Global Information Assurance Certification Paper

An Introduction To File Integrity Checking On Unix Systems

With an Example Deployment Using the No-Cost Samhain File Integrity Checker

By

Del Armstrong

Submitted in partial fulfillment of the requirements of the GIAC GCUX certification

Table of Contents

1 Introduction

1.1 Goals of this paper

This paper is an introduction to file integrity checking. It also outlines how to establish a host file integrity checking capability across multiple heterogeneous Unix systems, using the no-cost Samhain file system integrity checker as an example. This paper is based on Samhain 1.7.6.

This paper will discuss the security issues unique to the Samhain system, and it will show an exemplary Samhain configuration.

This paper is not a how-to on configuring a generic host securely, and it's not an all-encompassing document covering the myriad ways to configure Samhain. (Samhain is an incredibly rich program with a large number of configuration options.)

2 File Integrity Checkers

File integrity checkers are host-based tools used to verify that the files on a host have not been tampered with. By detecting unauthorized file modifications, a file integrity checker may be able to raise the alarm when a host has been compromised.

2.1 Why Use File Integrity Checkers

Having a "Defense in Depth" is a critical part of any serious security architecture. The term Defense in Depth means that multiple layers of defense have been established with each layer providing a backup defense in case the layers outside it are defeated. A file integrity checker is an important part of a Defense in Depth; it serves to raise the alarm if the outer layers of defense have been penetrated. A file integrity checker does not prevent an attack, but by detecting unauthorized changes to computer's file system, a file integrity checker may detect a successful intrusion of a host.

Malicious hackers have developed numerous techniques to hide their activity on a compromised machine. If a machine without a file integrity checker is compromised, a careful attacker's activity on the machine may never by detected, greatly increasing the amount of damage that may be done over the course of the compromise.

In the event of a compromise, a file integrity checker can help identify which files may have been damaged during the compromise, thus helping in the cleanup.

2.2 How File Integrity Checkers Work

Generally, file integrity checkers work in two phases. The first phase consists of generating a database made up of cryptographically strong hashes for all the critical files on a host. You can think of these hashes as being a unique fingerprint for each file; it's impossible to change the file without changing its fingerprint. The database usually also contains additional file information such as size, ownership, permissions and modification dates for each file.

In the second phase, once an initial copy of this database is built (the *Initial Baseline*) the same calculation is periodically recalculated for the same critical files on the host. If a file has been tampered with (let's say by an attacker who has compromised the host), one of these characteristics will have changed. Detecting these changes will allow the tampered file to be identified, providing a warning of the system compromise.

If an attacker succeeds in compromising a machine he/she will sometimes attempt to subvert any file integrity checker on the compromised machine. Some file integrity checkers take steps to protect their integrity. For example, some file integrity checkers cryptographically sign the initial baseline, and then periodically check the baseline for tampering. Some file integrity checkers will monitor the OS kernel to detect tampering with the system I/O routines (which could be used to hide changes from the integrity checker).

Finally, if a file is modified by an authorized person, a new hash needs to be calculated for the file and the database must be updated with the new information for the modified file (otherwise the file integrity checker will flag the modified file as having been tampered with).

2.3 Common Features

There are several features commonly found in file integrity checkers. Some of the features are related to the ability of the checker to detect changes, and some are related to protecting the checker against being subverted by attackers.

2.3.1 Cryptographic One-Way Hashes

A one-way hash is a calculation that takes some data as input and produces a unique value as output. The utility of a one-way hash lies in the uniqueness of the output value. A good one-way hash will effectively "fingerprint" the input data; if one bit of the input data changes, the hash will generate a new, unique, output value.

Another important characteristic of one-way hashes is that the original input data cannot be derived from the output value (hence the adjective "one-way"). The terms "checksums", "message digest" and "hashes" are sometimes used to refer to this class of algorithms, especially when the one-way characteristic is not an important feature.

Some hashes commonly used in file integrity checkers are:

- CRC-32: Cyclic Redundancy Checks (CRC) are commonly used to detect errors in data transmission. CRC-32 requires the least amount of processing of the algorithms listed here (i.e. it's fast), but it's insecure and can be fooled. It should only be used in situations requiring minimal security.

- HAVAL: HAVAL is a derivative of the popular MD5 algorithm. It can be configured to be more or less secure than MD5, and therefore to run faster or slower than MD5.

- SHA-1 (Secure Hash Algorithm): SHA-1 is the NIST standard message digest algorithm, developed by the NSA. Both SHA-1 and MD5 are enhancements to an older algorithm called MD4 (which had a known weakness). Since the NSA has kept the development process of SHA-1 secret, there is some worry that it may still share some of MD4's problems. However no weakness has been found in SHA-1, and the NSA is generally regarded as very good at cryptography. SHA-1 is a bit slower than MD5.

- MD5: MD5 is an enhancement to MD4. Unlike SHA-1, the design decisions leading to MD5 have been publicized … allowing for a more thorough analysis. It was developed by the RSA Corporation, a well known cryptography vendor. MD5 has been a very popular hashing algorithm, although a theoretical weakness has been found. Due to this theoretical weakness, SHA-1 is becoming more popular.

- TIGER: TIGER is a relatively new algorithm, which has been designed for high speed on modern 32 bit and 64 bit hardware. Unlike SHA-1 and MD5, it is not derived from MD4. It is faster then SHA-1 and MD5. TIGER currently has no known vulnerabilities.

Different hashes provide different tradeoffs between speed and cryptographic strength. A hash that is very strong (cryptographically) but runs too slowly, is just as useless as a very weak hash that runs quickly. Currently, TIGER probably provides the best balance between speed and cryptographic strength for use by a file integrity checker.

2.3.2 File Attributes

In addition to maintaining a hash to verify the contents of critical files, most file integrity checkers also keep track of the various file attributes maintained by the host operating system. In the Unix world, these attributes include:

2.3.2.1 Inode Number

This is a unique integer that can be used to index into a data structure (known as an "inode") that specifies many of the attributes of an active file on a file system. If a file's inode number changes, that means the file has been deleted and a new file created using the same name as the old one. The terms "inode" and "inode number" are sometimes used interchangeably.

2.3.2.2 Type

Indicates whether the file is a data file, or a special file such as a named pipe or an interface to a device (character special file, block special file …). Special files contain no data and take up no disk space.

2.3.2.3 Owner

Unix associates a unique owner (a Unix account) to each file.

2.3.2.4 Group

Unix supports an object that collects multiple Unix accounts into a single entity, known as a "group". Unix associates a single group with every file.

2.3.2.5 Permissions

Unix enforces access rights by dividing the world into three classes of users (really four), and giving each class some combination of three types of access.

The three classes of users are:

1. Owner: The putative owner of the file. This is the user account that was used to create the file.
2. Group: This is the Unix group that is associated with the file. All the user accounts that make up this group will be given access to this file as specified by the "group permissions".
3. Other: This class of users is everybody who doesn't fall into the Owner or Group classification (with the exception of root, see the next class).
4. root: While not formally a class of users, the root account is effectively a separate class of user. root is a special Unix account, often known as the "super user" account. File access for the root account isn't explicitly specified in the file permissions because it is the same for every file on the computer; root has complete and unrestricted access to a file.

The three types of access granted to each class are:

1. Read: This type of access grants classes of users the ability to read the contents of a file. Note that the ability to execute a program or script contained in a file requires a different type of access.
2. Write: Write access grants a class of users the ability to modify the file in any way.
3. Execute: This is the type of access required to run a program or script.

There are additional flags that are associated with a file in the permissions field.

- SUID: This flag known as the "Set User ID" flag is normally set on a file containing a program to run. When such a program is executed, Unix will treat the running program as though it was run via the user account identified as the "owner" of the file. A common use of this flag is to grant root's access rights to non-root users while running a particular program.
- SGID: This is the "Set Group ID" flag. It is analogous to the SUID flag, except that while the program is running, it is granted the access rights of "group" associated with the file (not the "owner" as in SUID).
- The Sticky Bit: This flag has different meanings depending on whether the file it's associated with is a directory or an executable program.
    - Directories: When the sticky bit is set for a directory, deletion of files in the directory is limited to the "owner" of the files, even if others have write access to the directory. This is a deviation from the normal behavior that allows anybody with write access to a directory to delete files in that directory. The flag is commonly used for the shared directory /tmp.
    - Executables: On an executable file, setting the sticky bit will cause shareable text segments in a program to persist in memory after the program has terminated. This is a performance optimization that can increase the startup speed for some often used programs.

2.3.2.6 Additional Attributes

Under some versions of Unix, files can have additional attributes. The most interesting are "append only" and "immutable". "append only" specifies that the file can only be appended to; once data has been written to the file it cannot be deleted or modified. The second, "immutable", means the file cannot be modified at all. Both of these attributes (and the other, similar ones) apply even to root. Some versions of Unix only permit modification of these attributes when the operating system is running in a special mode that requires physical access to the computer.

2.3.2.7 Timestamps

Three dates are associated with each file. They are the date and time the file's data was last modified, when the file's attributes (including its data) was last modified and the time it was last read.

2.3.2.8 Size

How large the file is, in bytes. Special files will be zero bytes in size.

2.3.2.9 Reference Count

A single file can be referred to via multiple names using a mechanism called "links". The reference count keeps track of how many names refer to the same file.

2.3.2.10 Link Name

If a filename is a link to another file, this attribute is the name of the file the link "points to".

2.3.2.11 Device Numbers

The physical device on which a file resides is indexed by a "major device number" combined with a "minor device number".

2.3.2.12 Examples

Using the Unix '`ls`' command, we can see some examples of these characteristics.

```
chief: ls -li /etc/passwd
17772 -rw-r--r--  1 root  wheel  733 Mar 17 23:28 /etc/passwd
```

In this example:

- The inode number is 17772
- The file permissions are '`-rw--r--r--`', which translates to:
  - The file is a plain file ('`-rw-r--r--`')
  - Owner has Read and Write access ('`-`***`rw-`***`r--r--`')
  - Group has Read access ('`-rw-`***`r--`***`r--`')
  - Others have Read access ('`-rw-r--`***`r--`***')
- The reference count on the file is 1 (i.e. there are no other files linked to it)

- The file *owner* is the root account
- The file *group* is the wheel group
- The file is 733 bytes long
- The file was last modified at 23:28 on March 17 (within the last year)
- This information is about the file /etc/passwd

```
chief: ls -li /dev/tty
14564 crw-rw-rw-  1 root  wheel    1,   0 Apr 18 11:18 /dev/tty
```

In this example:

- The inode number is 14564
- The file permissions are '`crw-rw-rw-`', which translates to:
  - The file is a character special file ('***c***`rw-rw-rw-`')
  - Owner has Read and Write access ('`c`***`rw-`***`rw-rw-`')
  - Group has Read and Write access ('`crw-`***`rw-`***`rw-`')
  - Others have Read and Write access ('`crw-rw-`***`rw-`***')
- The reference count on the file is 1 (i.e. there are no other files linked to it)
- The file *owner* is the root account
- The file *group* is the wheel group
- The file is 0 bytes long (since it's a special file)
- The file was last modified at 11:18 on April 18 (within the last year)
- This information is about the file /dev/tty

```
chief: ls -lio /etc/passwd
17772 -rw-r--r--  1 root  wheel  schg 733 Mar 17 23:28 /etc/passwd
```

In this final example:

- The inode number is 17772
- The file permissions are '`-rw--r--r--`', which translates to:
  - The file is a plain file ('`-rw-r--r--`')
  - Owner has Read and Write access ('`-`***`rw-`***`r--r--`')
  - Group has Read access ('`-rw-`***`r--`***`r--`')
  - Others have Read access ('`-rw-r--`***`r--`***')
- The reference count on the file is 1 (i.e. there are no other files linked to it)
- The file *owner* is the root account
- The file *group* is the wheel group
- The *immutable* attribute is set ('schg' – OpenBSD semantics)
- The file is 733 bytes long
- The file was last modified at 23:28 on March 17 (within the last year)
- This information is about the file /etc/passwd

2.3.3 Database Protection/Verification

One feature all file integrity checkers have in common is that they maintain a database of file hashes and/or file attribute information. This database is a natural target for any attacker since, if an attacker can covertly modify the database, he/she will be able to modify critical system files undetected.

There are three primary techniques used to protect the file information database. The first is to maintain the database on read-only media. For example storing the database on a floppy (with the write protect tab enabled) or a CD-ROM can ensure that only those with physical access to the computer can modify the file information database. This technique works well for a small number of computers or on computers that see infrequent changes. However, since the media must be manipulated or replaced every time the file information database is updated, this technique can become very time consuming for some environments.

Some sites approximate this technique by using a NFS mounted file-system that has been exported read-only. From a security perspective, this is a sub-optimal technique since NFS is a complicated package with a rich history of security problems. Using a NFS server to protect the file information database lowers the overall security of the file integrity checker to the security level of the NFS server.

A second technique is to conduct cryptographic verification of the file information database via either a one-way hash or a cryptographic signature (such as PGP signing the database). This technique allows the file information database be maintained more efficiently since the physical media does not need to be manipulated. It does however significantly increase the complexity of the file integrity checker software.

A third technique is to maintain the database remotely on a dedicated server via a secure protocol. Unlike the case of using a NFS server, in this case a dedicated server can be hardened to a very high degree since it needs only support a secure protocol designed to communicate with the file integrity checker. This technique can also offer operational advantages if multiple machines are to be protected. If the file integrity checker is implemented as a client of a remote file information database server, then updates to the file information database can be implemented in a natural and efficient way.

Many file integrity checkers are also controlled via a configuration file that specifies which files are to be checked along with other operational details. Maintaining these configuration files on a central server allows changes to be easily propagated to numerous clients.

Some file integrity verification systems combine both the second and third techniques (i.e cryptographic verification and remote database servers). However, the majority do not protect themselves against corruption of the database, and rely on the user to implement a protection scheme based on read-only media or offline backups.

2.3.4 Protection Against OS Corruption

In addition to attacking the file integrity checker by corrupting its database, another way to subvert the checker is to modify the operating system routines used to access a file. In some versions of Unix it's possible to load new kernel-level routines while the system is still running. This capability is often referred to as *loadable kernel modules* (LKM). In effect, an attacker can replace the operating system routines used to access a file. This technique allows an attacker to fool a file integrity checker since the replaced routine can be designed to feed misleading data to a checker.

Some file integrity checkers monitor the operating system to detect when new LKM's have been installed. This allows the file integrity checker to raise an alarm when this potentially malicious activity occurs.

2.3.5 Protection Against Debuggers

Some operating systems support debugging utilities such as '`ptrace`', which are capable of taking control of a running program. This ability can be used to subvert a file integrity checker. A few file integrity checkers are capable of protecting themselves against this attack.

2.4 Some Commonly Used Integrity Checkers

There are numerous programs that attempt to verify the file integrity of systems. They range from simple, hacked together shell scripts to very sophisticated and complex client/server based systems with graphical user front-ends and relational database back-ends.

2.4.1 Tripwire

The best known file integrity program is *Tripwire*. Tripwire was the first successful effort to build a robust file integrity checker for Unix systems. It was originally written in 1992 by Gene Kim. At the time, Gene Kim was a student a Purdue under the supervision of Gene Spafford.

The original Tripwire, also known as the "Academic Source Release" (ASR) features a robust set of supported hash functions, the ability to examine file attributes, and a good configuration capability. It was also freely available. Its reporting capabilities are limited in that it can only write results to the screen. It has no database protection or verification capability. This version of Tripwire is still freely available.

A derivative of the ASR is the "Tripwire Open Source, Linux Edition" which is the ASR updated and tweaked to run under Linux. This version is also freely available.

Finally, the company "Tripwire, Inc." was formed by Gene Kim and others to provide a commercial version of Tripwire. The commercial version of Tripwire is much more sophisticated and has been updated to the state-of-the-art in file integrity checking. It allows central administration of Tripwire on all the machines running it, it has features to protect its configuration and database files against tampering and it provides a very friendly graphical front end and a powerful report generating capability. This commercial version of Tripwire has also been ported to Windows.

2.4.2 Other Commonly Used Programs

The first file integrity checker was probably a script similar to:

```
#!/bin/sh
ls -lR / > /tmp/current-files
diff /etc/baseline-files /tmp/current-files
```

Although this script probably still sees a lot of use, Tripwire and other programs have advanced the state of the art. Some other freely available file integrity checkers are: AIDE, FCHECK, SigTree, ViperDB, Sentinel, Claymore (AKA triplight), MD5-tool, Jverify and mtree. Most of these programs are simple checkers along the Tripwire ASR model, although SigTree provides the ability to protect its configuration via PGP signatures.

Although technically not a dedicated file integrity checker, rdist can be used in "byte compare" mode to compare a filesystem against a remote "authoritative" identical filesystem. Other package distribution tools, such as RPM and pkg_info, can also be used to verify portions of a file system.

Even further afield, md5 and SHA-1 checksums for the critical files in a number of Unix operating systems are available at http://www.knowngoods.org/. These checksums can be used to verify that the system files on a machine are unchanged from those distributed with the operating system.

Sun provides a tool to generate MD5 checksums that can be submitted against a database of checksums for all the files in Sun's Solaris distributions. The web site
http://sunsolve.sun.com/pub-cgi/fileFingerprints.pl
uses this tool to allow verification of all the system files on a machine running Solaris.

2.5 The Samhain Package

Since this paper will use an exemplary Samhain configuration to demonstrate the use of a file integrity checker, Samhain is described in more detail.

The Samhain package is a no-cost file integrity checker written by Rainer Wichmann. It contains a number of features to enhance its security and ability to resist tampering - even if the host it's monitoring becomes compromised. Some of these features include:

- File Integrity Checker
  - The usual file attribute and hash checks
  - Will search for new *SUID* executables anywhere on disk
  - Detects loadable kernel module *rootkits* (Linux and FreeBSD only)
  - Powerful configuration capabilities
- Centralized Administration
  - Logging to a central server via encrypted and authenticated connections
  - Can log to a relational database (PostgreSQL/MySQL)
  - Checksum database(s) and client configuration can be stored on a remote server
  - HTML status page for clients
- Tamper resistance
  - Database and configuration files can be PGP signed
  - Logfile entries and e-mail reports are signed
  - Support for stealth operation
    - Hiding and sanitizing of executables
    - Encryption of database and log files
    - Configs hidden via steganography
  - Remote server for database and configuration files
  - Runs in daemon mode
- Additional Features
  - Optional monitoring of login/logout events
  - Multiple logging facilities
  - Graphical, Web-based, front end is available separately

No other file integrity checker offers such a complete menu of features. The Stealth capabilities and the additional event monitoring (LKM attacks, login/logout events, and new SUID file creation) are unique to Samhain. Few other file integrity checkers have a client/server capability or cryptograpically signed database and configuration files. None of the no-cost options, including Tripwire ASR, come close to matching Samhain in terms of flexability or features.

The Samhain package consists of two components, the actual file integrity checker, named Samhain, and a separate program that serves as the remote log/configuration/database server, named Yule. For clarity, in this paper, the term *Samhain* refers just to the file integrity checker and the term *Samhain package* refers to the entire Samhain distribution.

Using Yule is optional. Also optional is which of the three services provided by Yule (logs, configuration files, database files) the file integrity checker uses.

A separately available package named Beltane provides a graphical interface to Samhain and Yule.

In combination, Samhain and Yule can be configured in a variety of ways.

2.5.1 Samhain

Samhain generates the data used to check the integrity of the system. Samhain is capable of running in either a stand-alone mode or as a client in a client-server configuration using Yule as a log/configuration/database server.

In addition to monitoring file integrity, Samhain can also be configured to monitor logins/logouts. Samhain is also capable of monitoring all filesystems for new SUID/SGID files.

Under some versions of Unix, Samhain has additional functionality. Under Linux and FreeBSD, Samhain has the ability to monitor the kernel for LKM based rootkits. Under Linux, Samhain can be configured to report if somebody attempts to debug it via ptrace.

Finally, Samhain can be run either on a one-shot basis (e.g. from cron) or as a daemon. Running as a daemon is generally preferable since this protects it from being tampered with between runs. Also when run as a daemon, Samhain will report a change from the baseline only once (greatly reducing "noise").

2.5.2 Yule

Yule acts as a log server for Samhain. Yule can accept logs from multiple machines running Samhain (multiple *clients*). As mentioned earlier, the use of Yule is optional; Samhain is fully capable of running standalone on the machine it monitors. In addition, Yule also provides the capability to remotely serve configuration files and databases to Samhain clients.

Yule will only communicate with clients that have been registered with it (i.e. clients which have been defined in its configuration file). The client registration process includes establishing a shared secret to be used for mutual authentication between the client and the Yule server. This shared secret is also required to establish a unique session key used to encrypt all communication between the client and the Yule server. The communication is encrypted using the Rijndael encryption algorithm with a 192 bit key. Rijndael was selected in 2000 by the NIST as the new Advanced Encryption Standard and is a fast and powerful encryption algorithm.

Using Yule to accept logs and serve configuration/database files is advantageous for several reasons:

- Sending logs to a remote host helps to protect the logs from being tampered with if the machine being monitored has been compromised.

- A machine dedicated to running Yule can be configured with a much more aggressive security stance than a general-purpose machine (which is where these files would otherwise reside).
- Having multiple clients log to one server greatly simplifies the task of managing and monitoring multiple Samhain installations.
- Having multiple clients retrieve configuration files and databases from one machine simplifies administration of these files.
- The logs and databases should be archived. Having them hosted on a central server greatly simplifies that task.

3 Building A Samhain Infrastructure

This section discusses building an exemplary file integrity checker based on the Samhain package. Appendix A of this document provides a record of an installation based on the discussion on this section.

The first step in building a Samhain infrastructure, and one of the most important, is to analyze the threat environment the Samhain package will be deployed in and then design an appropriate system architecture.

3.1 Analysis and Architecture Design

As mentioned previously, the Samhain package is a very flexible and rich utility that can be configured in a variety of ways. Samhain can quickly be configured to run in a mode equivalent to Tripwire ASR. With more effort (and complexity) Samhain can be configured to run in an ultra-paranoid mode that is very difficult for intruders to detect or subvert. There are advantages and costs associated with these choices as well as the numerous other ways to configure Samhain.

Because of this range of potential configurations, the very first step to installing Samhain is to conduct a thorough analysis of the machines/networks to be protected by Samhain. You need to decide what your threat environment is. More importantly, you need to determine what resources (both hardware and staff time) you have available to build and manage a Samhain deployment.

A file integrity checker is a last line of defense. Its only function is to detect when a host has been successfully compromised, and the attacker has begun making changes on the system. Typically, the attacker will have achieved root access by the time a file integrity checker notices his/her activity. This means that the threat environment the Samhain checker operates in is potentially quite hostile. You should assume that when you need it the most, an attacker has already achieved root access on the box running Samhain.

The threat environment that Yule operates in may be a bit more benign. Yule can be run on a dedicated machine, which allows the machine to be considerably more hardened than most hosts. There is no initial presumption that the Yule server has been compromised. On the other hand, a Yule server presents a very valuable target to an

attacker sophisticated enough to realize that Samhain is being used and who understands the value of the data on a Yule server.

The effort required to manage the Samhain checker can be substantial. There are three aspects to managing an individual checker:

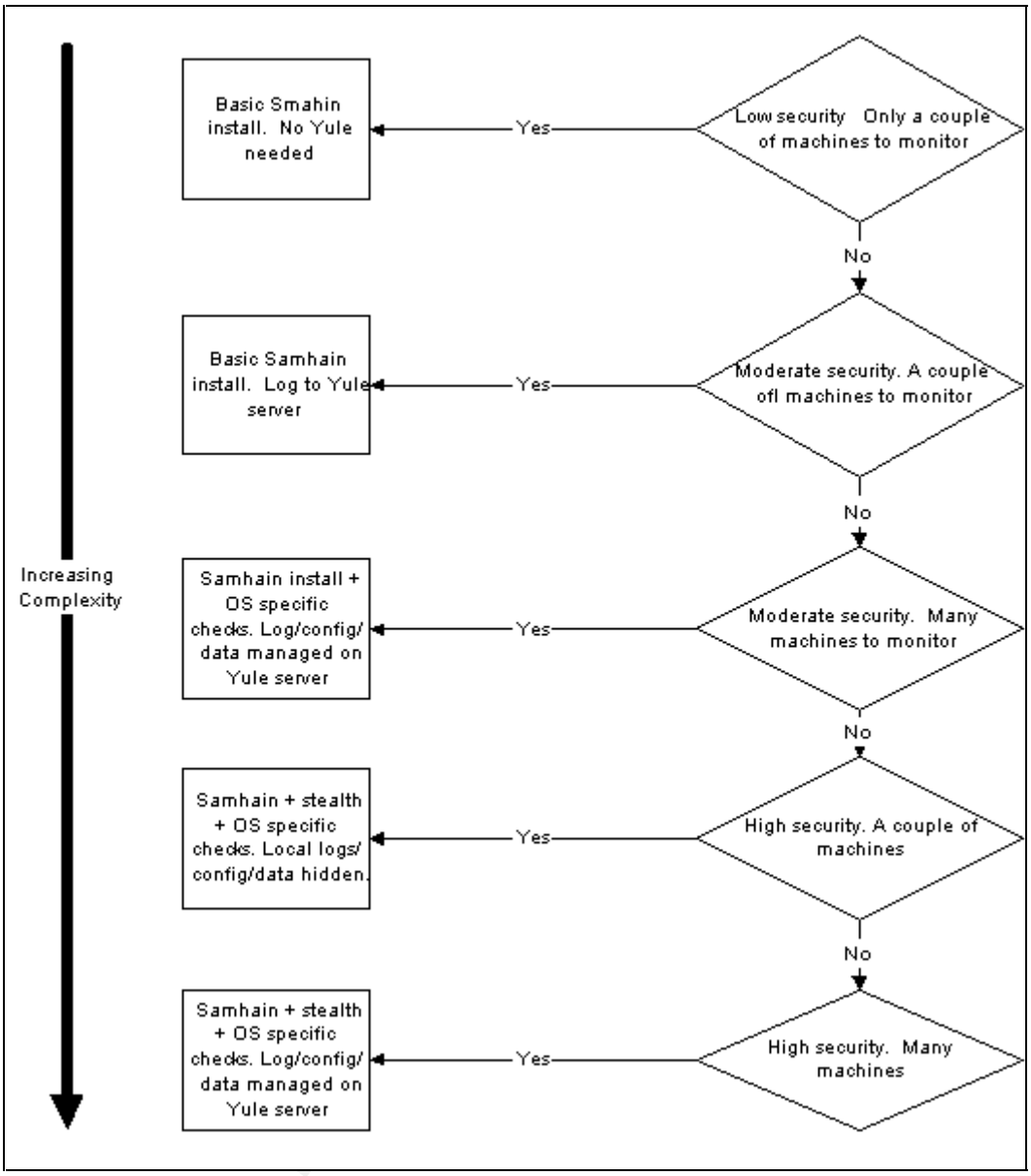
- The output must be monitored and analyzed.
- Authorized changes must be integrated into configs and databases.
- As new machines are acquired, they must have Samhain installed and configured.

All of these activities are labor intensive. The first two require some knowledge and experience to differentiate between benign and hostile changes. It's very common for integrity checkers to generate a numerous false alarms, especially on machines with a lot of activity. Determining which file changes to ignore is a critical activity, making the wrong choice can lead either to undetected compromises or to time-consuming false alarms.

Although the effort to manage a Yule server can also be substantial, it is often more efficient to make a single change on the Yule server and then propagate it to numerous Samhain clients. This is why in a large environment, using Yule offers a significant operational advantage.

Once you've done an analysis of your needs and available resources, you can begin to make design decisions regarding how to deploy Samhain.

As with most risk/benefit analysis, the conclusion you come to will be based on your analysis of the tradeoffs between risk and cost. Here is one possible set of tradeoffs.



3.2 Assumed Environment

Here we describe the hardware/software environment that we assume for this example.

For the purposes of this example, we'll assume a moderate security requirement across many machines. Factors that go into this decision include the fact that the machines to be checked are behind a strong firewall, and limited administrative resources are available to manage numerous Samhain clients.

On the network used to develop this example, one machine was dedicated to acting as a Yule server. That machine was running OpenBSD. The machines running Samhain were running various versions of Linux and OpenBSD.

3.2.1 Network

The network assumed in this paper is a generic IPv4 network. There is no need for clients and servers to be in the same broadcast network, however the logfile server and each of the client machines must be able to communicate via TCP. If there is an intervening firewall, it must permit the clients to open a TCP session to the server on port 49777 (or some other port if desired). By current network standards, the network need not be very fast, a 10 MB Ethernet is more than adequate. Communication between two Samhain clients and a Yule server has been demonstrated across an ISDN connection with acceptable performance.

3.2.2 Yule Server

The machine running the Yule server does not need to be a particularly fast machine. The Yule server is primarily an I/O limited service, although it does do some cryptographic calculations to authenticate clients and their requests. The areas where hardware can limit a Yule server are disk storage and backup capabilities.

Yule will run under any of the current batch of Unix like operating systems. Yule requires an ANSI C compiler such as gcc, and a Posix compliant operating system. Since the Yule server is an attractive target for an attacker, it's important to configure it in as secure a way as possible. A dedicated server running the most recent version of OpenBSD is a good candidate for the Yule server. By default, OpenBSD will install in a minimalist, secure fashion. Other operating systems can also be used, but care must be taken to configure them in a secure way.

A Yule server can end up storing a fair amount of data in the form of raw logs from clients. In addition to reporting changes in files, clients may be configured to report login/logout events, some kernel events and, perhaps most space consuming of all, regular timestamps and status reports. Since each log entry is cryptographically signed, log entries can be much more verbose than might be expected. Also, the log messages tend to be fairly verbose. For example here is a random sample of log entries collected over a minute:

```
 <TCP>  :  [2003-03-14T19:10:47-0700] client=<chief>, msg=<INFO   :  [2003-03-14T19:10:47-0700] msg=<Checking logins>>
35EF2A46FF5AB10CC4967318FF7005C1D1F64FBFF1FD9A0D
MARK   :  [2003-03-14T19:10:49-0700] msg=<---- TIMESTAMP ---->
62536848F89C22DC39D56D4EB296E5C5B0B480C320381734
<TCP>  :  [2003-03-14T19:10:49-0700] client=<chief>, msg=<MARK   :  [2003-03-14T19:10:48-0700] msg=<---- TIMESTAMP ----
>>
CC1BC739BAB6E413FE874D53AC4F493F62FBF5DA9ABB1409
<TCP>  :  [2003-03-14T19:10:58-0700] client=<schleicher>, msg=<MARK   :  [2003-03-14T19:10:58-0700] msg=<Memory used:
max.=
3d792022, current=3d764565>>
58C29725551505AADB6D338378F5B28E1661E2F0EAFBB510
<TCP>  :  [2003-03-14T19:11:09-0700] client=<chief>, msg=<MARK   :  [2003-03-14T19:11:08-0700] msg=<---- TIMESTAMP ----
>>
367720009ED874F3199F5DC8D2697548CDF03889F0BBC362
<TCP>  :  [2003-03-14T19:11:29-0700] client=<chief>, msg=<MARK   :  [2003-03-14T19:11:28-0700] msg=<---- TIMESTAMP ----
>>
4412E3BA4EFC4ED952A8AEDBED9B6236C2B2E499659DCC93
<TCP>  :  [2003-03-14T19:11:38-0700] client=<skipper>, msg=<INFO   :  [2003-03-14T19:11:37-0700] msg=<Checking>,
path=</boot
>>
1C57018C82A7093F0D14A2202E110E9BBE0B4B9B88DDB966
```

Running a default configuration, a fairly quiescent Linux host can generate over 8MB of log entries per day. The amount of disk space a Yule server should be configured with varies, but the more space the better. This data needs to be accessible, and it eventually needs to be archived.

The Yule server in this particular example is also required to serve the configuration and database files to the Samhain clients. These files do not require much disk space, but they should be backed-up on a regular basis (especially the data files).

Due to these backup requirements a Yule server should be equipped with either a tape drive or a CD burner.

The Yule server should be run on a dedicated host; the host it's running on should not be running any other network accessible services, with two possible exceptions, explained below.

The Yule server will probably require remote access. Remote access to the Yule server should be restricted to those with a clear need. Access should be via SSH, and sshd should be locked down as much as possible. At a minimum, sshd should be configured to use tcp_wrappers, and the tcp_wrappers configuration should limit sshd connections to only those hosts that require it. Those authorized to access the Yule server remotely should use the SSH public-key authentication method rather than the Unix password authentication method.

Yule supports an optional web-based graphical management interface (named *Beltan*). In order to use this interface, you'll need to run MySQL and the Apache web server on the computer running Yule. As with SSH, Apache should be configured as securely as possible, for example by limiting which machines can connect to it and requiring all users to authenticate themselves.

Beltane is not part of the exemplary installation presented here. Taking advantage of Beltane requires adding substantial complication to the Yule installation, which in this case in inappropriate. For an installation of this scale, Beltane doesn't add real functionality, but the complexity it requires substantially impacts the overall security of the Yule server.

No other services besides SSH, Yule and possibly HTTPS should be provided by the Yule server. This should be regularly checked with pro-active scanners such as nmap or Nessus.

Naturally, the machine running Yule should be running the Samhain checker. However the configuration for the checker should specify a reporting method other than just logging to the Yule server. Samhain on the Yule server should use an out-of-band technique, such as email, to report significant events on the server.

Since the Yule server is storing sensitive data, it should be placed in a secure physical environment. Physical access to the machine should be restricted, the BIOS should be password protected and should restrict the boot devices to only the boot hard disk. In addition, all backup media from this host should be stored in a secure location.

3.2.3 Machines Running Samhain

Samhain is capable of adversely impacting the machine it runs on; it is both disk I/O intensive and computationally intensive since it calculates a hash for every file. However, Samhain provides configuration options to adjust the priority at which the checker runs (via 'nice'), how I/O intensive it is, as well as how often it rechecks things when it's running in daemon mode. By taking advantage of these configuration options, Samhain can be configured to have minimal impact on the machine it monitors. On the network used to develop this example, Samhain was run on a 120 MhZ Pentium (24MB ram) dedicated firewall/DNS server with no adverse result.

The Samhain checker requires a POSIX compliant Unix to run. Building Samhain requires the standard compilers and libraries included in most Unix distributions. Approximately 8MB is disk space is required to build the client. The compiled client requires less than 600K of disk space.

It's best if Samhain is installed and the initial baseline is built, when a machine has a fresh operating system installed and before the machine is accessible on a network. This may not be practical, but every effort should be taken to ensure that the initial baseline is built on a 'clean' machine.

3.3 Installing Yule

3.3.1 Building the Yule Server

In this example, the Yule server is a dedicated server running OpenBSD 3.1. OpenBSD was selected due to its strong default security stance. The hardware used in this example is a Pentium based computer with 256 MB of ram and 40 GB of disk space. Computationally, this hardware is overkill for the Yule server. 40 GB is an adequate amount of disk space. The machine used in this example did not have a backup capability so none is demonstrated.

OpenBSD is best installed using the CD set distributed by openbsd.org (http://www.openbsd.org/orders.html). The CD set comes with installation instructions, and the income from selling the CDs helps fund development of OpenBSD.

Install OpenBSD using the defaults given in the installation instructions. Do not select the install sets:

- game32.tgx
- xbase32.tgz
- xshare32.tgz
- xfont32.tgx

- xserv32.tgx

These install sets are not required for a dedicated Yule server. The rest of the OpenBSD installation can be done as described in the installation instructions.

After OpenBSD is installed, the gpg (GNU version of PGP) package needs to be installed if PGP signing of databases and configuration files is going to be used. To install gpg:

```
pkg_add -v ftp://ftp.openbsd.org/pub/OpenBSD/3.2/packages/i386/gnupg-1.0.7.tgz
```

If you're unfamiliar with using gpg, a good resource is:
http://www.gnupg.org/(en)/documentation/

Although not required, it's a good practice to add a user account '`yule`' for the Yule daemon to run under. If this account is missing, Yule will run under the '`daemon`' account instead.

```
adduser -group USER -shell nologin -batch yule
```

The default OpenBSD installation will require no additional hardening (although there are many additional steps which can be used to provide additional security).

3.3.2 Installing the Yule Software

There are several steps to be followed when installing Yule. Most of the work is preparation and setup. Actually building and installing Yule is pretty straightforward.

The very first step is to download and verify the Samhain package from the Samhain home site: http://www.la-samhna.de/samhain/index.html .

Once the tar file has been downloaded, unpack it unto a working directory. It will unpack into two files, a compressed tar archive and a PGP signature file. The PGP signature file will contain a PGP signature of the compressed tar file as signed by Rainer Wichmann.

You can download the public key for Rainer Wichmann via the instructions at:
http://www.la-samhna.de/samhain/s_rkey.html .

After you've added Rainer Wichmann's key to your PGP keyring, you should verify the key you received with the PGP fingerprint:

```
EF6C EF54 701A 0AFD B86A  F4C3 1AAD 26C8 0F57 1F6C
```

(This fingerprint is based on a key obtained from a pgp.net keyserver in Jan of 2003.)

Once you've verified the distribution you received, unpack the compressed tar archive. You may wish to save this tar archive for building Samhain clients on other machines.

Now begins the most critical phase of the entire installation. Sit down and study the manual included in the Samhain package distribution. The manual is provided both in postscript form, and as a series of web pages (MANUAL-1_7.ps and MANUAL-1_7.html.tgz).

This paper is not a replacement for reading the manual, only the manual included in the Samhain package will give you the numerous details required to successfully run Samhain.

3.3.3 Selecting Features To Use

The next part of installing Yule is to select which features to use. Unlike Samhain, Yule does not have many compile options. Compile options are selected during the 'Configure' process. Suggested options are:

--enable-ptrace --enable-static --enable-network=server

Note: the enable-ptrace option causes Yule to crash under OpenBSD, however it is recommended for use on Linux machines.

Building and installing the Yule server is done with the usual '`make`' followed by '`make install`'.

The Yule and Samhain distributions also provide the make targets '`make install-user`' and '`make install-boot`'. The first will install Yule to run as the user 'yule' and the second will install scripts to start Yule and Samhain when a machine boots.

Note: for both Yule and Samhain, '`make install`' will remove read access for 'others' from /etc. This is a bug that will break many things, and you should change the permissions to restore read access right away.

3.3.4 Configuring Yule

The Yule server is configured via changes to the configuration file, '/etc/yulerc'. The default Yule configuration is pretty good. You might consider enabling daemon mode for Yule by adjusting the line:

```
Daemon=no
```

To:

```
Daemon=yes
```

3.4 Installing Samhain

The Samhain client is built from the same distribution used to build Yule. If you've already downloaded and verified the Samhain distribution to build Yule, simply do a

'`make clean`' and proceed to build the Samhain client. Otherwise first follow the download and verification procedures described for installing Yule.

3.4.1 Selecting features to use

Which features to use when compiling Samhain is determined largely by the results of the analysis and architecture design process. Some of the compile-time options to consider using are:

`--enable-network=client` *(This option creates the Samhain client rather than Yule)*
`--enable-static` *(Compiling as a static executable is more secure)*
`--enable-suidcheck` *(Search entire filesystem for rogue SUIG/SGID files)*
`--enable-login-watch` *(Keep track of new logins and logouts)*
`--enable-db-reload` *(Reload database on HUP signal, eases administration)*
`--with-gpg=/usr/local/bin/gpg` *(Use the PGP program gpg to authenticate configs and database file)*
`--with-logserver=127.0.0.1` *(Use the Yule server at the given IP address)*
`--enable-base=1887410363,10649531` *(Use the given values to generate key used to authenticate log entries)*
`--with-config-file=REQ_FROM_SERVER/etc/samhainrc` *(Where to get config files)*
`--with-data-file=REQ_FROM_SERVER/etc/data.samhain` *(Where to get data files)*
`--with-recipient=name@some-email-address` *(Default email address to use)*
`--with-kcheck=SYSTEM_MAP` *(Check for attempts to subvert kernel)*
`--enable-ptrace` *(Abort if attempt is made to attach debugger)*
`--enable-stealth=XOR_VAL` *(Enable stealth options, obfuscate strings in executable)*
`--enable-install-name=NAME` *(Hide Samhain by renaming files to NAME)*

There are numerous other configuration options, mostly designed to deal with specific issues or provide less commonly used features.

3.4.2 Building Samhain

Building the Samhain client is the same as building Yule. Issue the command '`make`', and install with '`make install`'. You must be root to install.

3.4.3 Configuring Samhain

Samhain and Yule use a shared secret to authenticate and encrypt their communications. If the Samhain client is to be used with Yule, the Samhain executable must have this password installed and the Yule configuration file must also include this password. This is a requirement for all Samhain clients to communicate with a Yule server.

The password must consist of 16 hexadecimal characters. You can use the command:

```
yule --gen-password
```

to generate a random password. You can embed this password in the Samhain executable with the `samhain_setpwd` command. For example:

```
# samhain_setpwd /usr/local/sbin/samhain new 77C9CEFC65770CCD
INFO   old password found
INFO   replaced:  f7c312aaaa12c3f7  by:  77c9cefc65770ccd
INFO   finished
#
```

In order for Yule to recognize the password for a client, a hash of the password must be added to the Yule configuration file, '`yulerc`'. Use the Yule '-P' option to convert the hex password into a string to add to the Yule configuration. Take the output from this command, replace the string '`HOSTNAME`' with the hostname that the Samhain client will run on and add to the bottom of the `yulerc` file. Here is an example of the output from '`yule -P`':

```
# yule -P 77C9CEFC65770CCD
Client=HOSTNAME@0052C56C8641A615@2C72F87F5547613DB7B8461532452AB67A4F43
598DEFFD1AA0CBD5A326EEE0B1F2395F146861F1729D82BC54C19
E7AED45A5A85E9CE3641E7D2C6C5F9A8BE8CB8587807B97B96B2A83856BEBCA5649F80D
0D7A4235F4773DD62E68A36215369AE2F483377CC3AF54E467A8E
0146BB37AF43C737766D68467AB54D6242C556821
```

If when you built Samhain, you specified using PGP to verify the config and database files, the config file must be signed using the gpg '`–clearsign`' before Samhain can use it. The account used to run Samhain must have the public key needed to verify this key in its default keyring.

Complete the installation by building a baseline database. Build a baseline by invoking the Samhain client with the '`-t init`' option. The baseline will be placed in the location specified via the '`--with-data-file`' option when Samhain was built.

When using PGP to verify the database, after the baseline database has been created and copied to the Yule server, it also must be PGP signed in the same manner as the config file.

You can now run Samhain in check mode, using the '`-t check`' option. It's a good idea to verify the correct operation of the Samhain checker by making a trivial modification to a critical file.

3.4.4 Optimizing Samhain

On most machines, Samhain will run with minimal impact. However there is a potential for Samhain to impact low end or heavily loaded machines. Samhain impacts a machine in two ways:

- Computationally as it calculates the fingerprint for each file
- Increased I/O load as it reads the contents of all the files it checks

You can reduce Samhain's impact on a machine by taking the following steps:

- Used MD5 instead of TIGER to fingerprint files. This is controlled by the `DigestAlgo` option in the Samhain configuration file. Using MD5 will speed up fingerprinting by about 20%.
- Renice Samhain (i.e. adjust the priority the OS assigns to running Samhain). This can be controlled via the `SetNiceLevel` option in the Samhain configuration file. See the Unix man page on the '`nice`' command for more information on nice levels.
- Reduce the maximum I/O rate Samhain allows itself. You can accomplish this by reducing the `SetIOLimit` parameter in the Samhain Configuration file.
- Throttle back the SUID file checker. In the Samhain configuration file, lower `SuidCheckFps`, which controls how many files-per-second the SUID checker will check.

On operating systems that support it, consider turning on the 'immutable' bit for the Samhain executable. This won't improve performance, but it will enhance the security of Samhain.

3.5 Maintenance

Samhain and Yule are complex systems that require ongoing maintenance. There are two aspects of maintaining these systems: keeping up with updates and maintaining the config and data files.

The first step to maintaining Samhain and Yule is to subscribe to the *samhain-announce* mailing list. You can subscribe at http://www.la-samhna.de/elist.html. This list will keep you informed of new versions, and most importantly, inform you of any security updates. Given that Samhain and Yule can be critical to your security stance, it's important that you stay informed regarding any security problems that may arise with these programs.

Although it is outside the scope of this paper, it's important to backup the configuration and database files residing on the Yule server.

3.6 Operations

As with all aspects of Samhain, there are many choices in how you choose to use it in production. Samhain is capable of operating in a sophisticated and complex way, or of operating in a simple and easy to manage way.

While it's easy to be seduced by all the bells and whistles available in Samhain, erring on the side of simplicity will lead to the most secure installation. The bells and whistles are best reserved only for those situations that require them.

In the example presented here, we stick to using simple techniques to monitor and manage the Samhain installation. The techniques used here are suitable for small and medium sized Samhain installation, probably on the order of 25 clients or less.

There are two primary tasks associated with running a Samhain installation:

1. Monitoring the events generated by the clients.
2. Updating the configuration and data files to maintain a good signal to noise ratio in the events generated by the clients.

3.6.1 Monitoring Events

In the Samhain installation used for this example, there is one primary source of information to monitor. All the Samhain clients are logging to the Yule server, and the Yule server is combining all the input streams into one log file: `/var/log/yule/yule_log`.

This is very convenient, since only one file needs to be monitored. Yule does have the ability to maintain separate log files for each client, controlled by the `UseSeparateLogs` configuration option in `yulerc`.

By default, serious events are labled with a severity of 'CRIT', which stands for critical. By monitoring the combined input stream for events with a critical serverity, you can be sure to catch any potential security events.

The primary tools used for monitoring the log file in this example are `tail` and `grep`. To monitor the Yule log file for 'critical' events, issue the command:

```
tail –f /var/log/yule/yule_log | grep CRIT
```

You can use additional grep commands to trim the output. For example, to ignore the numerous messages about 'POLICY ADDED' when reloading a Samhain configuration file, use the following command to monitor the log file:

```
tail –f /var/log/yule/yule_log | grep CRIT | grep –v 'POLICY ADDED'
```

In the installation used in this example, the command above is run in a small window that is continuously logged-in to the Yule server.

Each logged event takes two lines in the log file, the first line contains the event and the second line is a verification signature. The signature can be checked with the '-L' option to Samhain.

The log entry below is a simple informational entry. It consists of two lines. The first line contains:

- The protocol used to send the entry to Yule (TCP)
- The date and time the log entry was received
- The client logging the event (skipper)
- The severity of the event (INFO)
- The time of the event as seen by the client
- Additional info (the event is notification that Samhain is checking the files in /boot)

The second line is a signature of the event.

```
<TCP>  :  [2003-05-02T00:19:51-0600] client=<skipper>, msg=<INFO   :  [2003-05-
02T00:19:51-0600] msg=<Checking>, path=</boot>>
C7110026372393F31A54B8C543CE7B53E4F130D1968954D8
```

This next log entry is a Critical event entry. It alerts to the appearance of an unexpected SUID file named /usr/X11R6/bin/xmcd-

- The protocol used to send the entry to Yule (TCP)
- The date and time the log entry was received
- The client logging the event (schleicher)
- The severity of the event (CRIT)
- The time of the event as seen by the client
- Additional info (the event is notification that Samhain has found an unexpected SUIR/SGID file name /usr/X11R6/bin/xmcd-)

The second line is a signature of the event.

```
<TCP>  :  [2003-05-02T00:24:26-0600] client=<schleicher>, msg=<CRIT   :  [2003-05-
02T00:24:23-0600] msg=<POLICY SUIDCHK  suid/sgid file not in database>,
path=</usr/X11R6/bin/xmcd->>
879FCE4A69A6311EFEF591A160FD030310117F11F1EAD1B1
```

Determining which events are significant requires some knowledge of the normal activity on the system being monitored. In general, if the files being monitored are selected with some care, then any 'critical' change noticed by Samhain should be investigated. Over time experience will show which file modifications are normal, and which are abnormal and should be investigated.

Some changes are obviously of concern, for example changes to important executables should be taken very seriously. Other changes may require more investigation, especially if the purpose of the modified file is unfamiliar. As another example, the appearance of a new SUID file is a very serious event, which should be investigated immediately.

3.6.2 Monitoring the Yule Server and Samhain Clients

In addition to monitoring the output from the Samhain clients for important events, it's important to be aware of potential problems with the Yule server and the Samhain clients.

The Yule server maintains a file of status information for both itself and the clients using it. This file is formatted as HTML, and is suitable for displaying in any web browser.

Server Status

Current time: 21-04-2003 13:46:21
Startup time: 14-03-2003 19:03:03

Connections: (maximum 57 simultaneously)
open 2
total 1625183

Last connection: 21-04-2003 13:46:20

Clients

chief.domain-name.com	Message	[2003-04-21T13:46:20-0600]
	POLICY	[2003-04-21T13:45:59-0600]
	File transfer	[2003-04-20T21:32:33-0600]
	Started	[2003-04-20T21:31:52-0600]
schleicher.domain-name.com	Message	[2003-04-21T13:46:11-0600]
skipper.domain-name.com	POLICY	[2003-04-21T13:46:20-0600]
	Message	[2003-04-21T13:45:55-0600]
	File transfer	[2003-04-21T12:49:02-0600]
	Exited	[2003-03-17T17:01:30-0700]
	Started	[2003-03-17T16:30:34-0700]
	PANIC	[2003-03-17T16:22:26-0700]
	FAILED	[2003-03-17T16:21:04-0700]

This file is updated continuously and will refresh every couple of minutes in most web browsers.

It is not recommended that a web server run on the Yule server in order to make this file available. Rather, run a web browser on the Yule server. For example, configure ssh to pass X-Window traffic back to the originating workstation and then view the file with a graphical web browser such as Mozilla or Opera. Lynx also does a fine job of monitoring the file, although it will need to be refreshed manually to view updates.

3.6.3 Updating Samhain Configuration Files

As was mentioned earlier, Samhain alert messages can be pretty verbose. For example, here's the type of log entry generated by a change to /etc/passwd:

```
<TCP>  :  [2003-04-20T14:51:58-0600] client=<skipper>, msg=<CRIT   :  [2003-04-
20T14:51:57-0600] msg=<POLICY [ReadOnly]>, path=</etc/passwd>, size_old=<906>
size_new=<916> ctime_old=<[2003-03-12T04:48:32]>, ctime_new=<[2003-04-20T20:41:25]>,
mtime_old=<[2003-03-12T04:48:32]>, mtime_new=<[2003-04-20T20:41:25]>,
chksum_old=<87359801E60DDB11224D1DFC2D292E89CE9487CAF2CAAE33>,
chksum_new=<C2AD9B5ECA2A4A914463001D3D8DA560363370F2B805F508>, >
```

Not only are the messages verbose, there's also a lot of noise in the default configuration. For example, in one sample of 331208 messages, only 550 actually involved potentially unauthorized activity. The remaining messages were status messages of various sorts. The 550 potentially important messages were repeated warnings about unexpected SUID files, e.g.:

```
<TCP>  :  [2003-04-16T12:02:47-0600] client=<skipper>, msg=<CRIT   :  [2003-04-
16T12:02:47-0600] msg=<POLICY SUIDCHK  suid/sgid file not in database>,
path=</usr/local/bin/ssh1>>
<TCP>  :  [2003-04-16T12:02:48-0600] client=<skipper>, msg=<CRIT   :  [2003-04-
16T12:02:47-0600] msg=<POLICY SUIDCHK  suid/sgid file not in database>,
path=</usr/local/bin/ssh>>
```

As you use Samhain, you will inevitably need to adjust the configuration files or the database files. When using Yule to serve the config files, making a change is a three-step process:

- Change the config file on the Yule server.
- Sign the modified file using PGP
- On the Samhain client machine, issue a HUP signal to the Samhain checker. This will cause Samhain to reload its configuration from the modified file.

It will also be necessary to adjust the database on a regular basis. As critical files are updated or modified you will want to update the database file so that Samhain is aware of the new attributes and hashes for these files.

With a new installation, it's best to monitor the logs files closely to get a picture of what sort of changes occur on your network. The logs will probably be pretty noisy - full of warnings about normal activity. As you get experience with using Samhain in your environment, you should update the Samhain configurations to minimize the number of false alarms you get.

In this case, the messages about unexpected SUID files are occurring because the configuration file for the host 'skipper' does not include a check for the directory /usr/local/bin. As a result, any suid files in /usr/local/bin are flagged as unexpected suid files. Worse yet, the files in /usr/local/bin are not being monitored for unauthorized changes.

To resolve issues like this, the configuration file for the host must be updated. To update the configuration file for the Samhain daemon on skipper:

1. Edit the file '`/var/lib/yule/rc.`*`FQDN`*' (where FQDN is the full domain name of the host)
2. To specify checking all of `/usr/local` except `/usr/local/src`:
    a. Add '`dir=-1/usr/local/src`' immediately after the line with '`[IgnoreAll]`'
    b. Add '`dir=4/usr/local`' immediately after the line with '`[ReadOnly]`'
3. For clarity, remove the PGP signature lines at the start and end of the file. This is optional; the lines can be left in with no ill effect.
4. Re-sign the file, e.g.: `gpg --clearsign /var/lib/yule/rc.`*`FQDN`*
5. Install the signed file:
   `mv /var/lib/yule/rc.`*`FQDN`*`.asc /var/lib/yule/rc.`*`FQDN`*
6. On the host running Samhain, make Samhain reload it's configuration by sending it the '`HUP`' signal: `kill -HUP `cat /var/run/samhain.pid``

Note that when you reload the configuration file, Samhain will generate a 'POLICY ADDED' event for every file in the database, e.g.:

```
<TCP>  :  [2003-04-20T17:18:46-0600] client=<skipper>, msg=<CRIT   :  [2003-04-
20T17:18:44-0600] msg=<POLICY ADDED>, path=</usr/bin/allcm>>
CF94AE7C4D1AE8044D66C49A9541D233BD2E5C8A8D8E72AA
```

3.6.4 Updating Samhain Data Files

In addition to adding or deleting files from the configuration file, sometimes the attributes of a file being monitored will change and the file information database will need to be updated.

Updating the database is not a time critical activity, since Samhain will only report on a file change once each time it's run. So typically this means that a given change will be reported when it's discovered, and then once every time the host reboots. However, over time these changes add up, and it's productive to update the database to prevent a flurry of reports every time a machine reboots.

To update the data file for a host, issue the command:

```
samhain -t update -l none
```

on the host to be updated. This will create a local, updated version of the file attribute database. The file must now be copied to the Yule server, signed with PGP, and installed where Yule can find it. This is done using the same process used to install the initial baseline.

On the client host:

```
skipper:~ 78% sudo /usr/local/sbin/samhain.new -t update -l none
INFO   :  [2003-04-21T15:37:40-0600] msg=<Downloading configuration file>
INFO   :  [2003-04-21T15:37:41-0600] msg=<Session key negotiated>
INFO   :  [2003-04-21T15:37:42-0600] msg=<File download completed>
skipper:~ 79% sudo scp /etc/data.samhain dela@chief:/tmp/
```

Now that the new database has been copied to the Yule server, it needs to be signed and put in place. On the Yule server:

```
chief: gpg --clearsign /tmp/data.samhain
gpg: Warning: using insecure memory!
You need a passphrase to unlock the secret key for
user: "Del Armstrong (Work) <del_armstrong@somedomain.com>"
1024-bit DSA key, ID DB9FA9A2, created 2000-10-20
Enter passphrase:
chief: sudo mv /tmp/data.samhain.asc /var/lib/yule/file.skipper.somedomain.com
chief: sudo chown root.wheel /var/lib/yule/file.skipper.somedomain.com
```

The updated database is ready. The next time the Samhain client on the host restarts, it will use this updated database. Note: once started, the Samhain client maintains an image of the database in memory, so it won't use the updated database until its restarted.

3.6.5 Future Enhancements

One of the strengths of Samhain is the varied ways in which it can report changes. Some of these reporting techniques include email, user written scripts, logging to a SQL database and the Beltane graphical front end . Once you've gotten a handle on what's normal activity, you should consider whether these advanced reporting capabilities are appropriate to your situation. In Appendix C, a Samhain configuration is shown which reports Critical events via email.

References

- Samhain Home Page - http://www.la-samhna.de/samhain/index.html
- **The Samhain Manual** (included in Samhain distribution)
- **Practical Unix & Internet Security, 3rd Edition** by Simson Garfinkel, Alan Schwartz, Gene Spafford. Especially Chapter 20.
- **Applied Cryptography: Protocols, Algorithms, and Source Code in C, Second Edition** by Bruce Schneier
- **Basic File Integrity Checking** *by* Jeremy Rauch - http://www.securityfocus.com/infocus/1408
- **The Design of a System Integrity Monitor: Tripwire**. Gene Kim and E. H. Spafford; Department of Computer Sciences, Purdue University; CSD-TR-93-071; Coast TR 93-01; 1993 - http://www.cerias.purdue.edu/homes/spaf/tech-reps/9371.ps
- **Experiences with Tripwire: Using Integrity Checkers for Intrusion Detection**. Gene H. Kim and E. H. Spafford; PROCEEDINGS OF THE SYSTEMS ADMINISTRATION, NETWORKING AND SECURITY CONFERENCE III (SANS); Washington DC; Coast TR 94-03; 1994. - http://www.cerias.purdue.edu/homes/spaf/tech-reps/9412.ps
- **Tripwire Users Manual.** Available from the Tripwire open-source distribution site - http://sourceforge.net/projects/tripwire/
- **Bypassing Integrity Checking Systems** by halflife; Phrack Magazine Volume 7, Issue 51 September 01, 1997, article 09 of 17 http://www.phrack.org/show.php?p=51&a=9
- OpenBSD home Page - http://www.openbsd.org/
- Building a Secure Mobile OpenBSD Workstation, GCUX Practical by Curtis Collicutt - http://www.giac.org/practical/GCUX/Curtis_Collicutt_GCUX.pdf
- LinuxSecurity.com Home Page - http://www.linuxsecurity.com/
- AIDE Home Page - http://www.cs.tut.fi/~rammer/aide.html
- FCHECK Home Page - http://www.geocities.com/fcheck2000/fcheck.html
- SigTree Download - http://www.discord.org/~lippard/software/sigtree.tgz
- ViperDB Home Page - http://panorama.sth.ac.at/viperdb/
- Sentinel Home Page - http://zurk.sourceforge.net/zfile.html
- Claymore Home Page - http://freshmeat.net/projects/triplight/?topic_id=43
- MD5-tool Home Page -http://razor.bindview.com/tools/desc/md5-tool_readme.html
- Jverify Home Page - http://freshmeat.net/projects/jverify/?topic_id=43%2C150%2C253
- Mtree – See 'man mtree' on most BSD based systems
- **Aide: Poor Man's File Integrity Checker** by Tapan Meshram - GSEC 0690 May 1, 2001 - http://www.sans.org/rr/audit/aide.php
- *Standard Cryptographic Algorithm Naming*, a useful page on hashing algorithms - http://www.users.zetnet.co.uk/hopwood/crypto/scan/md.html

- Tiger hash algorithm home page: http://www.cs.technion.ac.il/~biham/Reports/Tiger/
- RFC 1321, The MD5 Message-Digest Algorithm by R. Rivest: http://www.ietf.org/rfc/rfc1321.txt
- **HAVAL — A One-Way Hashing Algorithm with Variable Length of Output.** Yuliang Zheng, Josef Pieprzyk and Jennifer Seberry; Department of Computer Science, University of Wollongong - http://www.calyptix.com/files/haval-paper.pdf
- National Institute of Standards FIPS PUB 180-1 (SHA) http://www.itl.nist.gov/fipspubs/fip180-1.htm
- **Unix Security Checklist V2.0**. CERT. - http://www.cert.org/tech_tips/usc20_full.html
- **GNU Privacy Guard Home Page -** http://www.gnupg.org/
- **Beltane Home Page -** http://www.la-samhna.de/beltane/index.html
- **Advanced Encryption Standard Home Page (Rijndael) -** http://csrc.nist.gov/CryptoToolkit/aes/
- **NMAP Home Page -** http://www.insecure.org/
- **Nessus Home Page -** http://www.nessus.org/

Appendix A - An Exemplary Installation

In this example, we go through the complete process of installing Samhain and Yule, showing the command that were issued and the output from those commands. This Appendix was generated by going through a complete installation, using the Unix 'script' command to record the installation. The record was then edited to remove extraneous commands/output, clarify the output or obfuscate private information.

Comments and explanations will be in an *italic* font, commands issued will be in a **`Bold Courier`** font, and output will be in a `Courier` font.

In some places, the font-size will be reduced to minimize word wrap. But in some places word wrap is unavoidable.

The installation is the same installation described in section 3. There is a dedicated server running Yule, and two client machines running the Samhain checker (in this instance running as a client). The Yule server, named 'chief', is running OpenBSD 3.1. The two clients are named 'skipper' and 'schleicher', and are running different version of Linux.

Although it is desirable to work as an unprivileged user and use 'sudo' for root access, in this example are work is done as root (for clarity). Since future maintenance will be done as the unprivileged user 'dela', the PGP private key for 'dela' is used to sign the configuration and data file.

First we install Yule on the Yule server (a computer named 'chief').

Download the distribution

```
Chief# wget -nv http://www.la-samhna.de/samhain/samhain-current.tar.gz
22:536:38 URL:http://www.la-samhna.de/samhain/samhain-current.tar.gz
[792495/792495] -> "samhain-current.tar.gz.1" [1]
```

Unpack the distribution

```
chief# cd /usr/local/src
chief# mkdir samhain-1.7.6
chief# cd samhain-1.7.6
chief# zcat ~/samhain-current.tar.gz | tar xf -
chief# ls
samhain-1.7.6.tar.gz           samhain-1.7.6.tar.gz.asc
```

Verify the distribution using PGP. First download the key

```
chief# wget -nv http://www.la-samhna.de/samhain/s_rkey.html
```

```
23:01:20 URL:http://www.la-samhna.de/samhain/s_rkey.html [13627/13627]
-> "s_rkey.html" [1]
```

Now import the developer's public key into the keyring

```
chief# gpg --import s_rkey.html
gpg: key 0F571F6C: public key imported
gpg: Total number processed: 1
gpg:               imported: 1
```

Now verify the key against the known fingerprint (see section 3.3.2)

```
chief# gpg --fingerprint | grep 'EF6C'
     Key fingerprint = EF6C EF54 701A 0AFD B86A  F4C3 1AAD 26C8 0F57
1F6C
```

Finally, verify the distribution tar file

```
chief# gpg --verify samhain-1.7.6.tar.gz.asc samhain-1.7.6.tar.gz
gpg: Signature made Thu Apr 24 12:06:53 2003 MDT using DSA key ID 0F571F6C
gpg: Good signature from "Rainer Wichmann <rwichmann@la-samhna.de>"
gpg:                 aka "Rainer Wichmann <rwichmann@hs.uni-hamburg.de>"
Could not find a valid trust path to the key.  Let's see whether we
can assign some missing owner trust values.
No path leading to one of our keys found.
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
gpg: Fingerprint: EF6C EF54 701A 0AFD B86A  F4C3 1AAD 26C8 0F57 1F6C
```

Unpack the tar file

```
chief# zcat samhain-1.7.6.tar.gz | tar xf -
chief# cd samhain-1.7.6
```

Before we configure Yule, create the 'yule' user account. This account will be noticed by the configuration script and used by Yule.

```
chief# adduser
Use option ``-silent'' if you don't want to see all warnings and
questions.
Reading /etc/shells
Check /etc/master.passwd
Check /etc/group
Ok, let's go.
Don't worry about mistakes. I will give you the chance later to correct
any input.
Enter username [a-z0-9_-]: yule
Enter full name []: Yule Daemon
Enter shell csh ksh nologin sh tcsh [csh]: nologin
Uid [1004]:
Login group yule [yule]:
Login group is ``yule''. Invite yule into other groups: guest no
[no]: no
Enter password []:
Set the password so that user cannot logon? (y/n) [n]: y
```

```
Name:     yule
Password: ****
Fullname: Yule Daemon
Uid:      1004
Gid:      1004 (yule)
Groups:   yule
HOME:     /home/yule
Shell:    /sbin/nologin
OK? (y/n) [y]: y
Added user ``yule''
Copy files from /etc/skel to /home/yule
Add another user? (y/n) [y]: n
Goodbye!
```

The 'yule' account needs to have a PGP keyring with the public key needed to verify the signature on the configuration and data files. We just copy the public keyring from the account used to sign these files into place.

```
chief# mkdir ~yule/.gnupg
chief# cp ~/.gnupg/pubring.gpg ~yule/.gnupg
chief# chown -R yule:yule ~yule/.gnupg
```

Now we run configure to set up for building Yule

```
chief# ./configure --enable-static  --enable-network=server
checking for a BSD-compatible install... /usr/bin/install -c
checking whether make sets ${MAKE}... yes
checking build system type... i386-unknown-openbsd3.1
checking host system type... i386-unknown-openbsd3.1
checking for gcc... gcc
checking for C compiler default output... a.out
checking whether the C compiler works... yes
checking whether we are cross compiling... no
checking for suffix of executables...
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking how to run the C preprocessor... gcc -E
checking for a BSD-compatible install... /usr/bin/install -c
checking whether ln -s works... yes
checking for gawk... no
checking for mawk... no
checking for nawk... nawk
checking for hostname... /bin/hostname
checking for ANSI C header files... yes
checking for sys/types.h... yes
checking for sys/stat.h... yes
checking for stdlib.h... yes
checking for string.h... yes
checking for memory.h... yes
checking for strings.h... yes
checking for inttypes.h... yes
checking for stdint.h... no
checking for unistd.h... yes
checking for sys/types.h... (cached) yes
checking for sys/stat.h... (cached) yes
```

```
checking for stdlib.h... (cached) yes
checking for string.h... (cached) yes
checking for memory.h... (cached) yes
checking for strings.h... (cached) yes
checking for inttypes.h... (cached) yes
checking for stdint.h... (cached) no
checking for unistd.h... (cached) yes
checking for host-specific issues... none
checking for dirent.h that defines DIR... yes
checking for library containing opendir... none required
checking whether sys/types.h defines makedev... yes
checking whether time.h and sys/time.h may both be included... yes
checking whether stat file-mode macros are broken... no
checking for sys_siglist declaration in signal.h or unistd.h... yes
checking for stddef.h... yes
checking for sys/vfs.h... no
checking for sys/select.h... yes
checking for sys/socket.h... yes
checking for netinet/in.h... yes
checking for regex.h... yes
checking for glob.h... yes
checking for paths.h... yes
checking utmpx.h usability... no
checking utmpx.h presence... no
checking for utmpx.h... no
checking for struct statfs.f_flags... no
checking for special C compiler options needed for large files... no
checking for _FILE_OFFSET_BITS value needed for large files... no
checking for _LARGE_FILES value needed for large files... no
checking for strftime... yes
checking for memcmp... yes
checking for memcpy... yes
checking for memmove... yes
checking for memset... yes
checking for getpwent... yes
checking for endpwent... yes
checking for gettimeofday... yes
checking for strlcat... yes
checking for strlcpy... yes
checking for strstr... yes
checking for strchr... yes
checking for strerror... yes
checking for strsignal... yes
checking for seteuid... yes
checking for setreuid... yes
checking for setresuid... no
checking for lstat... yes
checking for getwd... yes
checking for getcwd... yes
checking for ptrace... yes
checking for usleep... yes
checking for setpriority... yes
checking for inet_aton... yes
checking for gethostbyname... yes
checking for setutent... no
checking for setrlimit... yes
checking for uname... yes
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
checking for statfs... yes
checking for va_copy()... no
checking for __va_copy()... no
checking whether va_lists can be copied by value... yes
checking for vsnprintf... yes
checking for working vsnprintf... (cached) yes
checking for mlock... yes
checking whether mlock is broken... no
checking how to get filesystem type... 4.4BSD/OSF
checking for gethostbyname in -lnsl... no
checking for socket in -lsocket... no
checking for gethostbyname in -lnsl... (cached) no
checking for res_search in -lsocket... no
checking for res_search in -lresolv... yes
checking for working long double with more range or precision than
double... no
checking for long long typedef... yes
checking for ptrdiff_t... yes
checking for size_t... yes
checking for char *... yes
checking size of char *... 4
checking for size_t... (cached) yes
checking size of size_t... 4
checking for unsigned long... yes
checking size of unsigned long... 4
checking for unsigned int ... yes
checking size of unsigned int ... 4
checking for unsigned short... yes
checking size of unsigned short... 2
checking whether struct tm is in sys/time.h or time.h... time.h
checking whether struct stat has a st_flags field... yes
checking whether st_flags field is useful... yes
checking for gcc option to accept ANSI C... none needed
checking for inline... inline
checking for an ANSI C-conforming const... yes
checking whether byte ordering is bigendian... no
checking signal.h usability... yes
checking signal.h presence... yes
checking for signal.h... yes
checking for SI_USER in signal.h... yes
checking for SA_SIGINFO in signal.h... yes
checking whether sa_sigaction is supported... yes
checking for mpz_init in -lgmp... yes
checking gmp.h usability... yes
checking gmp.h presence... yes
checking for gmp.h... yes
checking which random module to use... default
checking whether /dev/random exists... yes
checking for user yule... yes
checking base key setting .. collecting entropy... 1355145055
1883250792
checking key position... 2
configure: creating ./config.status
config.status: creating Makefile
config.status: creating samhain-install.sh
config.status: creating samhain.spec
config.status: creating init/samhain.startLSB
```

```
config.status: creating init/samhain.startLinux
config.status: creating init/samhain.startGentoo
config.status: creating init/samhain.startFreeBSD
config.status: creating init/samhain.startSolaris
config.status: creating init/samhain.startHPUX
config.status: creating init/samhain.startIRIX
config.status: creating rules.deb
config.status: creating deploy.sh
config.status: creating scripts/samhain.spec
config.status: creating scripts/redhat_i386.client.spec
config.status: creating config.h
config.status: executing default commands
 samhain has been configured as follows:
     System binaries: /usr/local/sbin
  Configuration file: /etc/yulerc
        Manual pages: /usr/local/man
                Data: /var/lib/yule
            PID file: /var/run/yule.pid
            Log file: /var/log/yule/yule_log
            Base key: 1355145055,1883250792
```

Configuration is done, now compile the server

```
chief# make
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer  -o encode ./encode.c
encode 0 config.h
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer  -o samhain_setpwd ./samhain_setpwd.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer  -o mkhdr ./mkhdr.c
creating sh_MK.h
./encode 0 samhain.c  -->  x_samhain.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_SERVER -o samhain.o -c x_samhain.c
./encode 0 sh_unix.c  -->  x_sh_unix.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_SERVER -o sh_unix.o -c x_sh_unix.c
./encode 0 sh_utils.c  -->  x_sh_utils.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_SERVER -o sh_utils.o -c x_sh_utils.c
./encode 0 sh_error.c  -->  x_sh_error.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_SERVER -o sh_error.o -c x_sh_error.c
./encode 0 sh_files.c  -->  x_sh_files.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_SERVER -o sh_files.o -c x_sh_files.c
./encode 0 sh_getopt.c  -->  x_sh_getopt.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_SERVER -o sh_getopt.o -c x_sh_getopt.c
./encode 0 sh_readconf.c  -->  x_sh_readconf.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_SERVER -o sh_readconf.o -c x_sh_readconf.c
./encode 0 sh_tiger0.c  -->  x_sh_tiger0.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_SERVER -o sh_tiger0.o -c x_sh_tiger0.c
```

```
./encode 0 sh_tiger1.c  -->  x_sh_tiger1.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_SERVER -o sh_tiger1.o -c x_sh_tiger1.c
./encode 0 sh_tiger2.c  -->  x_sh_tiger2.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_SERVER -o sh_tiger2.o -c x_sh_tiger2.c
./encode 0 sh_tiger1_64.c  -->  x_sh_tiger1_64.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_SERVER -o sh_tiger1_64.o -c
x_sh_tiger1_64.c
./encode 0 sh_tiger2_64.c  -->  x_sh_tiger2_64.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_SERVER -o sh_tiger2_64.o -c
x_sh_tiger2_64.c
./encode 0 sh_hash.c  -->  x_sh_hash.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_SERVER -o sh_hash.o -c x_sh_hash.c
./encode 0 sh_mail.c  -->  x_sh_mail.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_SERVER -o sh_mail.o -c x_sh_mail.c
./encode 0 sh_mem.c  -->  x_sh_mem.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_SERVER -o sh_mem.o -c x_sh_mem.c
./encode 0 sh_entropy.c  -->  x_sh_entropy.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_SERVER -o sh_entropy.o -c x_sh_entropy.c
./encode 0 sh_forward.c  -->  x_sh_forward.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_SERVER -o sh_forward.o -c x_sh_forward.c
./encode 0 sh_modules.c  -->  x_sh_modules.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_SERVER -o sh_modules.o -c x_sh_modules.c
./encode 0 sh_utmp.c  -->  x_sh_utmp.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_SERVER -o sh_utmp.o -c x_sh_utmp.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSYSTEMMAP="/boot/System.map" -o kern_head
./kern_head.c -lgmp  -lresolv
./kern_head > sh_ks.h
encode 0 sh_ks.h
./encode 0 sh_kern.c  -->  x_sh_kern.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_SERVER -o sh_kern.o -c x_sh_kern.c
./encode 0 sh_suidchk.c  -->  x_sh_suidchk.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_SERVER -o sh_suidchk.o -c x_sh_suidchk.c
./encode 0 sh_srp.c  -->  x_sh_srp.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_SERVER -o sh_srp.o -c x_sh_srp.c
./encode 0 sh_fifo.c  -->  x_sh_fifo.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_SERVER -o sh_fifo.o -c x_sh_fifo.c
./encode 0 sh_tools.c  -->  x_sh_tools.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_SERVER -o sh_tools.o -c x_sh_tools.c
./encode 0 sh_html.c  -->  x_sh_html.c
```

```
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_SERVER -o sh_html.o -c x_sh_html.c
./encode 0 sh_gpg.c  -->  x_sh_gpg.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_SERVER -o sh_gpg.o -c x_sh_gpg.c
./encode 0 sh_cat.c  -->  x_sh_cat.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_SERVER -o sh_cat.o -c x_sh_cat.c
./encode 0 sh_calls.c  -->  x_sh_calls.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_SERVER -o sh_calls.o -c x_sh_calls.c
./encode 0 sh_extern.c  -->  x_sh_extern.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_SERVER -o sh_extern.o -c x_sh_extern.c
./encode 0 sh_database.c  -->  x_sh_database.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_SERVER -o sh_database.o -c x_sh_database.c
./encode 0 sh_err_log.c  -->  x_sh_err_log.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_SERVER -o sh_err_log.o -c x_sh_err_log.c
./encode 0 sh_err_console.c  -->  x_sh_err_console.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_SERVER -o sh_err_console.o -c
x_sh_err_console.c
./encode 0 sh_err_syslog.c  -->  x_sh_err_syslog.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_SERVER -o sh_err_syslog.o -c
x_sh_err_syslog.c
./encode 0 sh_schedule.c  -->  x_sh_schedule.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_SERVER -o sh_schedule.o -c x_sh_schedule.c
./encode 0 bignum.c  -->  x_bignum.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_SERVER -o bignum.o -c x_bignum.c
./encode 0 trustfile.c  -->  x_trustfile.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_SERVER -o trustfile.o -c x_trustfile.c
./encode 0 rijndael-alg-fst.c  -->  x_rijndael-alg-fst.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_SERVER -o rijndael-alg-fst.o -c
x_rijndael-alg-fst.c
./encode 0 rijndael-api-fst.c  -->  x_rijndael-api-fst.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_SERVER -o rijndael-api-fst.o -c
x_rijndael-api-fst.c
./encode 0 slib.c  -->  x_slib.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_SERVER -o slib.o -c x_slib.c
./encode 0 zAVLTree.c  -->  x_zAVLTree.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_SERVER -o zAVLTree.o -c x_zAVLTree.c
gcc  -O -static -o yule -static samhain.o sh_unix.o sh_utils.o
sh_error.o sh_files.o  sh_getopt.o sh_readconf.o sh_tiger0.o
sh_tiger1.o sh_tiger2.o  sh_tiger1_64.o sh_tiger2_64.o sh_hash.o
sh_mail.o sh_mem.o  sh_entropy.o sh_forward.o sh_modules.o sh_utmp.o
sh_kern.o  sh_suidchk.o sh_srp.o sh_fifo.o sh_tools.o sh_html.o
sh_gpg.o  sh_cat.o sh_calls.o sh_extern.o sh_database.o sh_err_log.o
```

```
sh_err_console.o sh_err_syslog.o sh_schedule.o bignum.o  trustfile.o
rijndael-alg-fst.o rijndael-api-fst.o slib.o  zAVLTree.o -lgmp  -
lresolv
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer  -o sstrip ./sstrip.c
```

Now install the Yule server

```
chief# make install
 /usr/bin/install -c -s -m 700 samhain_setpwd
/usr/local/sbin/yule_setpwd
 ./sstrip /usr/local/sbin/yule_setpwd
 /usr/bin/install -c -s -m 700 yule /usr/local/sbin/yule
 ./sstrip /usr/local/sbin/yule
/bin/sh ./mkinstalldirs /usr/local/man/man8
/bin/sh ./mkinstalldirs /usr/local/man/man5
  /usr/bin/install -c -m 644 ./samhain.8 /usr/local/man/man8/yule.8
  /usr/bin/install -c -m 644 ./samhainrc.5 /usr/local/man/man5/yulerc.5
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_SERVER -DSH_IDENT=\"yule\" -DTRUST_MAIN -
DSL_ALWAYS_TRUSTED=0 -o trustfile ./trustfile.c
mkdir /var/log/yule
mkdir /var/lib/yule
./samhain-install.sh --destdir= --express --verbose install-data
  cp yulerc samhainrc.pre
  mv -f samhainrc.pre samhainrc.install
  /usr/bin/install -c -m 600 samhainrc.install /etc/yulerc
  chown yule /etc/yulerc
  checking whether paths are trustworthy
  configuration file /etc/yulerc ... OK
  state directory /var/run ... OK
  state directory /var/log/yule ... OK
  data directory /var/lib/yule ... OK
  ------------------------------------------------------
  The server will run as user yule if started with
  root privileges, otherwise as the user of the parent
  process (use --enable-identity=USER to change).
  You may want to use: make install-user
    - to add the user yule (if not existing already)
    - to chown the data            directory /var/lib/yule
    - to chown the log file        directory /var/log/yule
    - to chown the configuration   file      /etc/yulerc
  ------------------------------------------------------
  You can use 'samhain-install.sh uninstall' for uninstalling
  i.e. you might consider saving that script for future use
  Use 'make install-boot' if you want yule to start on system boot
```

Configure Yule to use the 'yule' account

```
chief# make install-user
  chown root /var/lib/yule
  chmod 755 /var/lib/yule
  chown yule /etc/yulerc
  chown yule /var/log/yule
chief#
```

Adding Yule to the boot configuration is not supported for OpenBSD, so we do it by hand here. Just adding a few lines to the end of rc.local is sufficient. (Note, end the input to the 'cat' command with a <control>-D)

```
chief# cat >> /etc/rc.local
# Start Yule (Samhain) daemon
if [ -x /usr/local/sbin/yule ]; then
   echo -n ' yule';   /usr/local/sbin/yule -D
chief#
```

Now we arrange to rotate the Yule log file using the OpenBSD 'newsyslog' command. We add one line to the end of the newsyslog config file (it's wrapped here).

```
chief# cat >> /etc/newsyslog.com
/var/log/yule/yule_log   yule.wheel     640  2    10000  *     Z
/var/run/yule.pid SIGABRT
```

The basic install of yule is completed. Now we need to install the Samhain checker on the Yule server. For ease of future maintenance, we'll configure the Samhain checker to obtain its configuration and database files 'remotely' from the machine it's running on. This will allow maintenance on the checker using the same techniques that will be used for maintenance on the other client Samhain checkers.

We build Samhain from the same distribution we used to build Yule. So first we clean the remnants from building Yule.

```
chief# make clean
rm -f core *.o
test -z "encode config_xor.h depend-gen sh_ks.h sh_ks_xor.h kern_head
internal.h sh_MK.h trustfile sstrip" || rm -f encode config_xor.h
depend-gen sh_ks.h sh_ks_xor.h kern_head  internal.h sh_MK.h trustfile
sstrip
chief#
```

Configure Samhain. Under OpenBSD the 'enable-pcheck' option causes Samhain to crash, so we exclude that option. (Note: This 'configure' command was typed as one line, but is displayed here word-wrapped.)

```
chief# ./configure --enable-network=client --enable-static --enable-
suidcheck --enable-login-watch --enable-db-reload --with-
gpg=/usr/local/bin/gpg --with-logserver=127.0.0.1 --with-config-
file=REQ_FROM_SERVER/etc/samhainrc --with-data-
file=REQ_FROM_SERVER/etc/data.samhain
checking for a BSD-compatible install... /usr/bin/install -c
checking whether make sets ${MAKE}... yes
checking build system type... i386-unknown-openbsd3.1
checking host system type... i386-unknown-openbsd3.1
checking for gcc... gcc
checking for C compiler default output... a.out
checking whether the C compiler works... yes
checking whether we are cross compiling... no
checking for suffix of executables...
```

```
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking how to run the C preprocessor... gcc -E
checking for a BSD-compatible install... /usr/bin/install -c
checking whether ln -s works... yes
checking for gawk... no
checking for mawk... no
checking for nawk... nawk
checking for hostname... /bin/hostname
checking for ANSI C header files... yes
checking for sys/types.h... yes
checking for sys/stat.h... yes
checking for stdlib.h... yes
checking for string.h... yes
checking for memory.h... yes
checking for strings.h... yes
checking for inttypes.h... yes
checking for stdint.h... no
checking for unistd.h... yes
checking for sys/types.h... (cached) yes
checking for sys/stat.h... (cached) yes
checking for stdlib.h... (cached) yes
checking for string.h... (cached) yes
checking for memory.h... (cached) yes
checking for strings.h... (cached) yes
checking for inttypes.h... (cached) yes
checking for stdint.h... (cached) no
checking for unistd.h... (cached) yes
checking for host-specific issues... none
checking for dirent.h that defines DIR... yes
checking for library containing opendir... none required
checking whether sys/types.h defines makedev... yes
checking whether time.h and sys/time.h may both be included... yes
checking whether stat file-mode macros are broken... no
checking for sys_siglist declaration in signal.h or unistd.h... yes
checking for stddef.h... yes
checking for sys/vfs.h... no
checking for sys/select.h... yes
checking for sys/socket.h... yes
checking for netinet/in.h... yes
checking for regex.h... yes
checking for glob.h... yes
checking for paths.h... yes
checking utmpx.h usability... no
checking utmpx.h presence... no
checking for utmpx.h... no
checking for struct statfs.f_flags... no
checking for special C compiler options needed for large files... no
checking for _FILE_OFFSET_BITS value needed for large files... no
checking for _LARGE_FILES value needed for large files... no
checking for strftime... yes
checking for memcmp... yes
checking for memcpy... yes
checking for memmove... yes
checking for memset... yes
checking for getpwent... yes
```

```
checking for endpwent... yes
checking for gettimeofday... yes
checking for strlcat... yes
checking for strlcpy... yes
checking for strstr... yes
checking for strchr... yes
checking for strerror... yes
checking for strsignal... yes
checking for seteuid... yes
checking for setreuid... yes
checking for setresuid... no
checking for lstat... yes
checking for getwd... yes
checking for getcwd... yes
checking for ptrace... yes
checking for usleep... yes
checking for setpriority... yes
checking for inet_aton... yes
checking for gethostbyname... yes
checking for setutent... no
checking for setrlimit... yes
checking for uname... yes
checking for statfs... yes
checking for va_copy()... no
checking for __va_copy()... no
checking whether va_lists can be copied by value... yes
checking for vsnprintf... yes
checking for working vsnprintf... (cached) yes
checking for mlock... yes
checking whether mlock is broken... no
checking how to get filesystem type... 4.4BSD/OSF
checking for gethostbyname in -lnsl... no
checking for socket in -lsocket... no
checking for gethostbyname in -lnsl... (cached) no
checking for res_search in -lsocket... no
checking for res_search in -lresolv... yes
checking for working long double with more range or precision than
double... no
checking for long long typedef... yes
checking for ptrdiff_t... yes
checking for size_t... yes
checking for char *... yes
checking size of char *... 4
checking for size_t... (cached) yes
checking size of size_t... 4
checking for unsigned long... yes
checking size of unsigned long... 4
checking for unsigned int ... yes
checking size of unsigned int ... 4
checking for unsigned short... yes
checking size of unsigned short... 2
checking whether struct tm is in sys/time.h or time.h... time.h
checking whether struct stat has a st_flags field... yes
checking whether st_flags field is useful... yes
checking for gcc option to accept ANSI C... none needed
checking for inline... inline
checking for an ANSI C-conforming const... yes
```

```
checking whether byte ordering is bigendian... no
checking signal.h usability... yes
checking signal.h presence... yes
checking for signal.h... yes
checking for SI_USER in signal.h... yes
checking for SA_SIGINFO in signal.h... yes
checking whether sa_sigaction is supported... yes
checking for mpz_init in -lgmp... yes
checking gmp.h usability... yes
checking gmp.h presence... yes
checking for gmp.h... yes
checking which random module to use... default
checking whether /dev/random exists... yes
checking for user samhain... no
checking for user daemon... yes
checking base key setting .. collecting entropy... 1882964237
1885642707
checking key position... 2
configure: creating ./config.status
config.status: creating Makefile
config.status: creating samhain-install.sh
config.status: creating samhain.spec
config.status: creating init/samhain.startLSB
config.status: creating init/samhain.startLinux
config.status: creating init/samhain.startGentoo
config.status: creating init/samhain.startFreeBSD
config.status: creating init/samhain.startSolaris
config.status: creating init/samhain.startHPUX
config.status: creating init/samhain.startIRIX
config.status: creating rules.deb
config.status: creating deploy.sh
config.status: creating scripts/samhain.spec
config.status: creating scripts/redhat_i386.client.spec
config.status: creating config.h
config.status: executing default commands
 samhain has been configured as follows:
     System binaries: /usr/local/sbin
  Configuration file: REQ_FROM_SERVER/etc/samhainrc
        Manual pages: /usr/local/man
                Data: /etc
            PID file: /var/run/samhain.pid
            Log file: /var/log/samhain_log
            Base key: 1882964237,1885642707
chief#
```

Save the Base key for use later on.

```
chief# echo '1882964237,1885642707' > ~/save-this-key
```

Build the Samhain client

```
chief# make
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer  -o encode ./encode.c
encode 0 config.h
```

```
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer  -o samhain_setpwd ./samhain_setpwd.c
./config.status
config.status: creating Makefile
config.status: creating samhain-install.sh
config.status: creating samhain.spec
config.status: creating init/samhain.startLSB
config.status: creating init/samhain.startLinux
config.status: creating init/samhain.startGentoo
config.status: creating init/samhain.startFreeBSD
config.status: creating init/samhain.startSolaris
config.status: creating init/samhain.startHPUX
config.status: creating init/samhain.startIRIX
config.status: creating rules.deb
config.status: creating deploy.sh
config.status: creating scripts/samhain.spec
config.status: creating scripts/redhat_i386.client.spec
config.status: creating config.h
config.status: config.h is unchanged
config.status: executing default commands
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer  -o mkhdr ./mkhdr.c
creating sh_MK.h
./encode 0 samhain.c  -->  x_samhain.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o samhain.o -c x_samhain.c
./encode 0 sh_unix.c  -->  x_sh_unix.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_unix.o -c x_sh_unix.c
./encode 0 sh_utils.c  -->  x_sh_utils.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_utils.o -c x_sh_utils.c
./encode 0 sh_error.c  -->  x_sh_error.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_error.o -c x_sh_error.c
./encode 0 sh_files.c  -->  x_sh_files.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_files.o -c x_sh_files.c
./encode 0 sh_getopt.c  -->  x_sh_getopt.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_getopt.o -c x_sh_getopt.c
./encode 0 sh_readconf.c  -->  x_sh_readconf.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_readconf.o -c x_sh_readconf.c
./encode 0 sh_tiger0.c  -->  x_sh_tiger0.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_tiger0.o -c x_sh_tiger0.c
./encode 0 sh_tiger1.c  -->  x_sh_tiger1.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_tiger1.o -c x_sh_tiger1.c
./encode 0 sh_tiger2.c  -->  x_sh_tiger2.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_tiger2.o -c x_sh_tiger2.c
./encode 0 sh_tiger1_64.c  -->  x_sh_tiger1_64.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_tiger1_64.o -c
x_sh_tiger1_64.c
```

```
./encode 0 sh_tiger2_64.c  -->  x_sh_tiger2_64.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_tiger2_64.o -c
x_sh_tiger2_64.c
./encode 0 sh_hash.c  -->  x_sh_hash.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_hash.o -c x_sh_hash.c
./encode 0 sh_mail.c  -->  x_sh_mail.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_mail.o -c x_sh_mail.c
./encode 0 sh_mem.c  -->  x_sh_mem.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_mem.o -c x_sh_mem.c
./encode 0 sh_entropy.c  -->  x_sh_entropy.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_entropy.o -c x_sh_entropy.c
./encode 0 sh_forward.c  -->  x_sh_forward.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_forward.o -c x_sh_forward.c
./encode 0 sh_modules.c  -->  x_sh_modules.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_modules.o -c x_sh_modules.c
./encode 0 sh_utmp.c  -->  x_sh_utmp.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_utmp.o -c x_sh_utmp.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSYSTEMMAP="/boot/System.map" -o kern_head
./kern_head.c -lgmp  -lresolv
./kern_head > sh_ks.h
encode 0 sh_ks.h
./encode 0 sh_kern.c  -->  x_sh_kern.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_kern.o -c x_sh_kern.c
./encode 0 sh_suidchk.c  -->  x_sh_suidchk.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_suidchk.o -c x_sh_suidchk.c
x_sh_suidchk.c: In function `filesystem_type_uncached':
x_sh_suidchk.c:1029: warning: unused variable `flags'
./encode 0 sh_srp.c  -->  x_sh_srp.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_srp.o -c x_sh_srp.c
./encode 0 sh_fifo.c  -->  x_sh_fifo.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_fifo.o -c x_sh_fifo.c
./encode 0 sh_tools.c  -->  x_sh_tools.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_tools.o -c x_sh_tools.c
./encode 0 sh_html.c  -->  x_sh_html.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_html.o -c x_sh_html.c
./encode 0 sh_gpg.c  -->  x_sh_gpg.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_gpg.o -c x_sh_gpg.c
./encode 0 sh_cat.c  -->  x_sh_cat.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_cat.o -c x_sh_cat.c
./encode 0 sh_calls.c  -->  x_sh_calls.c
```

```
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_calls.o -c x_sh_calls.c
./encode 0 sh_extern.c  -->  x_sh_extern.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_extern.o -c x_sh_extern.c
./encode 0 sh_database.c  -->  x_sh_database.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_database.o -c x_sh_database.c
./encode 0 sh_err_log.c  -->  x_sh_err_log.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_err_log.o -c x_sh_err_log.c
./encode 0 sh_err_console.c  -->  x_sh_err_console.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_err_console.o -c
x_sh_err_console.c
./encode 0 sh_err_syslog.c  -->  x_sh_err_syslog.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_err_syslog.o -c
x_sh_err_syslog.c
./encode 0 sh_schedule.c  -->  x_sh_schedule.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_schedule.o -c x_sh_schedule.c
./encode 0 bignum.c  -->  x_bignum.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o bignum.o -c x_bignum.c
./encode 0 trustfile.c  -->  x_trustfile.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o trustfile.o -c x_trustfile.c
./encode 0 rijndael-alg-fst.c  -->  x_rijndael-alg-fst.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o rijndael-alg-fst.o -c
x_rijndael-alg-fst.c
./encode 0 rijndael-api-fst.c  -->  x_rijndael-api-fst.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o rijndael-api-fst.o -c
x_rijndael-api-fst.c
./encode 0 slib.c  -->  x_slib.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o slib.o -c x_slib.c
./encode 0 zAVLTree.c  -->  x_zAVLTree.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o zAVLTree.o -c x_zAVLTree.c
gcc  -O -static -o samhain -static samhain.o sh_unix.o sh_utils.o
sh_error.o sh_files.o  sh_getopt.o sh_readconf.o sh_tiger0.o
sh_tiger1.o sh_tiger2.o  sh_tiger1_64.o sh_tiger2_64.o sh_hash.o
sh_mail.o sh_mem.o  sh_entropy.o sh_forward.o sh_modules.o sh_utmp.o
sh_kern.o  sh_suidchk.o sh_srp.o sh_fifo.o sh_tools.o sh_html.o
sh_gpg.o  sh_cat.o sh_calls.o sh_extern.o sh_database.o sh_err_log.o
sh_err_console.o sh_err_syslog.o sh_schedule.o bignum.o  trustfile.o
rijndael-alg-fst.o rijndael-api-fst.o slib.o  zAVLTree.o -lgmp  -
lresolv
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer  -o sstrip ./sstrip.c
```

Install Samhain

```
chief# make install
/usr/bin/install -c -s -m 700 samhain_setpwd
/usr/local/sbin/samhain_setpwd
 ./sstrip /usr/local/sbin/samhain_setpwd
 /usr/bin/install -c -s -m 700 samhain /usr/local/sbin/samhain
 ./sstrip /usr/local/sbin/samhain
/bin/sh ./mkinstalldirs /usr/local/man/man8
/bin/sh ./mkinstalldirs /usr/local/man/man5
  /usr/bin/install -c -m 644 ./samhain.8 /usr/local/man/man8/samhain.8
  /usr/bin/install -c -m 644 ./samhainrc.5
/usr/local/man/man5/samhainrc.5
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -DSH_IDENT=\"daemon\" -DTRUST_MAIN
-DSL_ALWAYS_TRUSTED=0 -o trustfile ./trustfile.c
./samhain-install.sh --destdir= --express --verbose install-data
  cp samhainrc.freebsd samhainrc
You need to sign the config file now
  /usr/local/bin/gpg -a --clearsign samhainrc
gpg: no default secret key: secret key not available
gpg: samhainrc: clearsign failed: secret key not available
./samhain-install.sh: ERROR: cannot find signed file samhainrc.asc
 ERROR: Failed to install the configuration file to
REQ_FROM_SERVER/etc/samhainrc. You need to install the configuration
file manually.
  You can use 'samhain-install.sh uninstall' for uninstalling
  i.e. you might consider saving that script for future use
  Use 'make install-boot' if you want samhain to start on system boot

chief#
```

The Samhain install procedure changes the permission on /etc, causing some other utilities to break. Here we fix the problem.

```
chief# ls -ld /etc/
drwx------  22 root  wheel  3072 Apr 26 16:14 /etc/
chief# chmod 755 /etc
```

Sign and install the Samhain configuration file. We put it where Yule will be able to find it so that Yule can serve it to the Samhain client when requested.

```
chief# gpg -a --clearsign --homedir ~dela/.gnupg samhainrc

You need a passphrase to unlock the secret key for
user: "Del Armstrong (Work) <del_armstrong@some-domain.com>"
1024-bit DSA key, ID DBF5B7A2, created 2000-10-20

chief#
chief# cp ./samhainrc.asc /var/lib/yule/rc.chief.sekurity.org
chief#
```

Install the shared secret used by Samhain and Yule to securely communicate. First generate a password.

```
chief# /usr/local/sbin/yule --gen-password
85117D8F397EC0F4
```

Next, install the password in the Samhain executable. (Note: This command was typed as one line, but is shown word-wrapped.)

```
chief# /usr/local/sbin/samhain_setpwd /usr/local/sbin/samhain new
85117D8F397EC0F4
INFO   old password found
INFO   replaced:  f7c312aaaa12c3f7  by:  85117d8f397ec0f4
INFO   finished
chief# ls -l /usr/local/sbin/samhain*
-rwx------  1 root  wheel  487424 Apr 26 16:21 /usr/local/sbin/samhain
-rwx------  1 root  wheel  487424 Apr 26 16:33
/usr/local/sbin/samhain.new
-rwx------  1 root  wheel   12288 Apr 26 16:21
/usr/local/sbin/samhain_setpwd
chief# mv /usr/local/sbin/samhain.new /usr/local/sbin/samhain
```

Finally, install the same key (converted to an ASCII format) in the Yule configuration file. (Note: This command was typed as one line, but is shown word-wrapped.)

```
chief# /usr/local/sbin/yule -P 85117D8F397EC0F4 | sed
s/HOSTNAME/chief.some-domain.com/ >> /etc/yulerc
```

Now start Yule, and check the logs to confirm that it's running OK

```
chief# /usr/local/sbin/yule -D
chief# ls -l /var/log/yule
total 3
-rw-r-----  1 yule  wheel  447 Apr 26 16:42 yule.html
-rw-r-----  1 yule  wheel  647 Apr 26 16:41 yule_log
-rw-r--r--  1 yule  wheel    5 Apr 26 16:41 yule_log.lock
chief# cat /var/log/yule/yule_log
[SOF]
ERROR  :  [2003-04-26T16:41:53-0600] msg=<Service failure>,
service=<console>, obj=</dev/console>
021DF17A03822C7E0F88DB8CBE439A6029C89EF829B28920[2003-04-26T16:41:53-
0600]
ALERT  :  [2003-04-26T16:41:53-0600] msg=<START>, program=<Yule>,
userid=<1004>, path=</etc/yulerc>,
hash=<08711294D18B0FDC707C9624DB40BCD82BFA187DDDC075A1>
3CAB3A6D94AD2FBDC43EB2BD3807C569F9B19C294BA32F6F
WARN   :  [2003-04-26T16:41:53-0600] msg=<Using insecure memory>
5657BE44E09347212B7B4E553DB4279FE8A2C8A6C2847FAF
```

(Note: the next line is the one we're looking for)

```
MARK   :  [2003-04-26T16:41:53-0600] msg=<Server up, simultaneous
connections: 57>, socket_id=<3>
0D999F4F020A706BD40FB507E4BF369D4AF0B1E1CDB17243
chief#
```

Now that Yule is running, we need to get Samhain running on the Yule server. First we tweak the default Samhain configuration file.

Edit the configuration file, and re-sign it.

```
chief# emacs /var/lib/yule/rc.chief.some-domain.com
chief# gpg -a --clearsign --homedir ~dela/.gnupg
/var/lib/yule/rc.chief.some-domain.com
You need a passphrase to unlock the secret key for
user: "Del Armstrong (Work) <del_armstrong@some-domain.com>"
1024-bit DSA key, ID DB3A37A2, created 2000-10-20
Enter passphrase:

chief# mv /var/lib/yule/rc.chief.some-domain.com.asc
/var/lib/yule/rc.chief.some-domain.com
```

Now, build the initial baseline

```
chief# /usr/local/sbin/samhain -t init
INFO   :  [2003-04-26T17:15:25-0600] msg=<Downloading configuration file>
INFO   :  [2003-04-26T17:15:26-0600] msg=<Session key negotiated>
INFO   :  [2003-04-26T17:15:27-0600] msg=<File download completed>
chief#
```

The baseline is built. Sign it and put it where Yule can find it so that it will be able to serve it to the Samhain client.

```
chief#  gpg -a --clearsign --homedir ~dela/.gnupg /etc/data.samhain
You need a passphrase to unlock the secret key for
user: "Del Armstrong (Work) <del_armstrong@some-domain.com>"
1024-bit DSA key, ID DB83A7A2, created 2000-10-20
Enter passphrase:

chief# mv /etc/data.samhain.asc /var/lib/yule/file.chief.some-
domain.com
```

Now start the Samhain client

```
chief# /usr/local/sbin/samhain -t check -D
INFO   :  [2003-04-26T17:30:36-0600] msg=<Downloading configuration file>
INFO   :  [2003-04-26T17:30:37-0600] msg=<Session key negotiated>
INFO   :  [2003-04-26T17:30:38-0600] msg=<File download completed>
chief#
```

We need to tweak the Samhain config file a bit. Edit and re-sign it and then HUP the Samhain daemon to cause it to reload the config file.

```
chief# emacs /var/lib/yule/rc.chief.some-domain.com
chief# gpg -a --clearsign --homedir ~dela/.gnupg
/var/lib/yule/rc.chief.some-domain.com
You need a passphrase to unlock the secret key for
user: "Del Armstrong (Work) <del_armstrong@some-domain.com>"
1024-bit DSA key, ID DB83A7A2, created 2000-10-20
Enter passphrase:

chief# mv /var/lib/yule/rc.chief.some-domain.com.asc
/var/lib/yule/rc.chief.some-domain.com
chief# kill -HUP `cat /var/run/samhain.pid`
```

Test the configuration with a minor change.

```
chief# touch /etc/passwd
chief# date
Sat Apr 26 18:16:49 MDT 2003
```

Here we see the log message from when the Samhain daemon detected the change. Notice that it took a bit over 1.5 hours for Samhain to notice the change.

```
chief# grep /etc/passwd /var/log/yule/yule_log
<TCP>  :  [2003-04-26T19:50:18-0600] client=<chief>, msg=<DEBUG  :
[2003-04-26T19:50:17-0600] msg=<Checksum>,
chk=<A8D9FD8BF996DC4B6F73FB426F76BCE96445B0AB8788BC7D>,
path=</etc/passwd.conf>>
<TCP>  :  [2003-04-26T19:50:19-0600] client=<chief>, msg=<DEBUG  :
[2003-04-26T19:50:18-0600] msg=<Checksum>,
chk=<E114806C425B2EFF8872FCEBB47983D1D12A9EBF43A3761A>,
path=</etc/passwd>>
```

(Note: This next line is the entry we care about)

```
<TCP>  :  [2003-04-26T19:50:20-0600] client=<chief>, msg=<CRIT   :
[2003-04-26T19:50:19-0600] msg=<POLICY [ReadOnly]>, path=</etc/passwd>,
ctime_old=<[2003-04-26T21:26:01]>, ctime_new=<[2003-04-27T00:14:54]>,
mtime_old=<[2003-04-26T21:26:01]>, mtime_new=<[2003-04-27T00:14:54]>, >
```

Now we install Samhain on a different machine. This machine, named 'skipper', is a Linux workstation that will be a client of the Yule server.

Since we're using the same download used in the installations above, there is no need to verify the distribution. The distribution is already unpacked on the target machine.

```
[root@skipper samhain-1.7.6]# uname -a
Linux skipper 2.2.12-20 #1 Mon Sep 27 10:25:54 EDT 1999 i586 unknown
[root@skipper samhain-1.7.6]#
```

Begin building Samhain by running configure. The value used for 'enable-base' is the value generated when building Samhain on the Yule server. This will allow Samhain on the Yule server to verify logs generated by this copy of Samhain. We will use the same 'base' on all copies of Samhain used on this network, so that logs can all be verified by the copy of Samhain running on the Yule server.

```
[root@skipper samhain-1.7.6]# ./configure --enable-network=client --
enable-static --enable-suidcheck --enable-login-watch
--enable-db-reload --with-gpg=/usr/local/bin/gpg --with-
logserver=192.168.1.10 --with-config-file=REQ_FROM_SERVER/etc/samh
ainrc --with-data-file=REQ FROM_SERVER/etc/data.samhain --with-
kcheck=/boot/System.map-2.2.12-20 --enable-ptrace --enable-
base='1882964237,1885642707'
checking for a BSD-compatible install... /usr/bin/install -c
checking whether make sets ${MAKE}... yes
checking build system type... i586-pc-linux-gnu
checking host system type... i586-pc-linux-gnu
checking for gcc... gcc
checking for C compiler default output... a.out
```

```
checking whether the C compiler works... yes
checking whether we are cross compiling... no
checking for suffix of executables...
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking how to run the C preprocessor... gcc -E
checking for a BSD-compatible install... /usr/bin/install -c
checking whether ln -s works... yes
checking for gawk... gawk
checking for hostname... /bin/hostname
checking for ANSI C header files... yes
checking for sys/types.h... yes
checking for sys/stat.h... yes
checking for stdlib.h... yes
checking for string.h... yes
checking for memory.h... yes
checking for strings.h... yes
checking for inttypes.h... yes
checking for stdint.h... yes
checking for unistd.h... yes
checking for sys/types.h... (cached) yes
checking for sys/stat.h... (cached) yes
checking for stdlib.h... (cached) yes
checking for string.h... (cached) yes
checking for memory.h... (cached) yes
checking for strings.h... (cached) yes
checking for inttypes.h... (cached) yes
checking for stdint.h... (cached) yes
checking for unistd.h... (cached) yes
checking for host-specific issues... LINUX use ioctl to get e2fs flags
checking linux/module.h usability... yes
checking linux/module.h presence... yes
checking for linux/module.h... yes
checking linux/unistd.h usability... yes
checking linux/unistd.h presence... yes
checking for linux/unistd.h... yes
checking for dirent.h that defines DIR... yes
checking for library containing opendir... none required
checking whether sys/types.h defines makedev... yes
checking whether time.h and sys/time.h may both be included... yes
checking whether stat file-mode macros are broken... no
checking for sys_siglist declaration in signal.h or unistd.h... yes
checking for stddef.h... yes
checking for sys/vfs.h... yes
checking for sys/select.h... yes
checking for sys/socket.h... yes
checking for netinet/in.h... yes
checking for regex.h... yes
checking for glob.h... yes
checking for paths.h... yes
checking utmpx.h usability... yes
checking utmpx.h presence... yes
checking for utmpx.h... yes
checking for struct statfs.f_flags... no
checking for special C compiler options needed for large files... no
checking for _FILE_OFFSET_BITS value needed for large files... 64
```

```
checking for _LARGE_FILES value needed for large files... no
checking for strftime... yes
checking for memcmp... yes
checking for memcpy... yes
checking for memmove... yes
checking for memset... yes
checking for getpwent... yes
checking for endpwent... yes
checking for gettimeofday... yes
checking for strlcat... no
checking for strlcpy... no
checking for strstr... yes
checking for strchr... yes
checking for strerror... yes
checking for strsignal... yes
checking for seteuid... yes
checking for setreuid... yes
checking for setresuid... yes
checking for lstat... yes
checking for getwd... yes
checking for getcwd... yes
checking for ptrace... yes
checking for usleep... yes
checking for setpriority... yes
checking for inet_aton... yes
checking for gethostbyname... yes
checking for setutent... yes
checking for setrlimit... yes
checking for uname... yes
checking for statfs... yes
checking for va_copy()... no
checking for __va_copy()... yes
checking whether va_lists can be copied by value... yes
checking for vsnprintf... yes
checking for working vsnprintf... (cached) yes
checking for mlock... yes
checking whether mlock is broken... no
checking how to get filesystem type... 4.3BSD
checking for gethostbyname in -lnsl... yes
checking for socket in -lsocket... no
checking for gethostbyname in -lnsl... (cached) yes
checking for res_search in -lsocket... no
checking for res_search in -lresolv... yes
checking for working long double with more range or precision than
double... no
checking for long long typedef... yes
checking for ptrdiff_t... yes
checking for size_t... yes
checking for char *... yes
checking size of char *... 4
checking for size_t... (cached) yes
checking size of size_t... 4
checking for unsigned long... yes
checking size of unsigned long... 4
checking for unsigned int ... yes
checking size of unsigned int ... 4
checking for unsigned short... yes
```

```
checking size of unsigned short... 2
checking whether struct tm is in sys/time.h or time.h... time.h
checking whether struct stat has a st_flags field... no
checking for gcc option to accept ANSI C... none needed
checking for inline... inline
checking for an ANSI C-conforming const... yes
checking whether byte ordering is bigendian... no
checking signal.h usability... yes
checking signal.h presence... yes
checking for signal.h... yes
checking for SI_USER in signal.h... yes
checking for SA_SIGINFO in signal.h... yes
checking whether sa_sigaction is supported... no
checking for mpz_init in -lgmp... yes
checking gmp.h usability... yes
checking gmp.h presence... yes
checking for gmp.h... yes
checking which random module to use... default
checking whether /dev/random exists... yes
checking for user samhain... no
checking for user daemon... yes
checking base key setting... 1882964237 1885642707
checking key position... 3
configure: WARNING: --with-gpg: cannot determine TIGER192 checksum of
/usr/local/bin/gpg
configure: creating ./config.status
config.status: creating Makefile
config.status: creating samhain-install.sh
config.status: creating samhain.spec
config.status: creating init/samhain.startLSB
config.status: creating init/samhain.startLinux
config.status: creating init/samhain.startGentoo
config.status: creating init/samhain.startFreeBSD
config.status: creating init/samhain.startSolaris
config.status: creating init/samhain.startHPUX
config.status: creating init/samhain.startIRIX
config.status: creating rules.deb
config.status: creating deploy.sh
config.status: creating scripts/samhain.spec
config.status: creating scripts/redhat_i386.client.spec
config.status: creating config.h
config.status: executing default commands
 samhain has been configured as follows:
     System binaries: /usr/local/sbin
  Configuration file: REQ_FROM_SERVER/etc/samhainrc
        Manual pages: /usr/local/man
                Data: /etc
            PID file: /var/run/samhain.pid
            Log file: /var/log/samhain_log
            Base key: 1882964237,1885642707
```

Now that configure is done, use the 'make' command to build Samhain.

```
[root@skipper samhain-1.7.6]# make
gcc -O2 -Wall -W  -fno-strength-reduce -fno-omit-frame-pointer -o
depend-gen ./depend-gen.c
```

```
update depend.dep ...
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer  -o encode ./encode.c
encode 0 config.h
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer  -o samhain_setpwd ./samhain_setpwd.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer  -o mkhdr ./mkhdr.c
creating sh_MK.h
./encode 0 samhain.c  -->  x_samhain.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o samhain.o -c x_samhain.c
./encode 0 sh_unix.c  -->  x_sh_unix.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_unix.o -c x_sh_unix.c
./encode 0 sh_utils.c  -->  x_sh_utils.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_utils.o -c x_sh_utils.c
./encode 0 sh_error.c  -->  x_sh_error.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_error.o -c x_sh_error.c
./encode 0 sh_files.c  -->  x_sh_files.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_files.o -c x_sh_files.c
./encode 0 sh_getopt.c  -->  x_sh_getopt.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_getopt.o -c x_sh_getopt.c
./encode 0 sh_readconf.c  -->  x_sh_readconf.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_readconf.o -c x_sh_readconf.c
./encode 0 sh_tiger0.c  -->  x_sh_tiger0.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_tiger0.o -c x_sh_tiger0.c
./encode 0 sh_tiger1.c  -->  x_sh_tiger1.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_tiger1.o -c x_sh_tiger1.c
./encode 0 sh_tiger2.c  -->  x_sh_tiger2.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_tiger2.o -c x_sh_tiger2.c
./encode 0 sh_tiger1_64.c  -->  x_sh_tiger1_64.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_tiger1_64.o -c
x_sh_tiger1_64.c
./encode 0 sh_tiger2_64.c  -->  x_sh_tiger2_64.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_tiger2_64.o -c
x_sh_tiger2_64.c
./encode 0 sh_hash.c  -->  x_sh_hash.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_hash.o -c x_sh_hash.c
./encode 0 sh_mail.c  -->  x_sh_mail.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_mail.o -c x_sh_mail.c
./encode 0 sh_mem.c  -->  x_sh_mem.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_mem.o -c x_sh_mem.c
./encode 0 sh_entropy.c  -->  x_sh_entropy.c
```

```
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_entropy.o -c x_sh_entropy.c
./encode 0 sh_forward.c  -->  x_sh_forward.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_forward.o -c x_sh_forward.c
./encode 0 sh_modules.c  -->  x_sh_modules.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_modules.o -c x_sh_modules.c
./encode 0 sh_utmp.c  -->  x_sh_utmp.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_utmp.o -c x_sh_utmp.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSYSTEMMAP="/boot/System.map-2.2.12-20" -o
kern_head ./kern_head.c -lgmp -lnsl  -lnsl -lresolv
./kern_head > sh_ks.h
encode 0 sh_ks.h
./encode 0 sh_kern.c  -->  x_sh_kern.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_kern.o -c x_sh_kern.c
./encode 0 sh_suidchk.c  -->  x_sh_suidchk.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_suidchk.o -c x_sh_suidchk.c
./encode 0 sh_srp.c  -->  x_sh_srp.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_srp.o -c x_sh_srp.c
./encode 0 sh_fifo.c  -->  x_sh_fifo.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_fifo.o -c x_sh_fifo.c
./encode 0 sh_tools.c  -->  x_sh_tools.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_tools.o -c x_sh_tools.c
./encode 0 sh_html.c  -->  x_sh_html.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_html.o -c x_sh_html.c
./encode 0 sh_gpg.c  -->  x_sh_gpg.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_gpg.o -c x_sh_gpg.c
./encode 0 sh_cat.c  -->  x_sh_cat.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_cat.o -c x_sh_cat.c
./encode 0 sh_calls.c  -->  x_sh_calls.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_calls.o -c x_sh_calls.c
./encode 0 sh_extern.c  -->  x_sh_extern.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_extern.o -c x_sh_extern.c
./encode 0 sh_database.c  -->  x_sh_database.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_database.o -c x_sh_database.c
./encode 0 sh_err_log.c  -->  x_sh_err_log.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_err_log.o -c x_sh_err_log.c
./encode 0 sh_err_console.c  -->  x_sh_err_console.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_err_console.o -c
x_sh_err_console.c
./encode 0 sh_err_syslog.c  -->  x_sh_err_syslog.c
```

```
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_err_syslog.o -c
x_sh_err_syslog.c
./encode 0 sh_schedule.c  -->  x_sh_schedule.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o sh_schedule.o -c x_sh_schedule.c
./encode 0 bignum.c  -->  x_bignum.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o bignum.o -c x_bignum.c
./encode 0 trustfile.c  -->  x_trustfile.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o trustfile.o -c x_trustfile.c
./encode 0 rijndael-alg-fst.c  -->  x_rijndael-alg-fst.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o rijndael-alg-fst.o -c
x_rijndael-alg-fst.c
./encode 0 rijndael-api-fst.c  -->  x_rijndael-api-fst.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o rijndael-api-fst.o -c
x_rijndael-api-fst.c
./encode 0 slib.c  -->  x_slib.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o slib.o -c x_slib.c
./encode 0 zAVLTree.c  -->  x_zAVLTree.c
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -o zAVLTree.o -c x_zAVLTree.c
gcc  -O -static -o samhain -static samhain.o sh_unix.o sh_utils.o
sh_error.o sh_files.o sh_getopt.o sh_readconf.o sh_tiger0.o sh_tiger1.o
sh_tiger2.o sh_tiger1_64.o sh_tiger2_64.o sh_hash.o sh_mail.o sh_mem.o
sh_entropy.o sh_forward.o sh_modules.o sh_utmp.o sh_kern.o sh_suidchk.o
sh_srp.o sh_fifo.o sh_tools.o sh_html.o sh_gpg.o sh_cat.o sh_calls.o
sh_extern.o sh_database.o sh_err_log.o sh_err_console.o sh_err_syslog.o
sh_schedule.o bignum.o trustfile.o rijndael-alg-fst.o rijndael-api-
fst.o slib.o zAVLTree.o -lgmp -lnsl  -lnsl -lresolv
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer  -o sstrip ./sstrip.c
```

Now install Samhain via 'make install'

```
[root@skipper samhain-1.7.6]# make install
 /usr/bin/install -c -s -m 700 samhain_setpwd
/usr/local/sbin/samhain_setpwd
 ./sstrip /usr/local/sbin/samhain_setpwd
 /usr/bin/install -c -s -m 700 samhain /usr/local/sbin/samhain
 ./sstrip /usr/local/sbin/samhain
/bin/sh ./mkinstalldirs /usr/local/man/man8
/bin/sh ./mkinstalldirs /usr/local/man/man5
  /usr/bin/install -c -m 644 ./samhain.8 /usr/local/man/man8/samhain.8
  /usr/bin/install -c -m 644 ./samhainrc.5
/usr/local/man/man5/samhainrc.5
gcc  -DHAVE_CONFIG_H -I. -I.   -O2 -Wall -W  -fno-strength-reduce -fno-
omit-frame-pointer -DSH_WITH_CLIENT -DSH_IDENT=\"daemon\" -DTRUST_MAIN
-DSL_ALWAYS_TRUSTED=0 -o trustfile ./trustfile.c
./samhain-install.sh --destdir= --express --verbose install-data
  cp samhainrc.linux samhainrc
You need to sign the config file now
```

```
  /usr/local/bin/gpg -a --clearsign samhainrc
gpg: no default secret key: secret key not available
gpg: samhainrc: clearsign failed: secret key not available
./samhain-install.sh: ERROR: cannot find signed file samhainrc.asc
 ERROR: Failed to install the configuration file to
REQ_FROM_SERVER/etc/samhainrc. You need to install the configuration
file manually.
  You can use 'samhain-install.sh uninstall' for uninstalling
  i.e. you might consider saving that script for future use
  Use 'make install-boot' if you want samhain to start on system boot
```

We have the same problem as before, the permissions on /etc need to be fixed after the install.

```
[root@skipper samhain-1.7.6]# chmod 755 /etc
```

Install the scripts which will start Samhain when the machine is booting. This version of Linux is supported by 'install-boot' so we don't have to do this by hand.

```
[root@skipper samhain-1.7.6]# make install-boot
./samhain-install.sh --destdir= --express --verbose install-boot
  Redhat based system detected
  /usr/bin/install -c -m 700  init/samhain.startLinux
/etc/rc.d/init.d/samhain
  cd /etc/rc.d/rc2.d/ && ln -f -s /etc/rc.d/init.d/samhain S99samhain
  cd /etc/rc.d/rc2.d/ && ln -f -s /etc/rc.d/init.d/samhain K10samhain
  cd /etc/rc.d/rc3.d/ && ln -f -s /etc/rc.d/init.d/samhain S99samhain
  cd /etc/rc.d/rc3.d/ && ln -f -s /etc/rc.d/init.d/samhain K10samhain
  cd /etc/rc.d/rc4.d/ && ln -f -s /etc/rc.d/init.d/samhain S99samhain
  cd /etc/rc.d/rc4.d/ && ln -f -s /etc/rc.d/init.d/samhain K10samhain
  cd /etc/rc.d/rc5.d/ && ln -f -s /etc/rc.d/init.d/samhain S99samhain
  cd /etc/rc.d/rc5.d/ && ln -f -s /etc/rc.d/init.d/samhain K10samhain
installing init scripts completed
```

Copy the Samhain configuration file to the Yule server where the Yule daemon can find it.

```
[root@skipper ~]# scp /usr/local/src/samhain-1.7.6/samhainrc
chief:/var/lib/yule/rc.skipper.some-domain.com
root@chief's password:
samhainrc             100%
|*****************************************************************
*|  7522     00:00
```

Log into the Yule server. First tweak, sign and install the configuration file for the Samhain client.

```
chief# emacs /var/lib/yule/rc.skipper.some-domain.com
chief# gpg -a --clearsign --homedir ~dela/.gpg
/var/lib/yule/rc.skipper.some-domain.com
You need a passphrase to unlock the secret key for
user: "Del Armstrong (Work) <del_armstrong@some-domain.com>"
1024-bit DSA key, ID DB83A7A2, created 2000-10-20
Enter passphrase:
```

```
chief# mv /var/lib/yule/rc.skipper.some-domain.com.asc
/var/lib/yule/rc.skipper.some-domain.com
```

Next, we need to generate a shared secret for the client and server for use to secure their communications.

```
chief# /usr/local/sbin/yule --gen-password
80D7AAB3F245D417
```

Install this new key in the Yule server configuration file. Look at the last line of yulerc to verify the addition. We also HUP the Yule daemon to cause it to reload it's configuration file.

```
chief# /usr/local/sbin/yule -P 80D7AAB3F245D417 | sed
s/HOSTNAME/skipper.some-domain.com/ >> /etc/yulerc
chief# tail -1 /etc/yulerc
Client=skipper.some-
domain.com@FB87EE16A0DCCFA8@E7FBFF70CFB39DEEE747684EE39D294AD1EB826DC1E
14C8B1886CFD486D4515AE37FEAD5D68D22FC961F37DEF3FFD8BE7CFA7A8EDA8328411A
1AD53F7BC652B341640EB3255FF3EA8F5F3B851EB8D223CB4554F7A542D68D301DC4802
E29889E718B8FD8DCB91FD4EFE238100B22E2C125392C4EF1333A8BE4EE75DCF14046A
chief# kill -HUP `cat /var/run/yule.pid`
```

Now return to the Samhain client to finish installing the shared secret.

```
[root@skipper ~]# /usr/local/sbin/samhain_setpwd
/usr/local/sbin/samhain new 80D7AAB3F245D417
INFO   old password found
INFO   replaced:  f7c312aaaa12c3f7  by:  80d7aab3f245d417
INFO   finished
[root@skipper /root]# mv /usr/local/sbin/samhain.new
/usr/local/sbin/samhain
mv: overwrite `/usr/local/sbin/samhain'? y
[root@skipper /root]#
```

Build the intial baseline.

```
[root@skipper /root]# /usr/local/sbin/samhain -t init
INFO   :  [2003-04-26T21:21:24-0600] msg=<Downloading configuration
file>
INFO   :  [2003-04-26T21:21:25-0600] msg=<Session key negotiated>
INFO   :  [2003-04-26T21:21:26-0600] msg=<File download completed>
[root@skipper /root]#
```

Copy the basline data file to the Yule server.

```
[root@skipper /root]# scp /etc/data.samhain
chief:/var/lib/yule/file.skipper.some-domain.com
root@chief's password:
data.samhain         100%
|*****************************************************************
*|  4011 KB    00:04
```

On the yule server, sign the data file for the client.

```
chief# gpg -a --clearsign --homedir ~dela/.gnupg
/var/lib/yule/file.skipper.some-domain.com
You need a passphrase to unlock the secret key for
user: "Del Armstrong (Work) <del_armstrong@some-domain.com>"
1024-bit DSA key, ID DB83A7A2, created 2000-10-20
Enter passphrase:

chief# mv /var/lib/yule/file.skipper.some-domain.com.asc
/var/lib/yule/file.skipper.some-domain.com
```

On the Samhain client, complete the installation by starting Samhain in check mode

```
[root@skipper /root]# /usr/local/sbin/samhain -D -t check
INFO   :  [2003-04-26T22:20:25-0600] msg=<Downloading configuration
file>
INFO   :  [2003-04-26T22:20:26-0600] msg=<Session key negotiated>
INFO   :  [2003-04-26T22:20:27-0600] msg=<File download completed>
[root@skipper /root]#
```

Start installation of the Samhain client on the next machine (named 'schleicher'). The new machine is also running Linux, so the installation will consist of copying the already-build Samhain client from the first Linux client to this machine, and installing it.

First copy the executable to the new client.

```
[root@skipper /root]# scp /usr/local/sbin/samhain
schleicher:/usr/local/sbin/
samhain             100%
|*****************************************************************
*|   600 KB    00:06
[root@skipper /root]#
```

Since we're using the same executable, we can use the same 'base key' on both Linux machines. However the Yule server needs to have an entry for each client in its configuration file. So we make a copy of the line for the Linux client we just finished, and change the name of the client to that of the new Linux client.

```
chief# tail -1 /etc/yulerc | sed s/skipper/schleicher/ >> /etc/yulerc
```

HUP the Yule server so that it will find the new configuration file.

```
chief# kill -HUP `cat /var/run/yule.pid `
```

On the Yule server, make a Samhain configuration file for the new client based on the configuration file for the original Linux Client. It's already signed, so we don't have to re-sign it.

```
chief# cp /var/lib/yule/rc.skipper.some-domain.com
/var/lib/yule/rc.schleicher.some-domain.com
```

On the new client, build the initial baseline.

```
[root@schleicher /root]# /usr/local/sbin/samhain -t init
INFO   :  [2003-04-26T23:05:06-0600] msg=<Downloading configuration file>
INFO   :  [2003-04-26T23:05:09-0600] msg=<Session key negotiated>
INFO   :  [2003-04-26T23:05:10-0600] msg=<File download completed>
[root@schleicher /root]#
```

Copy the initial baseline to the Yule server.

```
[root@schleicher /root]# scp /etc/data.samhain
chief:/var/lib/yule/file.schleicher.some-domain.com
data.samhain         100%
|*****************************************************************
*|  3082 KB    00:37
```

On the Yule server, the initial baseline needs to be PGP signed.

```
chief# gpg -a --clearsign --homedir ~dela/.gnupg
/var/lib/yule/file.schleicher.some-domain.com
You need a passphrase to unlock the secret key for
user: "Del Armstrong (Work) <del_armstrong@some-domain.com>"
1024-bit DSA key, ID DB83A7A2, created 2000-10-20
Enter passphrase:

chief# mv /var/lib/yule/file.schleicher.some-domain.com.asc
/var/lib/yule/file.schleicher.some-domain.com
```

Back on the new Samhain client, add code to start the Samhain client upon booting. This machine is configured differently than the other Linux machine in this respect, so the simplest way to accomplish this is to update 'rc.local' on this machine.

```
[root@schleicher /root]# cat >> /etc/rc.d/rc.local
# Start samhain
/usr/local/sbin/samhain -D -t check
[root@schleicher /root]#
```

Finally, start the Samhain client in check mode

```
[root@schleicher /root]# /usr/local/sbin/samhain -D -t check
INFO   :  [2003-04-27T01:48:31-0600] msg=<Downloading configuration file>
INFO   :  [2003-04-27T01:48:33-0600] msg=<Session key negotiated>
INFO   :  [2003-04-27T01:48:34-0600] msg=<File download completed>
```

That completes the exemplary installation

Appendix B – Yule Server Configuration File

```
#####################################################################
#
# 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 '#' 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]
# set threshold severity for log facilities
# values: debug, info, notice, warn, mark, err, crit, alert, none.
# 'mark' is used for timestamps.
#
# 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).
#
# MailSeverity=*
# MailSeverity=!warn
# MailSeverity==crit
#
MailSeverity=crit
PrintSeverity=info
LogSeverity=warn
SyslogSeverity=none


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

# the maximum time between client messages (seconds)
# (this is a server-only option; the default is 86400 sec = 1 day
#
```

```
# SetClientTimeLimit=1800

# 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).
#
# maximum time till next mail (seconds)
SetMailTime=86400

# maximum number of pending mails
SetMailNum=10

# where to send mail to
SetMailAddress=root@localhost

# mail relay host
# SetMailRelay=relay.yourdomain.de

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

# where to get time from
SetTimeServer=localhost

# timer for time stamps
SetLoopTime=60

# trusted users (root and the effective user are always trusted)
# TrustedUser=bin


# 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:
#               'BAD_CLIENT what.ever.it.is -- unregistered host'
#
# First entry is for challenge/response, second one for SRP
authentication.
#
[Clients]
#
Client=HOSTNAME@00000000@C39F0EEFBC64E4A8BBF72349637CC07577F714B420B628
82
```

```
#
Client=HOSTNAME@8F81BA58956F8F42@8932D08C49CA76BD843C51EDD1D6640510FA03
2A7A2403E572BBDA2E5C6B753991CF7E091141D20A2499C5CD3E14C1639D17482E14E15
48E5246ACF4E7193D524CDDAC9C9D6A9A36C596B4ECC68BEB0C5BB7082224946FC98E3A
DE214EA1343E2DA8DF4229D4D8572AD8679228928A787B6E5390D3A713102FFCC9D0B21
88C92
Client=chief.some-
domain.com@2A9AD1AD86F968A5@C8A4E520F57A98E06503CF61A1323CC9D5D12215048
E3CD680A1A119AC762514343FA38B46B04CCD08A37225347C8FC0109E5C9100A0DAE27E
F26F6832AAFB1648C6DEC9995B87B3770FECA4A6527E57414DC36C0BE5C343550A0FBA1
DE20B075CE80C5F8DB86983E79D6B6625CB62F0336C1293609E2819184A80FC36B74037
Client=skipper.some-
domain.com@FB87EE16A0DCCFA8@E7FBFF70CFB39DEEE747684EE39D294AD1EB826DC1E
14C8B1886CFD486D4515AE37FEAD5D68D22FC961F37DEF3FFD8BE7CFA7A8EDA8328411A
1AD53F7BC652B341640EB3255FF3EA8F5F3B851EB8D223CB4554F7A542D68D301DC4802
E29889E718B8FD8DCB91FD4EFE238100B22E2C125392C4EF1333A8BE4EE75DCF14046A
Client=schleicher.some-
domain.com@FB87EE16A0DCCFA8@E7FBFF70CFB39DEEE747684EE39D294AD1EB826DC1E
14C8B1886CFD486D4515AE37FEAD5D68D22FC961F37DEF3FFD8BE7CFA7A8EDA8328411A
1AD53F7BC652B341640EB3255FF3EA8F5F3B851EB8D223CB4554F7A542D68D301DC4802
E29889E718B8FD8DCB91FD4EFE238100B22E2C125392C4EF1333A8BE4EE75DCF14046A
```

Appendix C – Samhain Configuration File for 'chief'

Since chief is the yule server, this configuration has been modified to report critical events out-of-band via email.

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

##################################################################
#
# FreeBSD Configuration file  for samhain.
#
##################################################################
#
# -- empty lines and lines starting with '#' are ignored
# -- boolean options can be Y/N or T/F 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)  To each policy, you can assign a severity (further below).
# (iii) 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).
##################################################################

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

file=/usr/compat/linux/etc
file=/usr/compat/linux/etc/ld.so.cache

dir=/var/mail
dir=/var/spool/lp/tmp
dir=/var/tmp
dir=/var/dt/tmp
dir=/tmp


[LogFiles]
##
## for these files, changes in signature, timestamps, and size are
ignored
```

```
##

file=/var/run/utmp


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

file=/var/log/wtmp
file=/var/log/messages
file=/var/log/maillog
file=/var/log/lastlog
file=/var/log/cron
file=/var/log/auth.log


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

dir=/usr/share/man
dir=/usr/share/games
dir=/usr/share/misc
dir=/usr/X11R6/man


[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 ...
##


[ReadOnly]
## for these files, only access time is ignored
##
dir=/bin
dir=/boot
dir=3/etc
dir=/sbin
dir=1/stand
dir=/stand/etc
dir=/stand/modules
dir=/usr
dir=2/var/cron

file=/kernel
dir=/modules

dir=4/home/usr-local/
dir=4/home/local/
```

```
[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

# 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=info

# Files : file access problems
# Dirs  : directory access problems
# Names : suspect (non-printable) characters in a pathname
SeverityFiles=crit
SeverityDirs=crit
SeverityNames=warn

[Log]
## Set threshold severity for log facilities
## Values: debug, info, notice, warn, mark, err, crit, alert, none.
## 'mark' is used for timestamps.
##
## 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).
# MailSeverity=*
# MailSeverity=!warn
# MailSeverity==crit

MailSeverity=crit
#MailSeverity=none
PrintSeverity=none
##LogSeverity=warn
LogSeverity=none
SyslogSeverity=none
#SyslogSeverity=crit
#ExportSeverity=none
ExportSeverity=debug

## Check the filesystem for SUID/SGID binaries
##[SuidCheck]
##every 24 hrs=86400
##SuidCheckActive=1
##SuidCheckInterval=86400

# Directory to exclude
```

```
#
#SuidCheckExclude=/net/localhost

# Limit on files per second
#
#SuidCheckFps=250


[Kernel]
##
## Settings this to 1/true/yes will activate the check for loadable
## kernel module rootkits (Linux only)
##
#KernelCheckActive=1
#KernelCheckInterval = 20

[Utmp]
##
## 0 to switch off, 1 to activate logging of login/logout events
##
##LoginCheckActive=0
LoginCheckActive=1

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

# interval for login/logout checks
#
LoginCheckInterval=60

[Misc]
##
## whether to become a daemon process
## (this is not honoured on database initialisation)
##
Daemon=yes

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

# Set nice level (-19 to 19, see 'man nice'),
# and I/O limit (kilobytes per second) to reduce load on host
#
#SetNiceLevel=17
#SetIOLimit=500
SetNiceLevel=17
SetIOLimit=500

# Custom format for message header
#
# %S severity
# %T timestamp
```

```
# %C class
#
# %F source file
# %L source line
#
# MessageHeader="%S %T %F:%L "

# the maximum time between client messages (seconds)
# (this is a server-only option; the default is 86400 sec = 1 day
#
# SetClientTimeLimit=1800

# time till next file check (seconds) 900=15 mins, 7200=2hrs
SetFilecheckTime=7200

# The default is 'TRUE' (= report only new policy violations).
# 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=FALSE

# 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).
#
# maximum time till next mail (seconds) 86400=1day
SetMailTime=86400

# maximum number of queued mails
SetMailNum=10

# where to send mail to
#SetMailAddress=root@localhost
SetMailAddress=del_armstrong@yahoo.com

# mail relay host
# SetMailRelay=relay.yourdomain.de
SetMailRelay=pop.dnvr.qwest.net

# Syslog: lets log to local2 not AUTHPRIV, since authlog is extensively
#         used by other daemons.
SyslogFacility=LOG_LOCAL2

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

# where to get time from
# SetTimeServer=www.yourdomain.de

# where to export logs to
SetLogServer=localhost
```

```
# timer for time stamps
SetLoopTime=20

# trusted users (root and the effective user are always trusted)
# TrustedUser=bin


# 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 v1.0.6 (OpenBSD)
Comment: For info see http://www.gnupg.org

IDBB8QE+ra+pV9bf7NufV6IRAtp2AKCFJkkWKhYsK1rSYvh4HvSo2EA+/QCfUx8J
0lIIuc6UUvHJ18BMIlbFG0Y=
=537u
-----END PGP SIGNATURE-----
```

Appendix D – Samhain Configuration File for 'skipper'

Notice that in this configuration files, old PGP signatures have been left in the file. Allowing old signatures to 'build up' has no impact, other than poor aesthetics.

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

- -----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

Hash: SHA1

# default config file for samhain
# signature checking with the setup below takes 110 seconds on a 500MHz
Celeron
#
# -- empty lines and lines starting with '#' are ignored
# -- you can PGP clearsign this file -- samhain will check (if compiled
#    with support) or otherwise ignore the signature
# -- CHECK mail address, log server IP address (if remote logging)
#
# (i)   There are several policies, each has its own section. Put files
#       into the section for the appropriate policy (see below).
# (ii)  To each policy, you can assign a severity (further below).
# (iii) 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).

[Attributes]
#
# for these files, only changes in permissions and ownership are
checked
#
file=/etc/mtab
file=/etc/ssh_random_seed
file=/etc/asound.conf
file=/etc/resolv.conf
file=/etc/localtime
file=/etc/ioctl.save
file=/etc

[LogFiles]
#
# for these files, changes in signature, timestamps, and size are
ignored
#
file=/var/run/utmp
file=/etc/motd

#
# 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.
#
@HOSTNAME
file=/foo/bar
@end

[GrowingLogFiles]
#
# for these files, changes in signature, timestamps, and increase in
size
#                  are ignored
#
file=/var/log/warn
file=/var/log/messages
file=/var/log/wtmp
file=/var/log/faillog

[IgnoreAll]
#
# for these files, no modifications are reported
#
file=/etc/resolv.conf.pcmcia.save
#
# Added 4/20/03 - DA
#
dir=-1/usr/local/src
#
# These files change every reboot, and are unlikely
# to be of use to an attacker
#
file=/etc/issue.net
file=/etc/issue
file=/etc/HOSTNAME
file=/etc/aliases.db
file=/etc/mail/mailertable.db
file=/etc/mail/domaintable.db
file=/etc/mail/access.db
file=/etc/mail/virtusertable.db

[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 ...
#

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

```
dir=/sbin
dir=/usr/sbin
dir=/lib
dir=3/etc
dir=/boot
#
# Added 4/20/03 - DA
#
dir=4/usr/local
dir=3/usr/X11
dir=4/usr/lib

[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
#
#
SeverityIgnoreAll=info
#
# Files : file access problems
# Dirs  : directory access problems
# Names : suspect (non-printable) characters in a pathname
#
SeverityFiles=crit
SeverityDirs=crit
SeverityNames=warn

[Log]
# set threshold severity for log facilities
# values: debug, info, notice, warn, mark, err, crit, alert, none.
# 'mark' is used for timestamps.
#
# 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).
#
# MailSeverity=*
# MailSeverity=!warn
# MailSeverity==crit
#
MailSeverity=none
PrintSeverity=none
LogSeverity=none
SyslogSeverity=none
ExportSeverity=info
```

```
[Utmp]
# 0 to switch off, 1 to activate
#
LoginCheckActive=0

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

# interval for login/logout checks
#
LoginCheckInterval=60

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

# the maximum time between client messages (seconds)
# (this is a log server-only option; the default is 86400 sec = 1 day
#
# SetClientTimeLimit=1800

# time till next file check (seconds)
SetFilecheckTime=600

# 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).
#
# maximum time till next mail (seconds)
SetMailTime=86400

# maximum number of pending mails
SetMailNum=10

# where to send mail to
SetMailAddress=dela@gblx.net

# mail relay host
# SetMailRelay=relay.yourdomain.de
SetMailRelay=pop.dnvr.qwest.net

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

# where to get time from
# SetTimeServer=localhost

# where to export logs to
```

```
# SetLogServer=localhost
SetLogServer=192.168.1.10

# timer for time stamps
SetLoopTime=60

# trusted users (root and the effective user are always trusted)
# TrustedUser=bin

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


# everything below is ignored
[EOF]
- -----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.0.6 (OpenBSD)
Comment: For info see http://www.gnupg.org

IDBB8QE+pDmDV9bf7NufV6IRAr7aAJ9LfSzetKZE9n+jk2r6+eu8zCXvWACeMaEZ
gFNz4kqhGTmXga6zYVagSl0=
=DwZJ
- -----END PGP SIGNATURE-----
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.0.6 (OpenBSD)
Comment: For info see http://www.gnupg.org

IDBB8QE+q0yVV9bf7NufV6IRAm39AJ9tYAWDJ6G8tAdA1L7dvUpu3TLJlwCeIGxq
lbY8qtw0nFUkhBylBpza1IM=
=UD+S
-----END PGP SIGNATURE-----
```

Appendix E – Samhain Configuration File on 'schleicher'

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

- -----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

- - -----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

Hash: SHA1

# default config file for samhain
# signature checking with the setup below takes 110 seconds on a 500MHz
Celeron
#
# -- empty lines and lines starting with '#' are ignored
# -- you can PGP clearsign this file -- samhain will check (if compiled
#    with support) or otherwise ignore the signature
# -- CHECK mail address, log server IP address (if remote logging)
#
# (i)   There are several policies, each has its own section. Put files
#       into the section for the appropriate policy (see below).
# (ii)  To each policy, you can assign a severity (further below).
# (iii) 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).

[Attributes]
#
# for these files, only changes in permissions and ownership are
checked
#
file=/etc/mtab
file=/etc/ssh_random_seed
file=/etc/asound.conf
file=/etc/resolv.conf
file=/etc/localtime
file=/etc/ioctl.save
file=/etc

[LogFiles]
#
# for these files, changes in signature, timestamps, and size are
ignored
#
file=/var/run/utmp
file=/etc/motd

#
# 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.
#
@HOSTNAME
file=/foo/bar
@end

[GrowingLogFiles]
#
# for these files, changes in signature, timestamps, and increase in
size
#                  are ignored
#
file=/var/log/warn
file=/var/log/messages
file=/var/log/wtmp
file=/var/log/faillog

[IgnoreAll]
#
# for these files, no modifications are reported
#
file=/etc/resolv.conf.pcmcia.save
#
# Added 4/20/03 - DA
#
dir=-1/usr/local/src
#
# These files change every reboot, and are unlikely
# to be of use to an attacker
#
file=/etc/issue.net
file=/etc/issue
file=/etc/HOSTNAME
file=/etc/aliases.db
file=/etc/mail/mailertable.db
file=/etc/mail/domaintable.db
file=/etc/mail/access.db
file=/etc/mail/virtusertable.db

[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 ...
#

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

```
dir=/usr/sbin
dir=/lib
dir=3/etc
dir=/boot
#
# Added 4/20/03 - DA
#
dir=4/usr/local
dir=3/usr/X11
dir=4/usr/lib

[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
#
#
SeverityIgnoreAll=info
#
# Files : file access problems
# Dirs  : directory access problems
# Names : suspect (non-printable) characters in a pathname
#
SeverityFiles=crit
SeverityDirs=crit
SeverityNames=warn

[Log]
# set threshold severity for log facilities
# values: debug, info, notice, warn, mark, err, crit, alert, none.
# 'mark' is used for timestamps.
#
# 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).
#
# MailSeverity=*
# MailSeverity=!warn
# MailSeverity==crit
#
MailSeverity=none
PrintSeverity=none
LogSeverity=none
SyslogSeverity=none
ExportSeverity=info

[Kernel]
```

```
#
# Setings this to 1/true/yes will activate the check for loadable
# kernel module rootkits (Linux only)
#
KernelCheckActive=0
KernelCheckInterval = 20



[Utmp]
# 0 to switch off, 1 to activate
#
LoginCheckActive=0

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

# interval for login/logout checks
#
LoginCheckInterval=60

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

# the maximum time between client messages (seconds)
# (this is a log server-only option; the default is 86400 sec = 1 day
#
# SetClientTimeLimit=1800

# time till next file check (seconds)
SetFilecheckTime=600

# 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).
#
# maximum time till next mail (seconds)
SetMailTime=86400

# maximum number of pending mails
SetMailNum=10

# where to send mail to
SetMailAddress=dela@gblx.net

# mail relay host
# SetMailRelay=relay.yourdomain.de
SetMailRelay=pop.dnvr.qwest.net

# The binary. Setting the path will allow
# samhain to check for modifications between
# startup and exit.
```

```
#
# SamhainPath=/usr/local/bin/samhain
SamhainPath=/usr/local/sbin/samhain

# where to get time from
# SetTimeServer=localhost

# where to export logs to
# SetLogServer=localhost
SetLogServer=192.168.1.10

# timer for time stamps
SetLoopTime=60

# trusted users (root and the effective user are always trusted)
# TrustedUser=bin

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


# everything below is ignored
[EOF]
- - -----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.0.6 (OpenBSD)
Comment: For info see http://www.gnupg.org

IDBB8QE+pDmDV9bf7NufV6IRAr7aAJ9LfSzetKZE9n+jk2r6+eu8zCXvWACeMaEZ
gFNz4kqhGTmXga6zYVagSl0=
=DwZJ
- - -----END PGP SIGNATURE-----
- -----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.0.6 (OpenBSD)
Comment: For info see http://www.gnupg.org

IDBB8QE+q0yVV9bf7NufV6IRAm39AJ9tYAWDJ6G8tAdA1L7dvUpu3TLJlwCeIGxq
lbY8qtw0nFUkhBylBpza1IM=
=UD+S
- -----END PGP SIGNATURE-----
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.0.6 (OpenBSD)
Comment: For info see http://www.gnupg.org

IDBB8QE+q2ToV9bf7NufV6IRAjf7AJ9sZdfv6BLKxC3iDb1mLSM+WdlhkgCeNPu+
YFwReelrpgUQviMfAgOXdxg=
=3Wg7
-----END PGP SIGNATURE-----
```

Last Updated: January 21st, 2019

Upcoming Training

SANS 2019	**Orlando, FL**	**Apr 01, 2019 - Apr 08, 2019**	**Live Event**
SANSFIRE 2019	**Washington, DC**	**Jun 15, 2019 - Jun 22, 2019**	**Live Event**
Community SANS New York SEC506	**New York, NY**	**Jul 15, 2019 - Jul 20, 2019**	**Community SANS**
SANS Network Security 2019	**Las Vegas, NV**	**Sep 09, 2019 - Sep 16, 2019**	**Live Event**