [32], Tripwire ASR [33], Jverify [34], and mtree [35].

[20] Wichmann, Rainer. *Samhain User Manual*. 2004 http://www.la-samhna.de/samhain/manual/ (02 Aug 2004).

[21] Wichmann, Rainer. *Samhain Labs Documentation*. http://www.la-samhna.de/samhain/s_documentation.html. (02 Aug 2004).

[22] Armstrong, Del. *An Introduction to File Integrity Checking On Unix Systems*. 20 May 2003. http://www.giac.org/practical/GCUX/Del_Armstrong_GCUX.pdf. (02 Aug 2004).

[23] GFI LANguard. http://www.gfi.com. (30 Aug 2004).

[24] Data Sentinel. http://www.ionx.co.uk/html/products/data_sentinel/. (30 Aug 2004).

[25] Sentinel. http://www.runtimeware.com/?page=p_sentinel2. (30 Aug 2004).

[26] Veracity. http://www.rocksoft.com/veracity/. (30 Aug 2004)

[27] Tripwire. http://www.tripwire.com/products/servers/. (30 Aug 2004).

[28] AIDE. http://www.cs.tut.fi/~rammer/aide.html. (30 Aug 2004).

[29] FCHECK. http://www.geocities.com/fcheck2000/fcheck.html. (30 Aug 2004).

[30] SigTree. http://www.discord.org/~lippard/software/software.html. (30 Aug 2004).

[31] ViperDB. http://www.resentment.org/projects/viperdb/index.html. (30 Aug 2004).

[32] Triplight. http://freshmeat.net/projects/triplight/. (30 Aug 2004).

[33] Tripwire ASR. http://www.tripwire.com/products/tripwire_asr/index.cfm. (30 Aug 2004).

[34] Jverify. http://sourceforge.net/projects/jverify/. (30 Aug 2004).

[35] Mtree. http://www.free-x.ch/pub/mtree-tripwire.html. (30 Aug 2004).

References

1. Dasan, Vasanthan., Noordergraaf, Alex., and Ordorica, Lou. *The Solaris Fingerprint Database - A Security Tool for Solaris Operating Environment Files*. May 2001. http://www.sun.com/blueprints/0501/Fingerprint.pdf. (12 Sep 2004).

2. *Solaris Fingerprint Database: An Identification Tool for Solaris Software and Files*. http://sunsolve.sun.com/pub-cgi/show.pl?target=content/content7 (05 Sep 2004).

3. Wichmann, Rainer. *Samhain User Manual*. 2004. http://www.la-samhna.de/samhain/manual/ (02 Aug 2004).

4. Northcutt, Stephen., Zeltser, Lenny., Winters, Scott., Frederick, Karen Kent, and Ritchey, Ronald W. *Inside Network Perimeter Security*. Indianapolis: New Riders Publishing. 2003.

5. GNU Stow. 2004. http://www.gnu.org/software/stow/stow.html. (04 Aug 2004).

6. GnuPG. 2004. http://www.gnupg.org (02 Aug 2004).

7. GMP Library. 2004. http://www.swox.com/gmp/. (02 Aug 2004).

8. Limnocelli, Thomas., & Hogan, Christine. 2002. *The Practice of System and Network Adminstration*. Upper Saddle River, NJ: Addison Wesley, 2001.

9. Mellen, D., Garcia, J., Keegan, J., Gauthier, M., Royds, B., Gladstone, E., & Clausing, J. *Securing Solaris 8 & 9 Using the Center for Internet Security Benchmark*. SANS Press, 2003.

10. Wichmann, Rainer. Samhain. 2004. http://www.la-samhna.de/samhain/. (02 Aug 2004).

11. Wichmann, Rainer. *HOWTO: Samhain client/server: What can go wrong, and can you fix it?*. 2004. http://www.la-samhna.de/samhain/HOWTO-client+server-troubleshooting.html (29 Aug 2004).

12. Wichmann, Rainer. *HOWTO: Setting up a client/server samhain system*. 2004. http://www.la-samhna.de/samhain/HOWTO-client+server.html (29 Aug 2004).

13. Wichmann, Rainer. *HOWTO: Using samhain with GnuPG*. 2004. http://www.la-samhna.de/samhain/HOWTO-samhain+GnuPG.html (29 Aug 2004).

14. Sudo. 2004. http://www.courtesan.com/sudo/. (14 Aug 2004).

15. Wu, Tom. *The Stanford SRP Authentication Project*. 2004. http://srp.stanford.edu/ (04 Sep 2004).

Appendix 1. Yule Startup Script

```
#!/bin/sh

# Change location so that yule can run in chroot jail.
###SBINDIR=/opt/yule/bin
SBINDIR=/yule/opt/yule/bin
NAME=yule

if [ ! -f ${SBINDIR}/${NAME} ]; then
        exit 0
fi


log_stat_msg () {
case "$1" in
0)
echo "Service $NAME: Running";
break;
;;
1)
echo "Service $NAME: Stopped and /var/run pid file exists";
break;
;;
3)
echo "Service $NAME: Stopped";
break;
;;
*)
echo "Service $NAME: Status unknown";
break;
;;
esac
}


case "$1" in
    start)
        echo "${NAME} starting."
        ${SBINDIR}/${NAME} --chroot=/yule $1
        ;;
    stop)
        echo "${NAME} stopping."
        ${SBINDIR}/${NAME} $1
        ;;
    restart)
        echo "${NAME} restarting."
```

```
        ${SBINDIR}/${NAME} $1
        ;;
    reload|force-reload)
        echo "${NAME} reloading."
        ${SBINDIR}/${NAME} $1
        ;;
    status)
        ${SBINDIR}/${NAME}  $1
        ERRNUM=$?
        log_stat_msg ${ERRNUM}
        exit ${ERRNUM}
        ;;
    *)
        echo "Usage: $0 {start|stop|restart|reload|status}"
        exit 1
        ;;
esac

# Modification required for chroot
# Commands must be embedded in the case  statements above
###${SBINDIR}/${NAME}  $1

status=$?


if [ $status != 0 ]; then
    echo $status
    exit 1
fi

case "$1" in
    stop)
        if test -f /var/yule/run/yule.pid; then
            /bin/rm -f /var/yule/run/yule.pid
        fi
        ;;
    *)
        exit 0
        ;;
esac

exit 0
```

Appendix 2. Web.giac.org Client Configuration

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1
NotDashEscaped: You need GnuPG to verify this message

######################################################################
#
# SOLARIS Configuration file  for samhain.
#
# Based on a contribution by Sean Boran (sean [at] boran d.o.t com)
#
# HISTORY:
#  16.Aug.03 rw add plenty of comments
#  24.Jun.02 rw remove linux stuff, clean up a bit
#  06.Jun.02 sb <3>, add LOTS more Solaris stuff. Also and comment at bottom
#               of this file.
#  03.Jun.02 sb Separate Linux & Solaris
#  24.Apr.02 sb Use Samhain v.15 template and tune for Solaris.
#
# To do: logs /var/adm/messages and /var/cron/log are
# pruned weekly.
######################################################################
#
# -- empty lines and lines starting with '#', ';' or '//' are ignored
# -- boolean options can be Yes/No or True/False or 1/0
# -- you can PGP clearsign this file -- samhain will check (if compiled
#    with support) or otherwise ignore the signature
# -- CHECK mail address
#
# To each log facility, you can assign a threshold severity. Only
# reports with at least the threshold severity will be logged
# to the respective facility (even further below).
#
######################################################################
# SETUP for file system checking:
# (i)   There are several policies, each has its own section. Put files
#       into the section for the appropriate policy (see below).
# (ii)  Section [EventSeverity]:
#       To each policy, you can assign a severity (further below).
# (iii) Section [Log]:
#       To each log facility, you can assign a threshold severity. Only
#       reports with at least the threshold severity will be logged
#       to the respective facility (even further below).
######################################################################

######################################################################
```

```
#
# Files are defined with: file = /absolute/path
#
# Directories are defined with:                   dir = /absolute/path
# or with an optional recursion depth (N <= 99): dir = N/absolute/path
#
# Directory inodes are checked. If you only want to check files
# in a directory, but not the directory inode itself, use (e.g.):
#
# [ReadOnly]
# dir = /some/directory
# [IgnoreAll]
# file = /some/directory
#
# You can use shell-style globbing patterns, like: file = /path/foo*
#
##############################################################################

[Misc]
##
## Add or subtract tests from the policies
## - if you want to change their definitions,
##   you need to do that before using the policies
##
# RedefReadOnly = (no default)
# RedefAttributes=(no default)
# RedefLogFiles=(no default)
# RedefGrowingLogFiles=(no default)
# RedefIgnoreAll=(no default)
# RedefIgnoreNone=(no default)
# RedefUser0=(no default)
# RedefUser1=(no default)


[Attributes]
##
## for these files, only changes in permissions and ownership are checked
##

#file=/etc/ssh/ssh_random_seed
file=/etc/resolv.conf
# There are files in /etc that might change, thus changing the directory
# timestamps. Put it here as 'file', and in the ReadOnly section as 'dir'.
file=/etc

#file=/etc/skip/randseed
file=/etc/cron.d/FIFO
```

```
file=/etc/devlink.tab
file=/etc/.syslog_door
file=/etc/syslog.pid
file=/etc/mnttab
file=/etc/cron.d
file=/etc/mail
file=/etc/inet
##dir=/secure/tmp
dir=/etc/sysevent
#dir=/usr/tmp
#dir=/usr/aset/tmp
#dir=/usr/oasys/tmp
#dir=/var/spool/lp/tmp
#dir=/var/tmp
#dir=/var/dt/tmp
#dir=/tmp
#dir=/etc/osa

[LogFiles]
##
## for these files, changes in signature, timestamps, and size are ignored
##
#file=/var/run/utmp
#file=/etc/motd
file=/var/cron/log
file=/var/adm/wtmpx
file=/var/adm/wtmp
file=/var/adm/utmpx
file=/var/adm/lastlog

[GrowingLogFiles]
##
## for these files, changes in signature, timestamps, and increase in size
##                  are ignored
##

file=/var/adm/messages



[IgnoreAll]
##
## for these files, no modifications are reported
##

file=/etc/utmppipe
file=/usr/dt/bin/ttsnoop
```

```
file=/dev/mem
dir=/etc/saf
dir=/usr/share/man
dir=/usr/share/lib/terminfo
dir=/usr/demo
dir=/usr/lib/adb
dir=/usr/local/man
dir=/usr/local/doc
dir=/usr/dt/share/man
dir=/usr/openwin/lib/locale
dir=/usr/openwin/share/man
dir=/usr/openwin/share/src



[IgnoreNone]
##
## for these files, all modifications (even access time) are reported
##    - you may create some interesting-looking file (like /etc/safe_passwd),
##      just to watch whether someone will access it ...
##
#/etc/seantest

[ReadOnly]
##
## for these files, only access time is ignored
##
#dir=/usr/bin
#dir=/usr/sbin
#dir=/usr/lib

# SuSE (old) has the boot init scripts in /sbin/init.d/*,
# so we go 3 levels deep
###dir=3/sbin

# RedHat and Debian have the bootinit scripts in /etc/init.d/* or /etc/rc.d/*,
#        so we go 3 levels deep there too
dir=3/etc

# Various directories / files that may include / be SUID/SGID binaries
#
##dir=/usr/openwin/bin
##dir=/usr/dt/bin

##dir=/root

# Critical devices
```

```
#file=/dev/dsk
#file=/dev/rdsk
#file=/dev/null
#file=/dev/zero

# Root's environment
file=/.profile
file=/.kshrc
# OS
dir=/kernel
# Configuration
dir=/etc/default
file=/etc/dfs/dfstab
file=/etc/group
file=/etc/passwd
file=/etc/hosts
dir=/etc/inet
dir=/etc/init.d
dir=/etc/rc0.d
dir=/etc/rc2.d
dir=/etc/rc3.d
file=/etc/profile
file=/etc/remote
file=/etc/rpc
file=/etc/system
file=/etc/ttydefs
file=/etc/ttysrch
# setuid/setgid root programs
file=/usr/lib/lp/bin/netpr
file=/usr/lib/fs/ufs/quota
file=/usr/lib/fs/ufs/ufsdump
file=/usr/lib/fs/ufs/ufsrestore
file=/usr/lib/pt_chmod
file=/usr/lib/utmp_update
file=/usr/lib/acct/accton
file=/usr/lib/sendmail
file=/usr/openwin/bin/xlock
file=/usr/openwin/lib/mkcookie
file=/usr/dt/bin/dtaction
file=/usr/dt/bin/dtappgather
file=/usr/dt/bin/dtprintinfo
file=/usr/dt/bin/dtsession
file=/usr/bin/at
file=/usr/bin/atq
file=/usr/bin/atrm
file=/usr/bin/crontab
```

```
file=/usr/bin/eject
file=/usr/bin/fdformat
file=/usr/bin/login
file=/usr/bin/newgrp
file=/usr/bin/passwd
file=/usr/bin/rcp
file=/usr/bin/rdist
file=/usr/bin/rlogin
file=/usr/bin/rsh
file=/usr/bin/su
file=/usr/bin/tip
file=/usr/bin/admintool
file=/usr/bin/cancel
file=/usr/bin/lp
file=/usr/bin/lpset
file=/usr/bin/lpstat
file=/usr/bin/volcheck
file=/usr/bin/volrmmount
file=/usr/sbin/allocate
file=/usr/sbin/mkdevalloc
file=/usr/sbin/mkdevmaps
file=/usr/sbin/ping
file=/usr/sbin/sacadm
file=/usr/sbin/traceroute
file=/usr/sbin/deallocate
file=/usr/sbin/list_devices
file=/usr/sbin/lpmove
#
file=/usr/ucb/sparcv7/ps
file=/usr/bin/sparcv7/ps
file=/usr/bin/sparcv7/uptime
file=/usr/bin/sparcv7/w
file=/usr/sbin/sparcv7/whodo
#
file=/usr/ucb/sparcv9/ps
file=/usr/bin/sparcv9/ps
file=/usr/bin/sparcv9/uptime
file=/usr/bin/sparcv9/w
file=/usr/sbin/sparcv9/whodo
# important programs
file=/usr/bin/cat
file=/usr/bin/du
file=/usr/bin/last
file=/usr/bin/ls
file=/usr/bin/more
file=/usr/bin/netstat
file=/usr/bin/ps
```

```
file=/usr/bin/vi
file=/usr/bin/w
file=/usr/bin/who
file=/usr/sbin/df
file=/usr/local/bin/lsof
# important daemons
file=/usr/sbin/in.*
# We don't run these, but check them anyway
file=/usr/sbin/rpc.*
file=/usr/dt/bin/rpc.ttdbserverd
file=/usr/lib/print/in.lpd
# Apache Web Server files
dir=/web/bin
dir=/web/lib
# A few libraries
file=/usr/lib/ld.so
file=/usr/lib/ld.so.1
file=/usr/lib/libc.*
file=/usr/lib/libc2.*
file=/usr/lib/libresolv*
file=/usr/lib/libmalloc*
file=/usr/lib/libnls*
file=/usr/lib/libpam*
file=/usr/lib/libcrypt*
file=/usr/lib/libsocket*
file=/usr/lib/libmp*
file=/usr/lib/libcmd*
file=/usr/lib/libkstat*
# Samhain
dir=/opt/samhain/bin

[User0]
[User1]
## User0 and User1 are sections for files/dirs with user-definable checking
## (see the manual)

[EventSeverity]
##
## Here you can assign severities to policy violations.
## If this severity exceeds the treshold of a log facility (see below),
## a policy violation will be logged to that facility.

# Severity for verification failures.
#
# SeverityReadOnly=crit
# SeverityLogFiles=crit
# SeverityGrowingLogs=crit
```

```
# SeverityIgnoreNone=crit
# SeverityAttributes=crit
# SeverityUser0=crit
# SeverityUser1=crit

# We have a file in IgnoreAll that might or might not be present.
# Setting the severity to 'info' prevents messages about deleted/new file.
#
# SeverityIgnoreAll=crit
SeverityIgnoreAll=info

# Files : file access problems
# SeverityFiles=crit

# Dirs  : directory access problems
# SeverityDirs=crit

# Names : suspect (non-printable) characters in a pathname
# SeverityNames=crit

[Log]
##
## Switch on/OFF log facilities and set their threshold severity
##
## Values: debug, info, notice, warn, mark, err, crit, alert, none.
## 'mark' is used for timestamps.
##
## Use 'none' to SWITCH OFF a log facility
##
## By default, everything equal to and above the threshold is logged.
## The specifiers '*', '!', and '=' are interpreted as
## 'all', 'all but', and 'only', respectively (like syslogd(8) does,
## at least on Linux). Examples:
## MailSeverity=*
## MailSeverity=!warn
## MailSeverity==crit

## E-mail
##
# MailSeverity=none
MailSeverity=crit

## Console
##
# PrintSeverity=info

## Logfile
```

```
##
# LogSeverity=mark
LogSeverity=warn

## Syslog
##
# SyslogSeverity=none

## Remote server (yule)
##
# ExportSeverity=none
ExportSeverity=warn

## External script or program
##
# ExternalSeverity = none

## Logging to a database
##
# DatabaseSeverity = none


#########################################################
#
# Optional modules
#
#########################################################

# [SuidCheck]
##
## --- Check the filesystem for SUID/SGID binaries
##

## Switch on
#
# SuidCheckActive = yes

## Interval for check (seconds)
#
# SuidCheckInterval = 7200

## Alternative: crontab-like schedule
#
# SuidCheckSchedule = NULL

## Directory to exclude
#
```

```
# SuidCheckExclude = NULL

## Limit on files per second (0 == no limit)
#
# SuidCheckFps = 0

## Alternative: yield after every file
#
# SuidCheckYield = no

## Severity of a detection
#
# SeveritySuidCheck = crit

## Quarantine SUID/SGID files if found
#
# SuidCheckQuarantineFiles = yes

## Method for Quarantining files:
#  0 - Delete the file.
#  1 - Remove SUID/SGID permissions from file.
#  2 - Move SUID/SGID file to quarantine dir.
#
# SuidCheckQuarantineMethod = 0

## For method 1 and 3, really delete instead of truncating
#

# [Utmp]
##
## --- Logging of login/logout events
##

## Switch on/off
#
# LoginCheckActive = True

## Severity for logins, multiple logins, logouts
#
# SeverityLogin=info
# SeverityLoginMulti=warn
# SeverityLogout=info

## Interval for login/logout checks
#
# LoginCheckInterval = 300
```

```
# [Database]
##
## --- Logging to a relational database
##

## Database name
#
# SetDBName = samhain

## Database table
#
# SetDBTable = log

## Database user
#
# SetDBUser = samhain

## Database password
#
# SetDBPassword = (default: none)

## Database host
#
# SetDBHost = localhost

## Log the server timestamp for received messages
#
# SetDBServerTstamp = True

## Use a persistent connection
#
# UsePersistent = True

# [External]
##
## Interface to call external scripts/programs for logging
##

## The absolute path to the command
## - Each invocation of this directive will end the definition of the
##   preceding command, and start the definition of
##   an additional, new command
#
# OpenCommand = (no default)

## Type (log or rv)
```

```
## - log for log messages, srv for messages received by the server
#
# SetType = log

## The command (full command line) to execute
#
# SetCommandLine = (no default)

## The environment (KEY=value; repeat for more)
#
# SetEnviron = TZ=(your timezone)

## The TIGER192 checksum (optional)
#
# SetChecksum = (no default)

## User who runs the command
#
# SetCredentials = (default: samhain process uid)

## Words not allowed in message
#
# SetFilterNot = (none)

## Words required (ALL of them)
#
# SetFilterAnd = (none)

## Words required (at least one)
#
# SetFilterOr = (none)

## Deadtime between consecutive calls
#
# SetDeadtime = 0

## Add default environment (HOME, PATH, SHELL)
#
# SetDefault = no




#########################################################
#
# Miscellaneous configuration options
#
```

```
#########################################################

[Misc]

## whether to become a daemon process
## (this is not honoured on database initialisation)
#
# Daemon = no
Daemon = yes

## whether to test signature of files (init/check/none)
## - if 'none', then we have to decide this on the command line -
#
# ChecksumTest = none
ChecksumTest=check

## Set nice level (-19 to 19, see 'man nice'),
## and I/O limit (kilobytes per second; 0 == off)
## to reduce load on host.
#
# SetNiceLevel = 0
# SetIOLimit = 0

## The version string to embed in file signature databases
#
# VersionString = NULL

## Interval between time stamp messages
#
# SetLoopTime = 60
SetLoopTime = 600

## Interval between file checks
#
# SetFileCheckTime = 600
SetFileCheckTime = 7200

## Alternative: crontab-like schedule
#
# FileCheckScheduleOne = NULL

## Alternative: crontab-like schedule(2)
#
# FileCheckScheduleTwo = NULL

## Report only once on modified fles
## Setting this to 'FALSE' will generate a report for any policy
```

```
## violation (old and new ones) each time the daemon checks the file system.
#
# ReportOnlyOnce = True

## Report in full detail
#
# ReportFullDetail = False

## Report file timestamps in local time rather than GMT
#
# UseLocalTime = No

## The console device (can also be a file or named pipe)
## - There are two console devices. Accordingly, you can use
##   this directive a second time to set the second console device.
##   If you have not defined the second device at compile time,
##   and you don't want to use it, then:
##   setting it to /dev/null is less effective than just leaving
##   it alone (setting to /dev/null will waste time by opening
##   /dev/null and writing to it)
#
# SetConsole = /dev/console

## Activate the SysV IPC message queue
#
# MessageQueueActive = False


## If false, skip reverse lookup when connecting to a host known
## by name rather than IP address (i.e. trust the DNS)
#
# SetReverseLookup = True

## --- E-Mail ---

# Only highest-level (alert) reports will be mailed immediately,
# others will be queued. Here you can define, when the queue will
# be flushed (Note: the queue is automatically flushed after
# completing a file check).
#
# SetMailTime = 86400

## Maximum number of mails to queue
#
# SetMailNum = 10

## Recipient (max. 8)
```

```
#
# SetMailAddress=root@localhost
SetMailAddress=holmesw@giac.org

## Mail relay (IP address)
#
# SetMailRelay = NULL

## Custom subject format
#
# MailSubject = NULL
MailSubject = "%H %T %S"

## --- end E-Mail ---


## Path to the executable. If set, will be checksummed after startup
## and before exit.
#
# SamhainPath = (no default)


## The IP address of the log server
#
# SetLogServer = (default: compiled-in)
SetLogServer = 10.0.0.6

## The IP address of the time server
#
# SetTimeServer = (default: compiled-in)

## Trusted Users (comma delimited list of user names)
#
# TrustedUser = (no default; this adds to the compiled-in list)

## Path to the file signature database
#
# SetDatabasePath = (default: compiled-in)

## Path to the log file
#
# SetLogfilePath = (default: compiled-in)

## Path to the PID file
#
# SetLockPath = (default: compiled-in)
```

```
## The digest/checksum/hash algorithm
#
# DigestAlgo = TIGER192


## Custom format for message header.
## CAREFUL if you use XML logfile format.
##
## %S severity
## %T timestamp
## %C class
##
## %F source file
## %L source line
#
# MessageHeader="%S %T "


## Don't log path to config/database file on startup
#
# HideSetup = False

## The syslog facility, if you log to syslog
#
# SyslogFacility = LOG_AUTHPRIV
SyslogFacility=LOG_LOCAL2

## The message authentication method
## - If you change this, you *must* change it
##   on client *and* server
#
# MACType = HMAC-TIGER


# everything below is ignored
[EOF]

#########################################################################
# This would be the proper syntax for parts that should only be
#    included for certain hosts.
# You may enclose anything in a @HOSTNAME/@end bracket, as long as the
#    result still has the proper syntax for the config file.
# You may have any number of @HOSTNAME/@end brackets.
# HOSTNAME should be the fully qualified 'official' name
#    (e.g. 'nixon.watergate.com', not 'nixon'), no aliases.
#    No IP number - except if samhain cannot determine the
```

```
#    fully qualified hostname.
#
# @HOSTNAME
# file=/foo/bar
# @end
#
# These are two examples for conditional inclusion/exclusion
# of a machine based on the output from 'uname -srm'
# $Linux:2.*.7:i666
# file=/foo/bar3
# $end
#
# !$Linux:2.*.7:i686
# file=/foo/bar2
# $end
#
##############################################################################
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.2.6 (SunOS)

iD8DBQFBS2zSxIz339qtZBwRApC0AJ9Fu04YJxCiw3qRRxUQLYf5dSPDEQCfaEId
DQUu1DkecWLbkUrqjzVUQGM=
=nHCG
-----END PGP SIGNATURE-----
```

Appendix 3. Yule Server Configuration

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1
NotDashEscaped: You need GnuPG to verify this message

######################################################################
#
# Configuration file template for yule.
#
######################################################################
#
# NOTE: This is a log server-only configuration file TEMPLATE.
#
# NOTE: The log server ('yule') will look for THAT configuration file
#       that has been defined at compile time with the configure option
#         ./configure --with-config-file=FILE
#       The default is "/usr/local/etc/.samhainrc" (NOT "yulerc").
#
######################################################################
#
# -- empty lines and lines starting with '#', ';' or '//' are ignored
# -- you can PGP clearsign this file -- samhain will check (if compiled
#    with support) or otherwise ignore the signature
# -- CHECK mail address
#
# To each log facility, you can assign a threshold severity. Only
# reports with at least the threshold severity will be logged
# to the respective facility (even further below).
#
######################################################################

[Log]
##
## Switch on/OFF log facilities and set their threshold severity
##
## Values: debug, info, notice, warn, mark, err, crit, alert, none.
## 'mark' is used for timestamps.
##
##
## Use 'none' to SWITCH OFF a log facility
##
## By default, everything equal to and above the threshold is logged.
## The specifiers '*', '!', and '=' are interpreted as
## 'all', 'all but', and 'only', respectively (like syslogd(8) does,
## at least on Linux). Examples:
## MailSeverity=*
```

```
## MailSeverity=!warn
## MailSeverity==crit

## E-mail
##
# MailSeverity=none
MailSeverity=err

## Console
##
# PrintSeverity=info

## Logfile
##
# LogSeverity=none
LogSeverity=warn

## Syslog
##
# SyslogSeverity=none

## External script or program
##
# ExternalSeverity = none

## Logging to a database
##
# DatabaseSeverity = none

# [Database]
##
## --- Logging to a relational database
##

## Database name
#
# SetDBName = samhain

## Database table
#
# SetDBTable = log

## Database user
#
# SetDBUser = samhain

## Database password
```

```
#
# SetDBPassword = (default: none)

## Database host
#
# SetDBHost = localhost

## Log the server timestamp for received messages
#
###SetDBServerTstamp = True

## Use a persistent connection
#
###UsePersistent = True

# [External]
##
## Interface to call external scripts/programs for logging
##

## The absolute path to the command
## - Each invocation of this directive will end the definition of the
##   preceding command, and start the definition of
##   an additional, new command
#
# OpenCommand = (no default)

## Type (log or rv)
## - log for log messages, srv for messages received by the server
#
# SetType = log

## The command (full command line) to execute
#
# SetCommandLine = (no default)

## The environment (KEY=value; repeat for more)
#
# SetEnviron = TZ=(your timezone)

## The TIGER192 checksum (optional)
#
# SetChecksum = (no default)

## User who runs the command
#
# SetCredentials = (default: samhain process uid)
```

```
## Words not allowed in message
#
# SetFilterNot = (none)

## Words required (ALL of them)
#
# SetFilterAnd = (none)

## Words required (at least one)
#
# SetFilterOr = (none)

## Deadtime between consecutive calls
#
# SetDeadtime = 0

## Add default environment (HOME, PATH, SHELL)
#
# SetDefault = no

########################################################
#
# Miscellaneous configuration options
#
########################################################

[Misc]

## whether to become a daemon process
## (this is not honoured on database initialisation)
#
# Daemon = no
Daemon = yes

[Misc]
# whether to become a daemon process
Daemon=yes

## Chroot configuration
SetChrootDir = /yule

## Interval between time stamp messages
#
# SetLoopTime = 60
SetLoopTime = 600
```

```
## The maximum time between client messages (seconds)
## This allows the server to flag clients that have exceeded
## the timeout limits; i.e. might have died for some reason.
#
# SetClientTimeLimit = 86400
SetClientTimeLimit = 7200

## Use client address as known to the communication layer (might be
## incorrect if the client is behind NAT). The default is to use
## the client name as claimed by the client, and verify it against
## the former (might be incorrect if the client has several
## interfaces, and its hostname resolves to the wrong interface).
#
# SetClientFromAccept = False

## If SetClientFromAccept is False (default), severity of a
## failure to resolve the hostname claimed by the client
## to the IP address of the socket peer.
#
# SeverityLookup = crit

## The console device (can also be a file or named pipe)
## - There are two console devices. Accordingly, you can use
##   this directive a second time to set the second console device.
##   If you have not defined the second device at compile time,
##   and you don't want to use it, then:
##   setting it to /dev/null is less effective than just leaving
##   it alone (setting to /dev/null will waste time by opening
##   /dev/null and writing to it)
#
# SetConsole = /dev/console

## Use separate logfiles for individual clients
#
# UseSeparateLogs = False

## Enable listening on port 514/udp for logging of remote syslog
## messages (if optionally compiled with support for this)
#
# SetUDPActive = False


## Activate the SysV IPC message queue
#
# MessageQueueActive = False
```

```
## If false, skip reverse lookup when connecting to a host known
## by name rather than IP address (i.e. trust the DNS)
#
# SetReverseLookup = True

## If true, open a Unix domain socket to listen for commands that should
## be passed to clients upon next connection. Only works on systems
## that support passing of peer credentials (for authentication) via sockets.
## Use yulectl to access the socket.
#
# SetUseSocket = False

## The UID of the user that is allowed to pass commands to the server
## via the Unix domain socket.
#
# SetSocketAllowUid = 0

## --- E-Mail ---

# Only highest-level (alert) reports will be mailed immediately,
# others will be queued. Here you can define, when the queue will
# be flushed (Note: the queue is automatically flushed after
# completing a file check).
#
# SetMailTime = 86400
SetMailTime = 3600

## Maximum number of mails to queue
#
# SetMailNum = 10
SetMailNum = 5

## Recipient (max. 8)
#
# SetMailAddress=root@localhost
SetMailAddress=holmesw@giac.org

## Mail relay (IP address)
#
# SetMailRelay = NULL

## Custom subject format
#
# MailSubject = NULL
MailSubject = Samhain

## --- end E-Mail ---
```

```
# The binary. Setting the path will allow
# samhain to check for modifications between
# startup and exit.
#
# SamhainPath=/usr/local/bin/yule

## The IP address of the time server
#
# SetTimeServer = (default: compiled-in)

## Trusted Users (comma delimited list of user names)
#
# TrustedUser = (no default; this adds to the compiled-in list)

## Custom format for message header.
## CAREFUL if you use XML logfile format.
##
## %S severity
## %T timestamp
## %C class
##
## %F source file
## %L source line
#
# MessageHeader="%S %T "


## Don't log path to config/database file on startup
#
# HideSetup = False

## The syslog facility, if you log to syslog
#
# SyslogFacility = LOG_AUTHPRIV


## The message authentication method
## - If you change this, you *must* change it
##   on client *and* server
#
# MACType = HMAC-TIGER


[Clients]
##
## This is a sample registry entry for a client at host 'HOSTNAME'. This entry
```

```
## is valid for the default password.
## You are STRONGLY ADVISED to reset te password (see the README) and
## compute your own entries using 'samhain -P <password>'
##
## Usually, HOSTNAME should be a fully qualified hostname,
## no numerical address.
## -- exception: if the client (samhain) cannot determine the
##               fully qualified hostname of its host,
##  the numerical address may be required.
##               You will know if you get a message like:
##               'Invalid connection attempt: Not in
##                client list  what.ever.it.is'
##
## First entry is for challenge/response, second one for SRP authentication.
#
# Client=HOSTNAME@00000000@CDE4239F0EEF4B2C64E442A8BBFEF72349637CB420B62882
Client=web.giac.org@5044353F2E0AF789@69F53300D6CD9CC233610D92BD7047175D13B3
0356B9E62D32B8CAD29F8C86695C0247CC6ACFD373CFD7A557B7986E58735878B4AB8AFB7C7
49F391832ED4B1AA8C05A05E3BB14AA343A1EC2A2558BC236BDAB3CC43BAFB0825B259C0E83
922AC5712EBAD3472562A2890D667A9AFDD79DD56694B50DB0FFCE6BC7410CC5ACE
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.2.6 (SunOS)

iD8DBQFBQ0WXdTNS1wSfL1YRAkQnAJ96kmtySEhQjHdAQ6Ndn61sw/mV8wCghol2
X1ZGl6cFp9inrUpuzLkaKT8=
=MXzv
-----END PGP SIGNATURE-----
```