

# The Ins and Outs of System Logging Using Syslog

Ian Eaton

# The Ins and Outs of System Logging Using Syslog

Paper version 1.0

## GIAC Security Essentials Certification (GSEC) Practical Assignment

Assignment version 1.4b (amended August 29, 2002)
Option 1 - Permission Granted

Location of Course Work:
SANS Darling Harbour, Sydney Australia, February 2003

Prepared by: Ian Eaton

# The Ins and Outs of System Logging Using Syslog

## Table of Contents


**INTRODUCTION** ..... 3
**WHAT IS LOGGING** ..... 4
**HOW DOES IT HELP** ..... 5
**INTRODUCTION TO SYSLOG** ..... 7
**BEFORE YOU START LOGGING** ..... 8
**LOCAL LOGGING** ..... 10
**SOURCE AND LEVEL OF LOGGING** ..... 15
THE SELECTOR ..... 15
FACILITY ..... 15
PRIORITY ..... 17
**MESSAGE DESTINATION (THE ACTION FIELD)** ..... 18
**LOG RETENTION AND ROTATION** ..... 21
**LOG PARSING AND REDUCTION TOOLS** ..... 22
**SECURING THE STORED SYSLOG FILES** ..... 23
GENERAL TECHNIQUES ..... 23
HASHING ..... 24
HOST SECURITY ..... 24
**CENTRAL SYSLOG SERVER** ..... 26
**WHERE DO I HOST MY LOG FILES?** ..... 33
LOCALLY ONLY ..... 33
REMOTE SERVER ONLY ..... 33
LOCALLY AND REMOTELY ..... 34
COMMENTS ..... 34
**LEVEL AND DESTINATION OF LOGS REVISITED** ..... 35
**TIME SYNCHRONISATION** ..... 36
**CHANGING THE TRANSPORT PROTOCOL AND MESSAGE RELIABILITY** ..... 37
**INTEGRITY AND AUTHENTICATION OF MESSAGES** ..... 39
**ENCRYPTING SYSLOG TRAFFIC** ..... 41
**HOW ABOUT A SYSLOG RELAY?** ..... 42
**NETWORK FILTERS, INTRUSION DETECTION AND OOB NETWORKS** ..... 43
**CONCLUSION** ..... 45
**APPENDIX A: REFERENCES** ..... 46
**APPENDIX B: RED HAT 8.0 SYSLOG.CONF** ..... 48
**APPENDIX C: EXPLANATION OF TCPDUMP COMMAND AND OUTPUT** ..... 49
**APPENDIX D: EXPLANATION OF HPING2 COMMAND** ..... 50
**APPENDIX E: PRI MATRIX** ..... 51

## Introduction

The intent of this paper is to help the reader follow a process of thinking that will provide them with the tools to understand the fundamentals of system logging. Hopefully at the end you will be able to identify the best implementation for your particular environment.

This paper focuses on logging using syslog which has become the de facto logging standard on UNIX based systems. Though this is syslog and UNIX specific I would hope the general discussions on logging would be helpful for any log implementation.

The structure of this paper begins with a discussion on what logging is, how it helps and what considerations are needed before we implement logging. We progress towards a discussion on syslog specifics, the elements that comprise a working implementation from the basic to the more advanced, detailing configuration options and shortcomings, including implementation ideas.

The last parts of the paper builds upon our knowledge of syslog, we look at methods to remove these shortcomings, and the outside factors that need consideration to provide us with a robust and secure implementation.

Throughout this paper I have chosen to focus on the syslog implementation that comes with Red Hat[1] 8.0 which provides the main features required to perform logging but falls short when we start our discuss on security requirements. Securing syslog could be a very important consideration for your organisation, during these discussions I have chosen to use SDSC Secure Syslog from the San Diego Supercomputer Centre[2].

Reasons for choosing this replacement will become apparent as we go deeper into syslog and its inner workings.

### Definitions

For the purpose of this paper some definitions of basic syslog components need to be set[3]:

*Log device* or just *device:* This is the equipment that generates log messages, this could be a server, router, firewall, or any other device that can generate syslog event data.

*Log collector* or just *collector:* This equipment is configured to receive messages generated by a log device where it will subsequently store these externally generated messages local to itself.

*Log relay* or just *relay:* Like a log collector it is configured to receive messages, these messages are not stored but forwarded to another device, usually a log collector but could be another relay

[1]Further information can be found at http://www.redhat.com/
[2]Further information can be found at http://security.sdsc.edu/software/sdsc-syslog/
[3] These definitions are from RFC3164 "The BSD Syslog Protocol"

## What is logging

I am yet to see anything in life that runs perfectly the first time and without a single failure, the world contains so many variables that nothing can be predicted to 100% accuracy. In the world of computers it is inevitable something will fail, a user will make a mistake, an attacker will think you are an interesting target, or that easy upgrade will keep you up all night analysing what went wrong.

During the life of an application there involves a few basic steps, we install it, configure it for our particular requirements, set it running on our system, and monitor it. When an application is first installed how do we know its running, how do we know it has been configured correctly, how do we know it is still running as specified over time.

We can try some tests to see if it reacts in expected ways, we can use commands on the host system to give us clues, like if the application is using systems resources, or it is listening on a network port etc. What happens if this application does not function exactly as planned or stops functioning, its not seen to be using system resources or listening on the network, what then.

Applications are generally designed to give clues to their condition or when something changes. This change could be from being idle to running, when an application starts or finishes a particular action, or reaches a particular point in execution. These messages allow us to understand what the application is doing, and how well it's functioning.

Individual applications deliver messages differently, some provide their own inbuilt method, while others use a specialised logging application designed to handle logs from many applications. Syslog is one such application designed to accept messages from multiple applications running on the system and manipulate them based on a single configuration file.

Logging is a fundamental requirement of any system, as things will go wrong, we need a way to diagnose and isolate the cause. No matter what operating system you use one of the best locations for diagnostic information is the system log.

## How does it help

Logging gives you sight, it will provide the administrator with information on how the system is working. Without even basic logging, when something unexpected happens you could experience excessive downtime due to the lack of diagnostic data. With a correctly configured logging system you can capture such information as application, kernel and network events, as well as any other system exceptions, this can be used for many benefits:

- General network administration
- With debugging problems
- Network base line
- Network health
- Proactive system/problem analysis
- Intrusion Detection
- Incident Containment
- Forensic analysis
- Legal

### Administration and debugging

Logging is a fundamental tool for the system administrator to identify unusual activity when trying to diagnose and isolate problems, or trying to ensure systems are running as configured. In my experience, when debugging a problem for a user, I question the user to understand the nature of the problem and then I head for the system logs to look for clues. I would be looking for unexpected changed in state and any messages that indicate miss configuration.

### System baseline and health

Logging can provide more information than any one person can process during any given moment. Finding the required information can be a daunting task. This problem can be reduced if an administrator takes time with each installed system to perform a baseline audit. A baseline audit identifies what is normal activity for your system, what system log entries are normal, what network traffic the system normally generates, and how the system reacts to certain conditions.

A baseline audit will help you get the most value from syslog data. In particular, a baseline audit helps identify what the system was like before you became suspicious of its behaviour and will help you determine what has changed. On a day to day basis the system logs can be used to determine network health and how well the network and computer systems are running. This provides the ability to proactively solve issues before the user is aware.

### Intrusion detection

With the right training and tools you can use your system logs as a form of Intrusion Detection. This could be as simple as identifying anomalies from your baseline to performing heuristic analysis. Heuristic analysis involves the correlation of a series or sequence of log messages to identify an attack or a total compromise.

Generally, when an attacker breaks into a system their first activity is to maintain anonymity and control without being noticed. To do this they need to cover their tracks and ensure no footprints are left. Even a default configuration of syslog will show some evidence of intrusion activity, therefore these logs are a prime target for the attacker to cover their activities once a full compromise has been successful.

**Incident containment and forensics**

If a compromise has been identified, a structured investigation needs to be started to determine the extent and severity of the compromise. This is when your Computer Incident Response Team (CIRT) gets involved. It should be noted that syslog data is but one part of the whole picture when investigating a computer incident. IDS logs, firewall logs, router logs, and other sources are correlated together to help the CIRT paint a complete picture of the incident.

The information from all these sources has great value to the forensic examiner in the investigation of computer incidents. If the integrity of these logs can be satisfactorily guaranteed (forensically sound) you can paint an accurate picture of the attacker's activity almost immediately.

During the initial throws of an investigation collected evidence will assist with containment. The goal of containment is to remove the opportunity for the incident to escalate beyond its current impact to the company.

Later this evidence will assist in restoring the systems to its intended role for the company by indicating what needs to be done to eradicate the vulnerability once the system is returned to production.

Evidence will continue to play an integral role in the ongoing investigation, which will eventually be used in the court room during prosecution of an identified attacker.

**Further comments**

A compromise of any magnitude is damaging to your company. The size of an incident or the motives of the attacker does not matter. Whatever an attacker does from just logging into your system to trashing the whole hard drive, creates serious doubts in the trustworthiness of the system. This will cause the company to spend excessive time and money recovering from an intrusion with the likely outcome being a total rebuild in an effort to regain confidence in the system.

Logging won't give you total confidence in your system without a total rebuild. Even if all the logged data is complete and verifiable the possibility remains that some portions of the attacker's actions have not been detected. The logged data will help you investigate what went wrong and what is needed to ensure this does not happen again.

Logging is only one aspect of a secure network but provides a great deal of value. It should be part of any defence in depth approach, correlated with information from other sources it becomes a very important tool.

A defence in depth approach to system security goes hand in hand with a defence in depth approach to logging. Each system has a different perspective of the network, not only because of its position within the network but by how it sees the information it monitors. IDS, Firewall, routers and hosts all provide varying degrees of logging; all these combine to provide a complete picture of network activity. We are taking the strengths of each component to build a robust, cohesive, and comprehensive system.

Given the importance of logging, it should not be a last minute consideration, it needs to be built into the network design from the start. The importance of logging sits with network filtering design, IDS placement, server placement, system management, security policy etc.

Thinking of syslog in this light will ensure you have a reliable network and system accounting environment that should be both manageable and expandable well into the future.

## Introduction to Syslog

Eric Allman[4] is the original inventor and author of syslog with the first implementation on the BSD (Burkley Software Distribution) operating system; it was designed as the logging system for the sendmail program that was also originally developed by Eric Allman.

Due to its flexible nature the BSD implementation has enjoyed sustained use over many years, it has been the source for an endless list of implementations on other operating systems and devices. According to the syslog(3) man page on a Red Hat 8.0 system under history[5]:

```
A syslog function call appeared in BSD 4.2.  BSD 4.3
documents   openlog(),   syslog()   closelog(),   and
setlogmask().  4.3BSD-reno also documents vsyslog()
```

Until recently there has never been a formal specification for the syslog protocol, due to this lack of standard and the inherent insecurities associated with the protocol there have been many implementations, they have tried to maintain backward compatibility while extending the protocol to provide more security and flexibility.

In August 2001 an RFC (Request for Comments) document was released by the IETF (Internet Engineering Task Force) titled RFC 3164 "The BSD Syslog Protocol", this was the first step towards providing a consistent and standard logging environment.

The Abstract of this document states that it "describes the observed behaviour of the syslog protocol"; its intention is too set a base that improvements can be built upon, or a reference paper that future implementations can refer to when adding enhancements.

The standards track for syslog is well on the way with such improvements including RFC 3195 "Reliable Delivery for syslog" and an internet draft document draft-ietf-syslog-sign-11 "Syslog-Sign Protocol"[6]. These documents expand on the initial RFC 3164 document providing some of the security enhancements that have been needed from its inception.

[4]look @ http://www.sendmail.org/~eric/ for his home page
[5]To access this man page if available type "`man 3 syslog`" at a shell prompt
[6]The Syslog working group's charter is located at http://www.ietf.org/html.charters/syslog-charter.html. These documents can be found at http://www.employees.org/~lonvick/index.shtml while you can locate all RFC's at http://www.ietf.org/rfc.html

## Before You Start Logging

Log data is so important in detecting suspicious or unusual behaviour, but during this process you are also capturing information about legitimate users and their behaviour. If you log every hiccup, burp or squeak your system generates you maybe violating someone's privacy.

A security policy should be produced by every company! A security policy will identify how your company collects, manages, and uses information, how and what assets need to be protected, by whom and by what methods the company uses collected data. Your company's attitude to the level and use of logging needs to be clearly articulated and defined, it should identify among other things the level of logging used, who monitors those logs, and how these logs are used.

Users need to understand what level of privacy they should expect while working on your network environment. The security policy should be presented to all employees the first day they start work and use company IT resources. New users need to understand and agree to the level of privacy provided by the company.

Over time the first day of work becomes a blur and the security policy falls to the back of the mind. Using logon banners is a great way to remind users of your logging policy while also having the added benefit of warning potential attackers of your security policy as well. A warning banner will go a long way to legitimising the companies position if the need arises to prosecute anyone identified doing what they should not be doing. An example warning banner may look like:[7]

```
This system is for the use of authorised users only.
Individuals using this computer system without authority,
or in excess of their authority, are subject to having
all of their activities on this system monitored and
recorded by system personnel.

In the course of monitoring individuals improperly using
this system, or in the course of system maintenance, the
activities of authorised users may also be monitored

Anyone using this system expressly consents to such
monitoring and is advised that if such monitoring reveals
possible evidence of criminal activity, system personnel
may provide the evidence of such monitoring to law
enforcement officials.
```

Not only do users benefit from a security policy but administrators as well, the policy will discuss issues related to the network administrators' work, they can use the policy to understand what is involved in their job, what responsibilities they have, the required level, use, collection, and maintenance of system logs. During an incident administrators will be able to use the policy to identify how and by who these logs should be handled in an investigation.

If your company plans to prosecute someone for unauthorised activity, how the evidence is collected, stored and analysed will come into scrutiny by the defence. System logs will be very important in your investigation so strong security, integrity and authentication of these logs will need to be part of your syslog design.

[7] US DoJ Warning Message taken from the SANS institute Track 1 – SANS Security Essential + CISSP CBK lecture notes day 6 section 3 page 24 (v1.0 – Hal Pomeranz – April 2002)

An indication of these requirements is highlighted in RFC 3227[8] titled "Guidelines for Evidence Collection and Archiving", here is an extract from RFC 3227 (section 2.4 – Legal Considerations)

```
Computer evidence needs to be

- Admissible: It must conform to certain legal
  rules before it can be put before a court.

- Authentic: It must be possible to positively tie
  evidentiary material to the incident.

- Complete: It must tell the whole story and not
  just a particular perspective.

- Reliable: There must be nothing about how the
  evidence was collected and subsequently handled
  that cast doubt about its authenticity and
  veracity.

- Believable: It must be readily believable and
  understandable by a court.
```

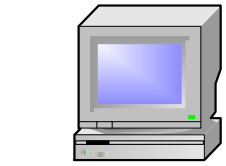
Syslog plays a part in providing for these requirements when configured to do so. A default installation of syslog will not allow you to fulfil these requirements, the original syslog protocol is just too insecure to be relied upon in a court of law. We will understand why this is the case later in the paper.

Because of these issues there is a push to provide a more secure syslog protocol to improve the storage and handling of logged information. The issues that address these shortcomings like authenticity and reliability will be identified throughout this paper.

The systems that you use to cover the requirements stated within RFC 3227 need to be documented within the security policy. With well educated IT staff and end users, the process of security will be easier and provide for a less hostile and chaotic environment when things do go wrong.

[8] RFC3227 – Guidelines for Evidence Collection and Archiving, can be locate through http://www.rfc-editor.org

## Local Logging




To begin our look at logging we need to start with the basics. No matter how complex you make your logging architecture it is always made up of basic building blocks, so we start at the first building block, local logging. In a local logging configuration the host performs both roles of device and collector, messages generated by the system are directed to the syslog(d) process which 'routes' these messages to a local destination on the host machine based on the rules set in a file `/etc/syslog.conf`.

Red Hat 8.0 installs syslog with local logging already configured; this requires no user intervention and provides what would be considered a minimum level of logging.

Due to the lack of standards subtle differences exist between syslog(d) implementations on varying operating systems, the fundamentals are essentially the same but some features may be added to improve or simplify the implementation. It's recommended to refer to your particular vendor documentation to look for these differences.

### Red Hat 8.0 Local Logging Configuration

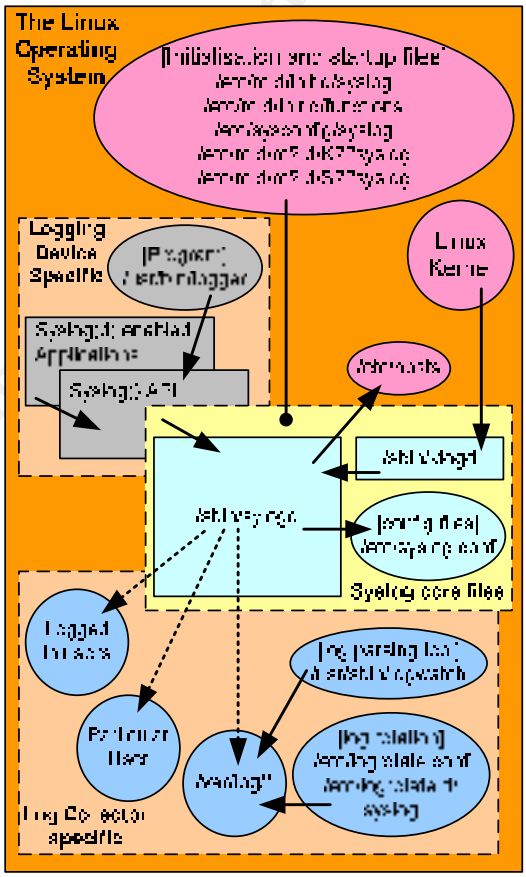



**Figure 1: This diagram shows how Red Hat 8.0 systems and files interact with the syslog daemon in a standalone local logging scenario**

This diagram shows the user controllable files and software that can and do affect a local syslog(d) configuration on Red Hat 8.0, each part is discussed in detail. The GNU/Linux implementation is compatible with the original BSD design, it does however provide a couple of extensions. These extensions include a separate syslog daemon dedicated to kernel logging (`/sbin/klogd`) and some syntax additions for use within `/etc/syslog.conf` that give the administrator more control over how messages are handled.

***The Daemon***

The daemon is the core of the syslog system. It effectively routes incoming messages directing then from varied sources to many potential destinations. On a Red Hat 8.0 system there are two daemons:

```
/sbin/syslogd
/sbin/klogd
```

`/sbin/syslogd` is the primary daemon that runs on all syslog implementations this is the workhorse of the whole syslog implementation and routes messages to their pre configured destination.

The `/sbin/klogd` daemon on the other hand is specific to GNU/Linux and is dedicated to kernel logging; developers decided kernel logging required a daemon designed to cater to the requirements of a Linux kernel.

Syslog(d) can still handle kernel messages if configured to do so but by default kernel messages are handled by `/sbin/klogd` which sends messages it receives from the kernel to `/sbin/syslogd` which then redirects them as configured.

A file created after the daemons are started is `/var/lock/subsys/syslog` that indicate the daemons are running. In addition to this, a file is created for each running daemon that holds its process ID:

```
/var/run/syslogd.pid
/var/run/klogd.pid
```

Having a file that contains the process ID is handy for programmers and makes the writing of scripts to handle running daemons easier.

**Managing the daemons**

These `*.pid` files are used by the GNU/Linux start-up and shutdown scripts executed by the host operating system. The scripts listed here control the syslog daemons, and are responsible for bringing the daemons into a running state:

```
/etc/rc.d/init.d/syslog
/etc/rc.d/init.d/functions
/etc/rc.d/rc0.d/K88syslog     [linked to /etc/rc.d/init.d/syslog]
/etc/rc.d/rc1.d/K88syslog     [linked to /etc/rc.d/init.d/syslog]
/etc/rc.d/rc2.d/S12syslog     [linked to /etc/rc.d/init.d/syslog]
/etc/rc.d/rc3.d/S12syslog     [linked to /etc/rc.d/init.d/syslog]
/etc/rc.d/rc4.d/S12syslog     [linked to /etc/rc.d/init.d/syslog]
/etc/rc.d/rc5.d/S12syslog     [linked to /etc/rc.d/init.d/syslog]
/etc/rc.d/rc6.d/K88syslog     [linked to /etc/rc.d/init.d/syslog]
/var/lock/subsys/syslog
/etc/sysconfig/syslog
```

After basic initialisation of a GNU/Linux system is complete the systems **init** process checks the `/etc/inittab` file to know what run level the system is to be started under. If the system is configured to perform without a GUI (Graphical User Interface) the scripts for run level 3 will be executed, these are located under the `/etc/rc.d/rc3.d/` directory and will bring up all the services for that run level.

The script `/etc/rc.d/rc3.d/S12syslog` will start both the syslogd and klogd daemons, this file is a symbolic link[9] to `/etc/rc.d/init.d/syslog`. This script can be executed directly by an adequately privileged user, providing the correct parameters you can start, stop (kill), and restart syslogd and klogd.

The `/etc/rc.d/init.d/functions` file provides reusable functions for scripts within the `/etc/rc.d/init.d/` directory including the `/etc/rc.d/init.d/syslog` script, this is used to streamline code and ensure programmers don't have to "reinvent the wheel" so to speak every time they need to write a new start-up script.

The syslog(d) daemons can be started with various options which are set within the `/etc/sysconfig/syslog` file. This is read by the `/etc/rc.d/init.d/syslog` script when initialising the daemons. On a Red Hat 8.0 system the default options are:

```
SYSLOGD_OPTIONS="-m 0"
KLOGD_OPTIONS="-x"
```

This will place syslog(d) in a mode where regular (mark) messages are **not** sent to the syslog message destination, mark messages are time stamps sent at regular intervals. Klogd is started with options indicating it should not process any "oops" messages.

**Looking for the daemons**

There are a couple of places to see if syslog(d) is running on your system, one way is to execute the command `netstat –na | more`; this should show a line like:

```
unix  3      [ ]         DGRAM                    73135  /dev/log
```

`/dev/log` is a UNIX domain socket from where local syslog(d) messages are read.

Another way to see if syslog(d) is running is by executing the command `ps ax | grep syslogd` looking for output like:

```
3445 ?        S      0:00 syslogd -m 0
```

Also running `ps ax | grep klogd` will show output like:

```
576 ?        S      0:00 klogd -x
```

As you can see the switches sent to the daemon are also viewable.

[9] A symbolic link is a special file that points to another file on the system

### *Daemon Configuration*

As the daemon starts up it reads a configuration file `/etc/syslog.conf` to determine the actions it should perform with messages.

Although this file is read during initialisation, syslog(d) can be forced to re-read the file if a SIGHUP signal is received. This file is configurable by the system administrator and tells syslog(d) what to do with all the messages it receives, which include, but are not limited to:

- Sending messages to all logged in users
- Sending messages to a particular user (hopefully they are logged in).
- Sending messages to a log passing program
- Sending messages straight to a file (usually located within `/var/log/`)
- Sending messages to another server over the network (we will see more on this when we talk about central log servers)

### *Documentation*

The best place to learn more about syslog(d) and the configuration files is to look at the documentation that comes with your Red Hat 8.0 system:

```
/usr/share/doc/sysklogd-1.4.1
/usr/share/man/man2/syslog.2.gz
/usr/share/man/man3/syslog.3.gz
/usr/share/man/man3/vsyslog.3.gz
/usr/share/man/man5/syslog.conf.5.gz
/usr/share/man/man8/klogd.8.gz
/usr/share/man/man8/sysklogd.8.gz
/usr/share/man/man8/syslogd.8.gz
```

To access this documentation use the command `/usr/bin/man`, type `man man` at the command line for information on how to use `man`

### *Log Rotation*

As will be discussed later in this paper log files can become rather large, even to the point of being unmanageable. To help manage these files a process of log rotation is used. These files set the parameters of log rotation, frequency of rotation, weather to use compression on the saved logs and when to purge old logs from the system.

```
/usr/sbin/logrotate
/etc/logrotate.d/syslog
/etc/logrotate.conf
/etc/cron.daily/logrotate
```

### *Log passing/reduction*

Another subject that will only be touched on here is log passing tools, on a Red Hat 8.0 system a program named "logwatch" is used to analyse the system logs and report via email to the root user what it finds. Files related to this process are:

```
/usr/sbin/logwatch
/etc/log.d/logwatch.conf
/etc/log.d/conf/services/syslogd.conf
/etc/log.d/scripts/services/syslogd
/etc/cron.daily/00-logwatch
```

### *Other files*

Other files related to the syslog system are

```
/usr/include/sys/syslog.h
/usr/include/syslog.h
```

These are called header files and provide definitions for applications using syslog to log their messages.

Not only applications can log messages, scripts or even users may want to log messages, you may want to test that you daemon is working, this can all be done with:

```
/usr/bin/logger[10]
```

Any user can send a message to your syslog(d) daemon, we will use this tool more but as an example of its ability you can type this at the command line:

```
/usr/bin/logger -p local0.notice -t APP "this is a user message"
```

You will understand the parts that make up this command as we read further, on a default Red Hat 8.0 system this message will be found in `/var/log/messages`. You need root privileges to view this file but you will see an entry towards the end like this:

```
Mar 30 09:49:25 localhost APP: this is a user message
```

With adequate rights you should see the file by typing `tail /var/log/messages` at the command line.

In this section we covered the core files that make up a working syslog implementation on Red Hat 8.0. Next we will learn more about how to configure local logging for our specific requirements.

[10] Type "`man 1 logger`" at command prompt for more details

## Source and Level of Logging

### *The Selector*

Syslog(d) compatible applications are designed to produce a message when they reach a particular event during operation, when these events are reached the system utilises the syslog API[11] to deliver a message to the host operating system.

Syslog(d) can receive messages from many sources and choose to log or not to log depending on the level set by an administrator within the `/etc/syslog.conf` file.

Each line that is not a comment[12] contains two basic fields separated only by tabs, it has been known to cause problems on some implementations if anything but a tab is used to sperate these fields, the basic structure is:

```
[      selector     ]        <action>
```

The action part will be discussed further in the next section but it should be sufficient to say this tells syslog(d) where to direct the messages defined by the selector.

The selector contains two parts, facility and priority, these are separated by a period ( . ) so a correctly formatted line within `/etc/syslog.conf` will look like:

```
<facility>.<priority>        <action>
```

An application will generate messages that will include, at a minimum, where the system message came from (the facility), the level of importance the message dictates (the priority), and any relevant information regarding the event that elicited the condition (the message itself).

These parts will be seen in greater detail when we look into the structure of a syslog message during our discussion of Central log servers.

### *Facility*

The facility helps syslog(d) differentiate the source of the message and who or what generated the message. There are 24 possible facilities and these identify the system that generated the message. These facilities are not application specific but are system specific. For example, the mail facility (2) would receive messages from mail related applications.

On a mail server we could configure a mail daemon (eg Exim) with anti spam (Spam Assassin) and anti virus (AmaVis) applications, these will log mail related messages to the mail facility while also logging information about start-up and shutdown events to another facility like `local7`.

Looking at parts of a default Red Hat 8.0 `/etc/syslog.conf` file[13] [14] we can further illustrate how the facility works.

```
authpriv.*                      /var/log/secure
mail.*                          /var/log/maillog
```

[11] To understand more about the syslog API review the man page by typing "`man 3 syslog`"
[12] Blank lines or lines that start with a '#' are considered comments
[13] For a more detailed explanation of this syntax please consult the `syslog.conf` man page
[14] For a full copy of a Red Hat 8.0 `syslog.conf` file please look at appendix B

The facility is the first part of each line preceding the separator character (.), the priority is indicated by a '*' symbol, a * in this position simply means all priorities.

The first line directs syslog(d) upon receiving a message with facility `authpriv` (10) indicating an event related to security and authorisation of any priority level send it to the file `/var/log/secure`. The second line directs syslog(d) to send messages with any priority and a facility of `mail` (2) indicating a message from a mail daemon, send it to the file `/var/log/maillog`.

### *All 24 facility values*

| ***Numerical Code*** | ***Facility name***[15] | ***RFC3164 Facility***[16] | ***Red Hat 8.0 Facility***[17] |
|---|---|---|---|
| 0 | kern | kernel messages | kernel messages |
| 1 | user | user-level messages | random user-level messages |
| 2 | mail | mail messages | mail system |
| 3 | daemon | system daemons | system daemons |
| 4 | auth | security/authorisation messages (note 1) | security/authorisation messages |
| 5 | syslog | messages generated internally by syslogd | messages generated internally by syslogd |
| 6 | lpr | line printer subsystem | line printer subsystem |
| 7 | news | network news subsystem | network news subsystem |
| 8 | uucp | UUCP subsystem | UUCP subsystem |
| 9 | cron | clock daemon (note 2) | clock daemon |
| 10 | authpriv | security/authorisation messages (note 1) | security/authorisation messages (private) |
| 11 | ftp | FTP daemon | FTP daemon |
| 12 | - | NTP daemon | reserved for system use |
| 13 | - | log audit (note 1) | reserved for system use |
| 14 | - | log alert (note 1) | reserved for system use |
| 15 | - | clock daemon (note 2) | reserved for system use |
| 16 | local0 | local use 0 (local0) | reserved for local use |
| 17 | local1 | local use 1 (local1) | reserved for local use |
| 18 | local2 | local use 2 (local2) | reserved for local use |
| 19 | local3 | local use 3 (local3) | reserved for local use |
| 20 | local4 | local use 4 (local4) | reserved for local use |
| 21 | local5 | local use 5 (local5) | reserved for local use |
| 22 | local6 | local use 6 (local6) | reserved for local use |
| 23 | local7 | local use 7 (local7) | reserved for local use |

*Note 1 - Various operating systems have been found to utilise Facilities 4, 10, 13 and 14 for security/authorisation, audit, and alert messages which seem to be similar.*

*Note 2 - Various operating systems have been found to utilise both Facilities 9 and 15 for clock (cron/at) messages.*

If you were only interested in keeping track of events from particular systems you can direct that facility anywhere you choose just by adding a line within the `/etc/syslog.conf` file e.g.

```
kern.*                                  /var/log/kernel
user.*                                  /var/log/user
mail.*                                  /var/log/mail
daemon.*                                /var/log/daemon
auth.*                                  /var/log/security
- . -                                      -
```

If you were not concerned with segregating messages from a particular facility you can direct messages from all facilities with a '*' symbol.

[15] As defined in `/usr/include/sys/syslog.h` file on a Red Hat 8.0 system
[16] As defined in RFC3164 "The BSD Syslog Protocol"
[17] As defined in `/usr/include/sys/syslog.h` file on a Red Hat 8.0 system

There are many very interesting and flexible ways of configuring a Red Hat `syslog.conf` file to achieve your aims, it is recommended the reader refer to the `syslog.conf` man page[18].

## *Priority*

Syslog(d) messages indicate a "level of importance" or priority and gives the administrator further flexibility when filtering messages.

There are 8 priority levels from debug to emergency.

**All 8 priority values**

| *Numerical Code* | *Severity name*[19] | *RFC Severity*[20] |
|---|---|---|
| 0 | emerg | Emergency: system is unusable |
| 1 | alert | Alert: action must be taken immediately |
| 2 | crit | Critical: critical condition |
| 3 | err | Error: error condition |
| 4 | warn | Warning: warning condition |
| 5 | notice | Notice: normal but significant condition |
| 6 | info | Informational: informational messages |
| 7 | debug | Debug: debug-level messages |

Looking again at a real world example taken from a default Red Hat 8.0 `syslog.conf` file:

```
*.emerg                                 *
uucp,news.crit                          /var/log/spooler
```

A standard entry tells syslog(d) that all messages from the indicated facility with an identified priority level and above require the specified action.

In the above example the first line directs syslog(d) to send messages from any facility (`*`) with priority `emerg` (0) to all logged in users (This is shown by a '`*`' symbol in the action field). Receiving a message of this priority indicates that the system is in an unusable state.

The second line specifies messages from two different facility levels `uucp` (8) and `news` (7) with a priority `crit` (2) or above should be sent to the file `/var/log/spooler`. In other words messages with a priority of `critical` (2), `alert` (1) and `emergency` (0) will be directed to the file `/var/log/spooler`.

Red Hat 8.0 offers some extra flexibility when specifying which priority levels to perform actions on. It is recommended the reader refer to the `syslog.conf` man page.

You can direct any priority anywhere you choose just by adding a line within the syslog.conf file e.g.

```
*.emerg                                 /var/log/emergency
*.alert                                 /var/log/alert
*.crit                                  /var/log/critical
*.err                                   /var/log/error
- . -                                       -
```

[18] Type "`man syslog.conf`" at a command line to view the `syslog.conf` man page.
[19] As defined in `/usr/include/sys/syslog.h` file on a Red Hat 8.0 system
[20] As defined in RFC3164 "The BSD Syslog Protocol"

**Comments**

How much should be logged depends on what information you need from your logs. The more you log the better chance you have at debugging that annoying problem, and the more likely you are to start ignoring those copious entries. If you don't log enough you may miss important information while excessive logging could have an impact on system performance, though today's servers should, under normal circumstances, have adequate capacity.

The security policy should describe the level of logging your company requires to do business and maintain knowledge of network health. If your policy does not mention this or you are identifying what needs to be in your policy, the best way to determine your requirements is through experimentation and look at best practice.

Too much is better than not enough but if you are prone to ignoring logs because they are too copious for your needs, then you may wish to limit your logging to the most appropriate subsystem on your host only if this ensures you will look at them on a regular base.

Personally, I would log the maximum possible and use another tool to scan through these logs and isolate interesting or important entries which will be looked at on a daily basis. If there is something that piques your interest you can use the entire log and investigate further. This will minimise the administrative impact of excessive logging while giving the flexibility to review them when the need arises. The use of log passing scripts/software will be expanded upon later

We have seen where messages come from and the varying levels of priority they can have, we will now see what syslog can do with all these messages.

## Message destination (the action field)

Now we know where messages may come from we need to think about where to send them. The manual page for `syslog.conf`[21] indicates what actions can be taken against messages selected by their facility and priority as they arrive. On Red Hat 8.0 systems we have 6 actions that can be performed:

- Directed to a specific terminal or console;
- A selected list of users;
- Everyone logged on;
- A regular file;
- To a named Pipe; or
- Over the network to a remote Machine.

[21] Type "`man syslog.conf`" at the command line.

**Terminal, Console or Users**

The first three items will direct messages to a console. You can specify a particular terminal or console, a list of users that must be logged in to receive the message (eg your admin account) or everyone logged into the system at the time the message is generated.

This form of logging will provide real time delivery of messages which could be very important for messages with a facility and priority combination that you don't expect to see a lot of messages from. If there are messages of this type you may just require real-time response.

A good example of this would be to log all authentication messages to the console, that is, all messages of priority debug and above on a system that has very little user or administrator interaction (Such systems could be your DNS or mail server within your DMZ/screened subnet). On such a system you should not see many messages. As soon as you do there could be a serious issue with you machine.

If messages are sent to a particular console, this console needs to be in constant view of an administrator. There is no use logging to a console if the machine is stuck in a room infrequently visited. Messages sent to a particular user may never be seen if that user is not logged, or even at the console. A lot of machines within a security gateway will be headless, (no monitor) so logging to a console in this case will not be effective.

If this machine is in an open location where the monitor can be viewed by anybody as they walk by, you could be giving a potential attacker some great reconnaissance right there on the screen.

The main problem with these message types is they could clutter your screen while you are trying to do work. Too many messages and you could loose data if not seen before it scrolls out of the screen buffer, not to mention the frustration involved when these messages constantly appear as you are trying to work.

> ***Personal Experience:*** *When administering a Gauntlet firewall built on a BSDi operating system, a subset of messages was displayed on the main console, this was really a great feature, I did not have to be logged in to see important messages as they scrolled past. While I was in the server room at one point I saw a port scan happen right in front of me. As I observed the screen, the messages scrolled across the screen too fast for me to understand what was really happening so I logged into the terminal to investigate further but was hampered by the continuation of these messages, what I needed to do was log into an alternate console so I could continue the analysis.*

From this story you can see some of the benefits and the downside of console logging. To get the benefit of real time alerts you need to have constant observation. These messages are not a permanent record of events and they will not be useful for forensics as you probably won't be able to view these messages the next day or even an hour later.

**Regular file**

Sending your logs to a regular file seems to be the most used action and for good reason. It provides a permanent record within a text file, these files are easily viewed and parsed through a third party tool or script. Storage and maintenance of text files is easy on a UNIX system as there are so many tools and programming languages that make management of text files very easy.

Sending a message straight to a file removes the benefit of real time alerts. A script could regularly parse the logs looking for interesting events and then alert you via other means, but you will always have a delay.

**Named pipes**

Named pipes allow you to send messages directly to third party programs or scripts for log reduction or special types of alerting etc. This can improve the notification of important messages while also increasing the complexity and configuration by involving additional applications. This maybe the best way to get near real time alerting with a custom program or script.

**Remote machine**

Sending your logs to a remote machine is an ideal solution for central log management, the pro's and con's of logging to a remote machine will be discussed when we talk further about log collectors

## Log Retention and Rotation

Log retention refers to the length of time log data is maintained on the local system before you feel comfortable permanently removing it. This should be part of your security policy and could be a legal requirement. If syslog is configured to log to a file, it will continue to append messages as they arrive with no regard to the size of the file or available space on the system. At a minimum, keeping the logged data for at least a month is advisable, but before you decide to delete the logs they should be backed up to more permanent storage like tape or CD ROM.

So these files don't become unmanageable, syslog files should be rotated at regular intervals; depending on the quantity this could be on a weekly, daily or even an hourly basis. If log files get quite large then a daily rotation should work well, with a weekly log rotation cycle being reserved for the least active machines.

I recommend rotating logs on a daily basis in a server environment. Rotate them into a file name that represents the date the logs were generated. This makes it extremely easy to identify the log you are seeking and keeps a clean chronology of log data. As an example you could quite easily generate log files named:

```
20030323_messages
2003-03-23-messages
2002032315_messages
```

The first two file names would hold all the logs for the 23rd of March 2003, the last example would also hold logs from this day but only from 3:00.00pm to 3:59.59pm.

Log files are just text documents and can be easily compressed. Compressing past logs will save considerable disk space allowing you to maintain a longer history.

Log files consume space. Anything that could increase the rate your log files are filled could be a denial of service or faulty software that could eventually consume all available disk space and kill your server.

UNIX systems have a file system structure that is fundamental to its operation. This structure starts with the / or root partition. If all the directories branching from this root partition are on the same physical disk partition and this fills to capacity, your system will stall and or refuse to function normally.

A solution to this is to create a separate physical partition that you mount as directories with in the root partition. All log files should be placed into a separate physical partition normally labelled `/var`. Not only do you reduce the chance that your log files will cause a denial of service condition but there are further benefits to doing this that will be discussed in the next section.

**Red Hat 8.0**

Red Hat 8.0 has `/usr/sbin/logrotate` to perform log rotation, the man page says:

```
Logrotate is designed to ease administration of
systems that generate large numbers of log files. It
allows automatic rotation, compression, removal, and
mailing of log files.  Each log file may be handled
daily, weekly, monthly, or when it grows too large.
```

This program can be easily configured to provide the features that you need, relevant files are:

```
/usr/sbin/logrotate
/etc/logrotate.d/syslog
/etc/logrotate.conf
/etc/cron.daily/logrotate
```

Rotating logs keeps them both manageable and available.

## Log Parsing and Reduction Tools

When you have performed a baseline audit of your systems you will have an understanding of normal activity. Using a log reduction tool you can hide or remove those events that you deem normal behaviour, thus reducing the size of the logged information and providing a more concise view of the system.

These scripts can do a few things, remove messages that you know are benign, draw your attention to messages that you know are important, and show you any messages that you have not seen before. Don't use these scripts to only display messages you think are important as you may miss something. Scripts should be designed to remove only what you know is ok; there is no way that you will know every possible message that the system will generate. Unusual entries need to be analysed and appropriate action taken.

**Red Hat 8.0**

Red Hat 8.0 uses `/usr/sbin/logwatch` to look at the log files, the man page states:

```
Logwatch is a customisable, pluggable log-monitoring
system.  It will go though your logs for a given
period of time and make a report in the area that
you wish with the detail that you wish.  Easy to use
- works right out of the package on almost all
systems.
```

This program can be easily configured to provide the features you need, relevant files are:

```
/usr/sbin/logwatch
/etc/log.d/logwatch.conf
/etc/log.d/conf/services/syslogd.conf
/etc/log.d/scripts/services/syslogd
/etc/cron.daily/00-logwatch
```

Over the last few sections we have looked how we can manipulate log files, we need to now look at how to secure them, how to assure integrity.

## Securing the Stored Syslog Files

The reason for securing your logs is to ensure log integrity and accuracy. If you need this level of confidence in your logs they need to be secured from attack, either intentional or accidental. Here we look at some of the methods that can be used.

### *General Techniques*

#### Printers

One of the best ways to maintain the security and integrity of your logs is to send them all directly to a printer as they are generated. One major draw back to this approach is the quantity of paper consumed; you will need to keep a steady flow. Having your logs on paper removes your ability to parse these logs through a software filter. This will have to be done manually, a tedious and error prone process. Very important logs in a mission critical system may require this kind of treatment.

#### WORM device

An alternative to the printer is to send the logs to write once media also know as a WORM device or a CDROM drive or another device that is infeasible to destroy electronically from over the network. This method provides the benefit of having an electronic copy of the logs but has the downside of always needing to be fed with new media.

#### Serial Line

Having a separate machine attached by a serial line over which messages will pass provides a great alternative for a log repository. You can have a large storage capacity and access to a machine that can parse your logs. This protects the stored log files as they are isolated electronically from an attacker.

#### /usr/bin/chattr[22]

Using the `/usr/bin/chattr` command you can set log files to be append only. The command `chattr +a <filename>` will set the file so it can only be opened in append mode for writing. This poses a problem for your log rotation script and so must be reversed just before rotation and then reset when completed.

Another feature of the `chattr` command is its ability to set a file immutable with `chattr +i <filename>`. A file with this attribute cannot be modified, that is it can't be deleted, renamed, opened for writing, or even linked against. This type of attribute is only applicable for log files that are not currently being used by syslog to write log data, you can set this on your log history.

An attacker needs root level access to change these attributes. If they have this level of access you have a lot more to worry about.

#### Separate partition

Another benefit of separating your file system into partitions is that you can control how each file system is mounted, the `/var` directory can be mounted with append only attributes. The mount point can be remounted without this setting but this can only be done by a root privileged user and we hope that you will notice this.

[22] Type "`man chattr`" at the command prompt for more details of this command

**Comments**

You should use at least one of these techniques, if not more, to protect your log data. The main goal is to make it so difficult for the attacker that they have to jump through many hoops to achieve their objective. It is hoped that the activity will be observed before they manage to hide their tracks. Once an attacker has root status a lot of these systems will not be protection enough to stop root from manipulating log data. This is where we need to try and maintain the integrity of the log files

## *Hashing*

When we take a static file, we can use a hash algorithm to create a number (hash value), this hash value represents the file being hashed. This is an irreversible, one-way transformation. The size of the hash file is fixed and is nothing more that a representation of the static file. If the file never changes, then each and every time the same hash algorithm is used against the file the same data will be generated. Any change, even a single bit, will change the hash value.

File integrity software like Tripwire[23] and AIDE[24] work on this premise. If a file is changed it would change the hash value and the system administrator would be alerted to this serious problem. This is another method used to determine changes from your baseline. Tripwire should be run during the initial build of any system, it will create and store hash values of selected files and as you rerun a check on the host during its life you use these stored hash values to determine what, if anything, has changed.

A log file, on the other hand, is not a static file. Messages are continually being generated and therefore changing the files attributes. This type of activity is not conducive to a normal form of file hashing. For syslog, hash techniques can really only protect the configuration files and binaries, and after the log file has been rotated, a hash can be generated of the now static log file.

Actually, we can hash syslog data, it just takes a fairly complicated configuration to get it to work right. We will look at this issue a little more when we discuss authenticating syslog traffic.

## *Host security*

Physical security of your host is of great importance to any networked device, once physical access is obtained by an attacker there is not much that an attacker cannot do. Very few devices are physically secure once in the hands of your adversary. Cisco routers and any common operating system are all vulnerable to very easy physical attacks.

On the flip side, there is no point in having your systems in a vault if it is not protected from remote access. There are many treatments on the methods required to harden a host from remote exploitation that will reduce the chance an attacker will get onto your host and have the opportunity to observe, modify or destroy your logs. Hardening a system at both the physical and electronic level is required for effective security.

[23] For the open source version of tripwire http://sourceforge.net/projects/tripwire and http://tripwire.org
[24] AIDE or Advanced Intrusion Detection Environment can be found http://www.cs.tut.fi/~rammer/aide.html

**System hardening resources**

OpenNA book: “Securing & Optimizing Linux: The Hacking Solution (v3.0)” can be located at http://www.openna.com while a free earlier release can be found at:

http://www.openna.com/products/books/sol/solus.php, from the website:

> Securing & Optimizing Linux: The Ultimate Solution (v2.0) has been written and achieved with tightening security to an incomparable level in mind. One of its main features is the easy path from beginning to end in a smooth manner, step by step for beginners as well as for experts.

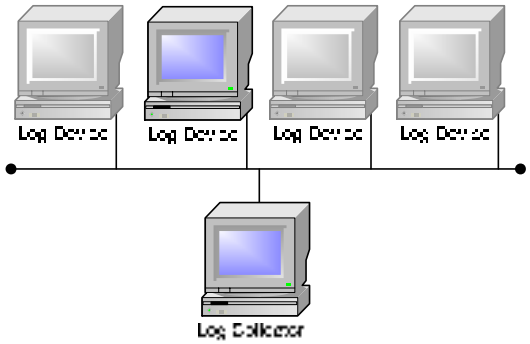
The SANS store http://store.sans.org/ offers papers on securing Linux like “Securing Linux: A Survival Guide for Linux Security”

Red Hat hardening script named Bastille Linux can be found at:

http://www.bastille-linux.org/, from the website

> The Bastille Hardening System attempts to "harden" or "tighten" Unix operating systems. It currently supports the Red Hat, Debian, Mandrake, SuSE and TurboLinux Linux distributions along with HP-UX and Mac OS X. We attempt to provide the most secure, yet usable, system possible.

## Central Syslog Server



The basic idea of central log server design is to consolidate and centralise logs from many devices in an attempt to make administration easier. In any distributed system, if there is task that needs to be done on multiple systems and at regular intervals, it is a real benefit if you can consolidate that task and do it from a single location.

Directing all your syslog data to a central server provides a single location for log management. All the tools we have talked about for managing your syslog data can be applied at one location, providing more convenience in backup, management, manipulation, and monitoring of your whole enterprise in one place.

With all this flexibility there have to be some negatives and one of the biggest negatives is that the standard syslog protocol uses UDP as its transport protocol. UDP has characteristics that are not conducive to secure and reliable system logging. In short, UDP is a connection less, non reliable protocol that does not guarantee message delivery. If a message gets lost, neither the log device nor the collector is going to know or care.

There is no connection between the two communicating hosts as UDP is a drop (onto the network) and forget protocol. If there is a network fault between communicating hosts (this could be caused by a physical break, miss configuration or DoS attack) communications would be disrupted and your logging ability will be lessened as messages will go missing. This places the administrator in a bad position as all the information required to diagnose the problem may not be available or there may be a missing entree that would reveal what is truly going on.

Messages can be injected onto the network directed towards the log collector, and due to the lack of authentication, these will be received and processed according to the rules within the collector's version of `/etc/syslog.conf` without once checking the authenticity of the message. Using a flood of specially crafted packets directed towards the collector, an attacker could cause a DoS (Denial of Service) of your syslog daemon or bring about misinformation to both the Network Administrator and the Intrusion Investigator.

To improve the level of reliability for syslog(d), required features need to be built above the UDP protocol, possibly within the application. An alternative is a total replacement of the UDP protocol with another more reliable one like TCP. TCP is a connection-oriented protocol that provides reliable communication between two hosts by establishing a connection before sending any data. Running syslog messages over TCP will make it infeasible to inject messages though it can't prevent aggressive forms of DoS. There are many ways to help reduce the impact of a DoS but this is beyond the scope of this paper.

### Red Hat 8.0

On Red Hat 8.0 syslog(d), by default, is configured for local logging only and will not accept or process any messages from outside sources. Some additional configuration is required by the administrator on both the device and the collector before we can utilise central logging.

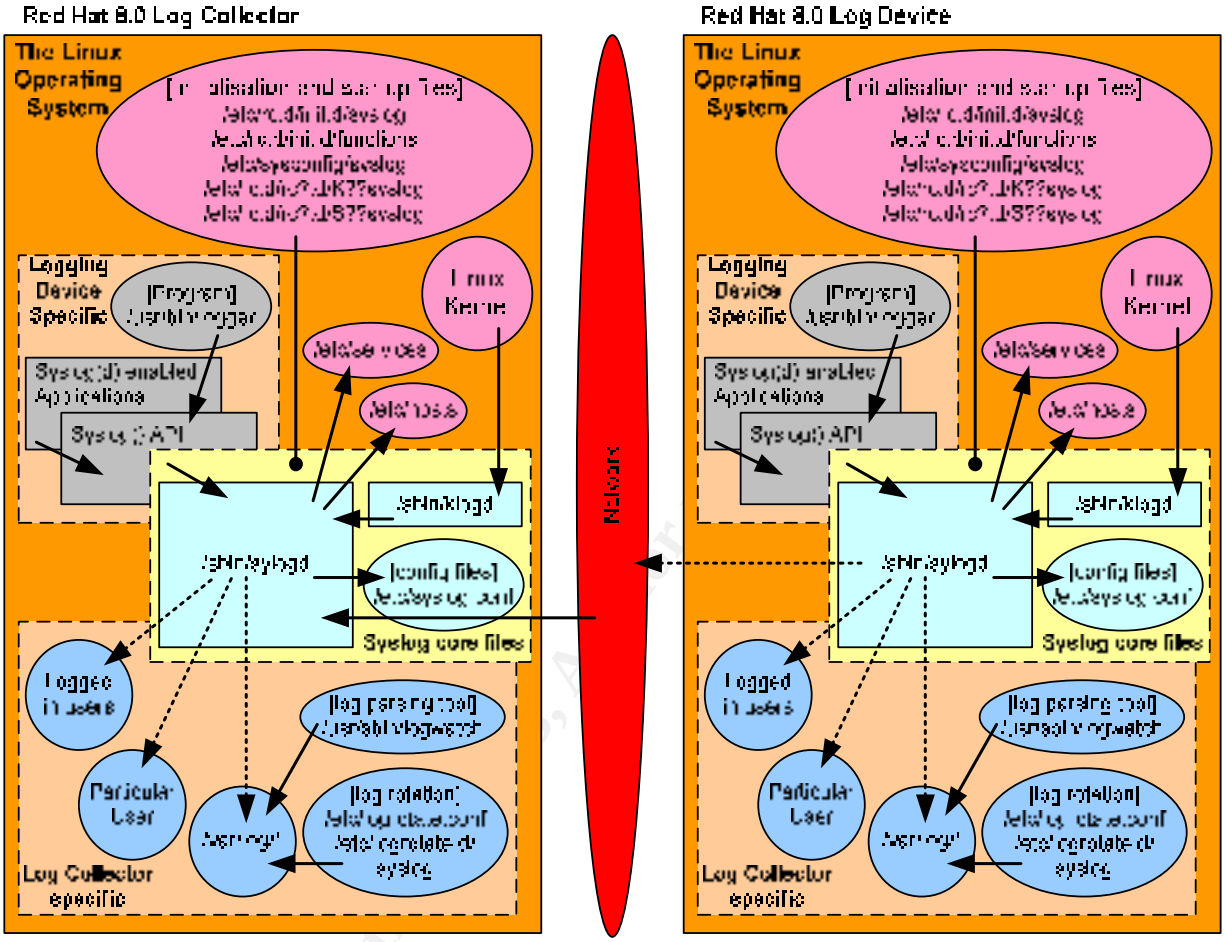



**Figure 2: This diagram shows how Red Hat 8.0 systems and files interact with the syslog daemon in a log device and collector scenario**

This diagram represents the interlinking of user controllable files, software and the network that can and do affect a central logging configuration on Red Hat 8.0. In this diagram you can see the interaction of the log device and the log collector. Remember, a log collector will also be a log device as it generates messages internally.

#### *Daemon configuration*

We need to make sure there is an entry in the `/etc/services` file that indicates the service port number that syslog should use for communication. UDP port 514 has been assigned to syslog and this entry needs to be on both the collector and the device:

```
syslog              514/udp
```

The collector needs to be specifically configured to accept log messages from an external source. For this to work the syslog daemon needs to be restarted with an additional `-r` switch. To do this edit the `/etc/sysconfig/syslog` file with a line like:

```
SYSLOGD_OPTIONS="-r -m 0"
```

When the service restarts it looks for an entry in `/etc/services`. If not found the daemon will fail to initialise. RFC 3164 states:

> It is recommended that the source port also be 514 to indicate that the message is from the syslog process of the sender.

Once an entry on the device is added to the `/etc/syslog.conf` file that directs syslog(d) to send messages to a collector, a UDP port is opened with the port number defined in `/etc/services`. Though this port appears to be open, a port scan indicates it is open, and sending syslog packets fills up the receive buffer, the messages never seem to be processed.

To ensure the port is active run the command `netstat -na --udp` on each machine, if syslog(d) is listening on the required port number `514/udp` you will see a line like:

```
udp        0      0 0.0.0.0:514             0.0.0.0:*
```

This port number can be easily changed as long as it is still a UDP port and not required by another service running on the machine. Doing this will only obscure the location of syslog in a way that will prevent only an unsophisticated, or scripted attack but make your job more difficult, as you will have to maintain this change across your whole enterprise. The daemon will fail to open a listening port if this is changed to a non UDP port.

To see if syslog(d) is running and with what switches, run the command `ps ax | grep syslogd` you should see a line similar to:

```
3445 ?        S      0:00 syslogd -m 0 -r
```

***Name resolution***

Name resolution is used by both the collector and device, there are two main methods for name resolution: centrally with DNS server or locally via a `/etc/hosts` file.

The log device requires host name to IP resolution. When you define a log collector within the `/etc/syslog.conf` file you would usually add the entry with a host name. This host name needs to be resolved to find the collectors IP address. A typical entry on the log device would look like:

```
*.*                                    @collector
```

IP address to name resolution is required by the log collector during the process of writing a message to the log file. Syslog(d) records the source IP address of the message and resolves the IP address to a host name as the message is written to the log. An example of this will be seen when we test this configuration.

For most network administrators, DNS would be considered a good idea because it provides central management of names and IP addresses that can be utilised by all systems. For our logging system to maintain communication, the DNS server needs to be always available. If it is not available due to a DoS attack, break in, or system failure, messages may cease to flow due to the inability to resolve IP addresses.

Though adding administrative overhead, a far more secure method of name resolution is through the use of a `/etc/hosts` file. Add all the required entries to this file and unless the file is deleted or modified names will continue to be resolved correctly.

### *Testing your installation*

To test the new configuration and see a syslog message as it crosses the network we can generate a message on the log device using the program `/usr/bin/logger`[25]. A command may look like:

```
logger -p local0.notice -t DEVICE1 "message local0.notice device1"
```

This tells logger to deliver a message with a facility of `local0` and a priority of `notice`. The `-t` option sets a tag that in this case is `DEVICE1` making it easier to identify the source of the message with the text in quotes being the actual message.

Using `tcpdump` with options like[26]:

```
sudo /usr/sbin/tcpdump -s0 -X -vv -nn -i eth0
```

We can capture a packet showing the collector IP address as `192.168.201.10` and the device IP as `192.168.201.20`:

```
16:36:06.298930 192.168.201.20.514 > 192.168.201.10.514:  [udp sum ok] udp
44 (DF) (ttl 64, id 0, len 72)
0x0000   4500 0048 0000 4000 4011 2735 c0a8 c914        E..H..@.@.'5....
0x0010   c0a8 c90a 0202 0202 0034 6d77 3c31 3333        .........4mw<133
0x0020   3e44 4556 4943 4531 3a20 6d65 7373 6167        >DEVICE1:.messag
0x0030   6520 6c6f 6361 6c30 2e6e 6f74 6963 6520        e.local0.notice.
0x0040   6465 7669 6365 310a                            device1.
```

This packet is of type UDP with the source and destination ports being `514/udp`, you can see the direction of the packet is from device (`192.168.201.20`) to collector (`192.168.201.10`) by the arrow. When we convert the packet into ASCII we can read the message as expected because the syslog packet as defined in RFC 3164 consists of ASCII codes[27].

Once the packet has been processed by the collector and sent to a file, this is what can be seen if the collector is unable to resolve the host name of the log device:

```
Apr  2 16:09:57 192.168.201.20 DEVICE1: message local0.notice device1
```

And this is what should be expected if the collector is able to resolve the host name:

```
Apr  2 16:35:30 device1 DEVICE1: message local0.notice device1
```

[25] Type "`man logger`" to see a full explanation of these and other switches
[26] Refer to Appendix E for a short explanation of the command line and resultant output
[27] USA Standard Code for Information Interchange, USASI X3.4-1968

### *Red Hat 8.0 syslog packet vs. RFC 3164*

From RFC3164:

> It is RECOMMENDED to transmit a syslog message in the format specified in this document, but it is not required.

Red Hat 8.0 syslog packet does not fully conform to the recommended format. To have a syslog daemon that does fully conform you need to replace Red Hat Syslog daemon with another. The one that I have chosen as a replacement is SDSC Secure Syslog from the Security Technologies group at the San Diego Supercomputer Centre. This daemon fully conforms to RFC3164 and has other benefits that will be highlighted later.

As an example of a message as specified within RFC3164 here is the payload generated by SDSC. It contains three major sections, the PRI, the Header and the message and some of these are further sub divided:

```
[PRI][         header      ] [                message                    ]
[PRI][(  timestamp  )(host)] [(tag )(       message                     )]
<133>Apr.28.15:12:30.device1.logger:.message.with.SDSC.syslog.using.UDP.
```

The PRI part contains a number representing the facility and priority of the message, and is bound by angle brackets. More information on how to interpret this field is in Appendix E which has a convenient table to decipher the facility and priority from this number.

The header consists of two fields, a timestamp and a hostname. The timestamp represents the local time of the device while the hostname is what the device knows itself without the domain part (the simple host name, eg `device1`, not `device1.localhost.localdomain`). The hostname is preferred, but could also be the IPv4 or IPv6 address of the machine if the hostname is unknown.

The message part also contains two fields, the tag and the message itself. The tag field contains the name of the program or process that generated the message while the message part is a free form text message that indicates what the problem being reported actually is.

This is an example Red Hat 8.0 payload:

```
<133>logger:.message.with.RedHat8.syslog.using.UDP.
```

If we compare this with the SDSC secure syslog output we can see it does have a PRI part, the header is missing, and the message including the tag is present.

### *Syslog DoS or adding a mysterious entry*

The syslog daemon on the collector does not provide any authentication for messages as they arrive and there is no control from whom it will accept messages. This scenario is ripe for abuse from a DoS or even the more nefarious activity of adding random messages to confuse the administrator. What makes this activity even easier is the fact that RFC3164 states:

> The payload of any IP packet that has a UDP destination port of 514 MUST be treated as a syslog message.

The implications of this can be demonstrated quite easily by using a tool named `hping2`[28]. The first step in this process is to understand how an IP packet holding a UDP payload is constructed[29] then understand what a correctly crafted syslog message looks like to place within the UDP packet[30].

First, we create a plain ASCI text file on a single line containing a correctly crafted syslog message. This will become the UDP payload. I used a text editor on Red Hat named `vi` to create a file `udpbody` with the content:

```
<133>DEVICE1: message local.notice spoofed with hping
```

Now using hping2 we can create a spoofed packet to send to the collector[31]:

```
sudo ./hping2 -I eth0 -c 1 -a 192.168.201.20 -y -2 -s 514 -p 514 \
-d 54 -E ../udpbody 192.168.201.10
```

`Hping` was used on a machine with the IP address `192.168.201.1` and directed to our collector with IP `192.168.201.10` spoofing the source address to be from device1 with IP `192.168.201.20`. Using `tcpdump` we can see the packet as it traverses the network

```
13:44:00.341787 192.168.201.20.514 > 192.168.201.10.514:  [udp sum ok] udp
54 (DF) (ttl 64, id 43674, len 82)
0x0000   4500 0052 aa9a 4000 4011 7c90 c0a8 c914        E..R..@.@.|.....
0x0010   c0a8 c90a 0202 0202 003e 7229 3c31 3333        .........>r)<133
0x0020   3e44 4556 4943 4531 3a20 6d65 7373 6167        >DEVICE1:.messag
0x0030   6520 6c6f 6361 6c2e 6e6f 7469 6365 2073        e.local.notice.s
0x0040   706f 6f66 6564 2077 6974 6820 6870 696e        poofed.with.hpin
0x0050   670a                                           g.
```

As you can see it is very similar to the legitimate packet generated by our Red Hat 8.0 log device seen earlier. When we look at the collectors log files we can see the entry created by this packet

```
Apr  5 13:59:02 device1 DEVICE1: message local.notice spoofed with hping
```

The collector assumes that the packet is legitimate and processes it, even resolving the assumed source host name as being from device1. A separate packet generated with `hping2` with a different but still spoofed source IP address of `192.168.201.11` was still accepted by the collector, processed and written to a log file. The only difference is the fact the collector was unable to resolve the host name:

```
Apr  5 14:04:40 192.168.201.11 DEVICE1: message local.notice spoofed with
hping
```

Using a simply obtained tool we are able to add random messages to the log files on the collector and create a state of confusion for the administrator. Any packet directed at UDP port 514 containing any form of ASCII text is processed and logged as a legitimate message. If a packet is not correctly constructed or does not contain expected information syslog will still process the message using the default facility and priority values of user.notice.

[28] Hping2 can be found at http://www.hping.org/
[29] Best source for this information is "TCP/IP Illustrated, Volume 1 – The protocols" by W. Richard Stevens
[30] This is defined within RFC3164 "The BSD Syslog Protocol"
[31] Look at appendix D to see an explanation of this command

### *Packet filtering*

Because syslog provides no discrimination for packets that arrive, we need to use a third party product to perform this task. Packet filters are designed to control communications between networked devices. This filtering can restrict which device can send messages to the log collector while also restricting with whom the log collector in turn can communicate.

There are two common places to put such filters, firstly on a dedicated firewall separating the device and collector, and/or a host based firewall installed on each device and collector. When we consider defence in depth, we would place filters at all possible locations. This not only protects the network segments (on each leg of the firewall) but also the device within its designated segment (from local attack).

Although this is not a total solution, it will lessen the chance of a DoS attack being directed towards your syslog service. Remember that we are dealing with UDP traffic. If we spoof a legitimate device IP address we are still susceptible to a DoS. This does require the attacker to have a greater knowledge of your network.

### *Host security*

As discussed with local logging, hardening the operating system of any device before deployment is essential. Unlike a log device the log collector should only collect, manage and store syslog messages.

A log collector should be accessible over the network on at most two ports (`22/tcp` and `514/udp`). You should know by now what port `514/udp` is for, but port `22/tcp` is used by secure shell[32] (ssh). With secure shell you can have secure, authenticated and encrypted command line communications with the log collector that is very similar in behaviour to telnet. If you are even more paranoid and you have easy and secure physical access to the collector then only having the syslog port open would be the most ideal.

For system hardening refer to the examples mentioned within the local logging section on host security.

All hosts should be firewalled on each interface with a packet filter even if they are behind a dedicated firewall within a DMZ or screened subnet. Each device needs to fend for itself to ensure that a neighbour, if compromised, does not attack adjacent machines. A firewall will provide general protection to the screened subnet but can not protect you from locally generated traffic. This is part of a defence in depth approach to network design.

All production servers within an environment should be physically secured from everyone except authorised staff. This is not only for protecting the data. Controlled access is required for any investigation when the integrity of the logs is placed into question.

### *Comments*

Although syslog does have some fundamental security problems, it does posses many positive aspects. Centralising your syslog data is an ideal way to mange these messages for any networked environment. Care needs to be taken to limit the known security implications and these issues should be addressed with the design and implementation of the network from the start. Syslog is not the only service that needs a secure, well thought out environment to improve its underlying security.

[32] Look at http://www.openssh.org/ for more information

## Where do I Host my log files?

Now that we have seen what is involved with local logging and what a basic central log collector configuration looks like, we should consider the best approach for log location. Do you log locally, centrally, or is it better to do both?

### *Locally only*

Although local logging is the default configuration for Red Hat 8.0, making it very easy to implement, its usability is limited for any serious multi host analysis. When debugging a problem on a single machine its good to have logs there with the machine you are working on, it can be quite a pain dealing with two machines to diagnose a problem.

The typical attack sequence is well documented. An attacker breaks into a machine and the first thing they do, once they have total control, is sanitise the logs. These logs will have some evidence of the intrusion and it is imperative for the attacker to not only hide his current actions but the actions that they took to break into the machine.

A root user can change anything within these logs and there are many root kits that will remove the whole log or just an entry. This poses a serious problem if this is your sole source of log information. Once the device is compromised you don't have full control of the machine and the integrity of your logs is questionable.

In a gateway environment, being able to backup your systems on a regular basis can be very difficult. The security systems a typical gateway has makes centralised backup very complex. Back up devices on every device is both expensive and difficult to administer. Backing up your data is essential.

If you monitor a small quantity of servers, or there are budgetary and staffing constraints local logging may be the only viable solution.

### *Remote server only*

When correlating messages from multiple hosts, having these logs centralised minimises the need to view logs on each device. If the company policy mandates daily inspection, centralised logging will reduce administration. You can derive greater benefit from your log reduction tools in a central configuration thus assisting the administrator to detect unusual behaviour, network faults, network health etc. This may make it easier to identify a specific malfunctioning host.

If a log device is attacked, log data will not be available to the attacker to sanitize their actions. The log collector should be highly secured and well out of reach for the attacker. This provides a more reliable store of events and, once correlated with other data sources (eg IDS or Firewall logs), is integral to intrusion detection, incident containment and forensic analysis.

Back up is improved with the collector being the only device requiring back up. You will find when specifying requirements for a central log collector you will need to place special emphasis on disk space, hardware redundancy and a suitably specked backup device.

Hardware redundancy is very important for the log collector. RAID, dual NIC's, etc maybe necessary to assure the required uptime, and there may be a need to consider two central log servers. Two log collectors will ensure that there is somewhere to send log messages if for any reason one of them becomes unavailable.

This machine becomes integral for reporting. If the system was to fail for whatever reason there will be no logging what so ever until either this machine is restored, you update the configuration on all the log devices, or you update the name resolution to redirect messages to another collector.

### *Locally and Remotely*

There are very few negatives to this configuration: it is the best of both worlds. If there is a failed network path between the client and the server you have logs available on the local server, if not the full set. An idea is to place a smaller subset of logs to assist in local trouble shooting and hold a central repository with your entire log to get a larger picture of network health.

The complexity of this configuration can be a little daunting. There is extra planning required and each log device requires configuration of log level, rotation and retention. You should find that you can reproduce the configuration across all log devices with a special configuration needed for the collector's thus reducing this complexity.

### *Comments*

No matter where you log, the integrity of each device and the collector is paramount. All major network components need to be well secured physically and hardened at both the OS and application to the highest recommended levels. This may not always be practical as a log device will usually be performing an integral role, eg web, DNS, ftp server, or router.

A central log server holds what could be very valuable information. It has only one role in life and therefore should be easier to secure from a network and physical perspective. A defence in depth approach is essential when designing a network infrastructure. This approach includes, but is not limited to, firewalls, antivirus, intrusion detection systems, filtering at the device and logging.

In my experience, the most vulnerable systems are the ones within a DMZ or screened subnet. In a highly secure environment, servers within these volatile locations have their configurations stored on a central image server. If compromised the machine is forensically imaged for an investigation, and a fresh clean image applied and, if possible modified to prevent the recurrence of the incident. A central log server is integral in this scenario, all vital forensic data is directed away from the DMZ devices.

## Level and destination of Logs Revisited

In a central logging scenario we have a few extra options. There are now two locations where we can store data and we should take advantage of this flexibility. It takes a bit of planning and experimentation to know how much logging should be sent to each destination, but with the benefits of central logging, an easy decision should be to send all you logs to a central log device. The time consuming part in this equation becomes how much log data is require locally, and how much information do you provide an administrator for local problem solving.

The CERT Coordination Centre has put together a lot of great documentation covering all sorts of security related issues. Here is an example logging policy based on a recommendation from CERT[33]

- Forward all messages to a log collector
- Log device store some messages
    - Messages directed to the console
        - All messages with priority “err” or higher
        - All messages with priority “notice” for the more critical facilities “kern” and “mail”
        - All messages from daemons and authentication system
    - Messages directed to a single file
        - All messages with priority “err” or above
        - All messages from facilities “auth”, “daemon”, “mark”, and “kern”
        - All messages from facilities “user” and “mail” with priority “warning” or above
    - Important messages directed to a separate file
        - All messages from facility “lpr” to `lpr.log`
        - All messages from facility “mail” to `mail.log`
        - All messages from facility “daemon” to `daemon.log`
        - All messages from facility “auth” to `auth.log`

This should give you a great start that you can tailor to your specific needs. Refer back to our previous discussion on message destination to understand the benefits of this configuration.

[33] Example based on a paper from the CERT Coordination Centre paper: Configuring and using syslogd to collect logging messages on systems running Solaris 2.x, see References.

## Time Synchronisation

When you have many dispersed assets, time synchronisation becomes all the more critical if you want to get the most from your logs. As soon as you need to correlate information from multiple devices, including Intrusion Detection Systems, Firewalls, routers, etc, it will be much easier when everything is in sync.

If these systems are not in sync you add more complication. You will need to perform rather complex calculations to determine the time difference between systems to identify the true order of events and create an accurate time line.

There are many Internet based time sources that you can synchronise your machines with. Highly secure environments will not use these sources though, as the idea of trusting a time source from an unknown third party to synchronise all your machines, can leave you open to certain attacks. To reduce your risks, use an independent source like a GPS system. These systems can be expensive so for smaller networks using an internet based time source is preferred over no time source at all.

According to RFC3339[34] "true interoperability is best achieved by using Coordinated Universal Time (UTC)". The reason for this is UTC does not take into account daylight saving time, time is not repeated because UTC does not change when an hour is reversed to cater for daylight savings local to the device. An offset is calculated by subtracting the UTC time from the local time, this offset may change due to daylight saving but the base UTC time does not.

Utilising a common time source with a common method of representing your time is very convenient for log file continuity for not only local networks but networks spread across many time zones. As an example, here is a log entry taken from SDSC syslog. We will talk more about this implementation later, but one feature is its ability to use RFC3339 formatted time stamps. This feature though is not consistent with the standard BSD syslog and at this time will work only with SDSC syslog.

```
<133>2003-04-27T21:49:28+10:00.device1.logger:.My.message.with.SDSC.syslog.
```

This represents 49 minutes and 28 seconds after the 21st hour of April 27th, 2003 with an offset of +10:00 from UTC

Another aspect of syslog that helps with log correlation is the mark message. Mark messages should be turned on to make it easier to analyse the time line. When these messages are sent at regular intervals they can help identify possible gaps within log files.

To turn on mark messages you need to restart the syslog daemon changing the switches used during start-up, to do this we edit the `/etc/sysconfig/syslog` file with a line like:

```
SYSLOGD_OPTIONS="-m 10"
```

The -m switch takes a number as an option, this is how many minutes should transpire between each mark message. An example message will look like:

```
Apr 25 16:16:26 device1 -- MARK --
```

Time synchronisation and mark messages help with log analysis particularly when we are dealing with more than one log device.

[34] RFC 3339 – Date and Time on the Internet: Timestamps, locate a copy through http://www.rfc-editor.org

## Changing the transport Protocol and Message Reliability

We have discussed the security issues regarding the UDP protocol. There is no guaranteed delivery or error control in the UDP protocol. Simply put, the receiving host does not have a communications association with the sending host, it does not know that the packet is coming, and does not care if it does not arrive.

The TCP protocol is quite different in its implementation. It can guarantee delivery, it does provide error control, and, via a three way hand shake a communications association is set up before data is delivered[35]. A lot of the improved Syslog implementations use TCP as their transport protocol because of these benefits. Example replacements are Syslog-ng, msyslog, and SDSC Secure Syslog.

To take advantage of these benefits on a Red Hat 8.0 system we need to replace the syslog daemon with one of these implementations. I have chosen to use SDSC Secure Syslog from the Security Technologies group at the San Diego Supercomputer Centre. SDLC Syslog is fully compatible with RFC3164 by using the traditional UDP protocol, it also fully implements RFC3195[36] "Reliable Delivery for syslog". This RFC specifies how to transfer messages over TCP for reliable delivery of messages.

Here is an edited `tcpdump` trace to highlight the benefits of using TCP as a transport protocol again the collector IP address is `192.168.201.10` and the device IP is `192.168.201.20`:

```
01 ->    15:26:15.471298 192.168.201.20.1038 > 192.168.201.10.514: S
         2954639064:2954639064(0)
02 ->    15:26:15.472369 192.168.201.10.514 > 192.168.201.20.1038: S
         2851226966:2851226966(0) ack 2954639065
03 ->    15:26:15.473165 192.168.201.20.1038 > 192.168.201.10.514: . 1:1(0)
         ack 1
Packet 04 to 17 not shown
18 ->    15:26:47.660483 192.168.201.20.1038 > 192.168.201.10.514: P
         654:766(112) ack 393
         ..F....P.............R..ANS.1.3...271.87.0..<133>Apr.28.15:26:45.2
         003.+1000.device1.logger:.message.with.SDSC.syslog.using.BEEP.RAWE
         ND..
19 ->    15:26:47.661366 192.168.201.10.514 > 192.168.201.20.1038: .
         393:393(0) ack 766
         ..9.....S........R......
```

Firstly lets talk about packets 01 to 03. These 3 packets make up the 3 way hand shake. In very simple terms, log `device1` requests to communicate with the `collector` (packet #01). The `collector` responds acknowledging this request while also asking for a return communications path with `device1` (packet #02). `Device1` finishes by acknowledging this reverse communications path (packet #03) finishing the three way handshake which has established a duplex communications path between `device1` and the log `collector`.

[35] To understand these protocols refer to "TCP/IP Illustrated, Volume 1 – The protocols" by W. Richard Stevens
[36] RFC3195 – Reliable Delivery for syslog, locate a copy through http://www.rfc-editor.org

`Device1` and the collector have established a relationship. Any information that travels between these two machines through this path will be acknowledged by the other party, as shown with packets 18 and 19. Packet 18 contains a syslog message from log `device1`. (the message is highlighted in blue) Once the `collector` receives this packet it performs some integrity checks to make sure the packet has not been corrupted during transit and then sends an acknowledgment of recept back to `device1` (packet #19). This forms the basis of reliable delivery; the continual checking and acknowledging of packets as these two machines communicate.

By using TCP as the transport protocol we have some form of acknowledgement that is performed at the transport layer. RFC3195 adds another level of acknowledgment and reliability by utilising the Blocks Extensible Exchange Protocol Core[37] (BEEP). BEEP is a toolkit that can be used for building application protocols.

The main concept behind using the BEEP protocol is to simplify development of network applications, it provides the key ingredients developers need when designing an application protocol. RFC3195 uses BEEP to provide a required protocol framework without needing to design it from scratch.

This section looked at how messages once sent from a log device will be reliably received by the collector. The question then becomes, is this message the same one sent by the log device in the first place? Has this message been changed in any way during transit over the network or while being stored on the log collector.

We provide some level of message integrity by using RFC3195 to provide reliable delivery, and using some of the techniques talked about in “securing the stored syslog files” to protect the log files from tampering. In the next section we look at how to verify all these forms of protection have worked and the once the message has been generated it has not been tampered with.

[37] RFC 3080 “The Blocks Extensible Exchange Protocol Core” explains in detail how BEEP works, locate a copy through http://www.rfc-editor.org

## Integrity and Authentication of messages

Protecting our files from outside attack or manipulation is important. We also need to consider how to ensure the integrity of these messages, is the message saved on the collector the exact same message generated by the log device. The process of authenticating syslog messages give us the ability to ensure that messages have not been tampered with in any way.

The TCP protocol performs certain checks on packets as they arrive, allowing the receiving system to verify the integrity of the message. TCP does this by means of a checksum. This checksum is calculated by the sending host and added to the packet before placing it onto the network. Once the receiving host opens the packet, it recalculates the checksum and compares its calculation with the one in the received packet. If they match, then authenticity of the packet is confirmed, and this packet is deemed to be the same packet as sent.

This provides message integrity while it traverses the network, once saved within a log collector message integrity becomes an issue again. Once this has been done, the authentication data associated with TCP is stripped off and the remaining message is sent up to the syslog application. To perform integrity checks on this message now, another form of authentication needs to be implemented by the syslog application itself.

During our discussion on hashing, we discovered only static files are valid candidates for hashing. A local log file is continually changing so does not fit this profile. If we consider that each message within a log file is static in of itself, we can hash each message individually and keep a separate table of these hashes.

### *Syslog sign*

A draft RFC titled "Syslog-Sign Protocol" has been issued that hopes to provide this form of message authentication (as of this writing draft-ietf-syslog-sign-11.txt[38]). Taken from the Abstract:

> Syslog-sign, a mechanism adding origin authentication, message integrity, replay-resistance, message sequencing, and detection of missing messages to syslog.

The SDSC Secure Syslog implementation that we have covered in this paper has this as a mandatory requirement[39] for implementation once the RFC is ratified for future releases.

The main idea behind syslog sign is to provide a mechanism that can be easily built on top of an RFC3164 compliant syslog daemon with minimum impact. Syslog messages are generated and delivered as normal, with syslog sign adding two additional message types that provide the true benefits of this protocol, signature blocks and certificate blocks.

Syslog sign is designed to utilise UDP as the transport mechanism but recognises RFC3195 may be used to provide reliable delivery of messages. Though all messages must conform to RFC3164, there is a slight superficial variance between this RFC and the original RFC3164. The tag field is now considered part of the header rather than the message or content. A less superficial change is RFC 3339 formatted timestamps should be used, we talked bout this during our discussion on time synchronisation.

[38] This document can be found at http://www.employees.org/~lonvick/index.shtml
[39] http://security.sdsc.edu/software/sdsc-syslog/requirements for the items they plan to incorporate in SDSC syslog

As each message is generated a signature of the entire syslog message is generated using a predetermined hashing algorithm. These signatures are collected together and placed within a signature block containing signature information of up to 99 hash values. Each individual hash ensures that non of the information has changed since when it was generated.

There are many fields within a signature block that help with message integrity, replay resistance, message sequencing, and detection of missing messages. The signature block has information to allow syslog sign protocol to keep track of how many signature blocks have been sent previous to this one, and how many message hashes have been sent previous and which are included within this signature block.

Syslog sign uses public key technology for key exchange. Keys are transported by certificate blocks which form part of the key management process used to verify messages as they arrive or after they have been saved to disk. Syslog sign uses these certificate blocks to share key material before sending messages. The keys are carried inside the payload block within the certificate block which has the role of getting the payload block to the collector.

There are two methods of verifying messages online and offline. This means you can verify messages as they arrive at the collector by building an authenticated log file, or an administrator may verify and view messages at regular intervals offline, as you would a normal log file. In reality it is a little different because an extra process of authentication is included.

When reviewing messages offline all messages are sent to a file as they are received, complete with signature blocks and certificate blocks. The administrator can then use a script to separate the certificate block, signature blocks, and messages into three separate files. The payload block contained within the certificate block is used to verify the integrity of the signatures which in turn verify the generated messages and show if there are any missing entries.

Online analysis is the real-time verification of messages as they arrive. The collector builds up a log file of messages that have been verified by the signature blocks as they enter.

## Encrypting Syslog Traffic

Throughout this paper we have seen network traces of log messages. So far they have all been in clear text, and they have been open for anyone to read who can sniff the network. This can provide an opening for attackers to gather some very valuable reconnaissance, such as what information you are monitoring or what actions will generate a response. This can allow an attacker to tailor their next move so that it might not be recorded.

Encrypting your data removes the chance that an attacker will be able to read your messages in real time. The attacker still has the ability to capture this traffic and then try to break the encryption code to read the message off line. The length of time it takes an attacker to break this code depends on the strength of the algorithm. The stronger the algorithm, the longer this will take, hopefully cracking the encryption will take so long that once it is decrypted the message is of no value any longer.

Take note of this very important point: the strength of the algorithm used should be determined by the length of time that you wish to keep the message secret.

How important is it to keep these messages private? In the case of authentication messages this could be very important. If there is an authentication message that sends failed username and password combinations within the syslog message it would be very easy for an attacker to use this against you. Most likely this error message is created by a user that has just incorrectly typed their password, which may only be by one character giving an attacker the keys to your systems. You should ensure that these types of messages are not generated in the first place. Knowing of a failed logon attempt is important, not the password/username combination.

With encryption of other messages you are reducing the attacker's ability to obtain reconnaissance from these messages, though there are other ways to obtain this information. Traffic analysis can still be performed by testing to see what actions will generate a message, even though the message is not readable you still have a clue as to the actions that will generate a message.

One way to combat this is to have a consistent flow of data that is both regular and consistent in size. Somehow you should be able to send all the legitimate data inside these packets and when there aren't any interesting logs to send, fake data is sent within the packets. These issues are consistent with the way we have to deal with IDS (Intrusion Detection System) traffic. The issue is that you remove your real time alerting, the information will need to be delayed until the next scheduled packet.

An encrypted packet provides protection because when it is encrypted only the people or systems with the right keys can decrypt the message. Firewalls and Network Intrusion Detection systems are unlikely to hold these keys.

One of the facets of a Network Intrusion Detection (NID) system is its ability to match pattens in the traffic. When this traffic is encrypted a NID has no ability to scan for suspicious behaviour and this reduces their effectiveness. One way to perform Intrusion Detection on this traffic is to look for suspicious behaviour at the sending or receiving host using Host Intrusion Detection (HID). The packet is decrypted at the host because it does have the required keys and the HID system can monitor the effects of this packet to make sure it does not do anything suspicious.

SSH is a program that allows you to send traffic through an encrypted tunnel. This tunnel can hold any type of data, it does not have to be syslog data. If we send this encrypted tunnel through a firewall, the firewall can do only one of two things, let the packet through or drop it. Dropping the packet could cause serious problems with communication, but because the packet cannot be further inspected, letting it through provides a path for any traffic to pass through the firewall that would otherwise not be allowed, a very undesirable result.

One way to still perform filtering on these packets it to decrypt them somewhere between the two communicating hosts. The firewall is one place where you could decrypt the data, perform the required filtering actions, and optionally re-encrypt the data before forwarding it onto the destination host. This can be complex and requires the firewall to hold required keys to perform this task. There are many issues that make this problematic.

## How About a Syslog Relay?

General network design, and designs for secure environments, data is usually directed through choke points. These choke points are ideal places for security controls such as packet filtering, Intrusion Detection, bandwidth limiting, etc. Some common locations for choke points are between departments or sites, between networks that have different levels of trust, like the Internet and your screened subnet and the screened subnet and your internal network.

Generally you will see a firewall at these points filtering the traffic as it passes through. A firewall can itself be a log relay or an additional box can perform this task. A Syslog relay placed at these points helps to control messages by limiting the source of data and forwarding required logs to your desired destination, which could be more than one log collector. In very large networks it may be beneficial to deliver logs within a local network to a log relay which in turn can forward these logs to a central log server at you head office or network operations centre.

A Syslog relay can perform another very important function and that is to convert non standard messages into standard ones. RFC3164 describes the requirements of a relay if it receives a message that is non standard in section 4.3. To sum up, if a non standard message is received by a relay agent it must modify the message to conform to RFC3164 before sending it on to the collector or another relay.

## Network Filters, Intrusion Detection and OOB networks

There are many aspects to network security, various security systems rely on other systems like firewalls, IDS systems, logging and network design, play a role in a whole security system. There are many points where these security devices actually interact and use the strengths that each other provide to build a total security solution.

### Network Filters

Iptables is the traffic filtering module built into the Linux 2.4.x kernels. When loaded and configured correctly it acts as a firewall that can be installed on a gateway located at a choke point, or on each individual host to protect it from local network based attacks. In a defence in depth approach, packet filtering should be on every host as you never know where that attack may come from, it could be the local network.

Syslog and iptables need each other, iptables logs messages to syslog and syslog needs iptables to protect the machine it is running on.

### OOB networks

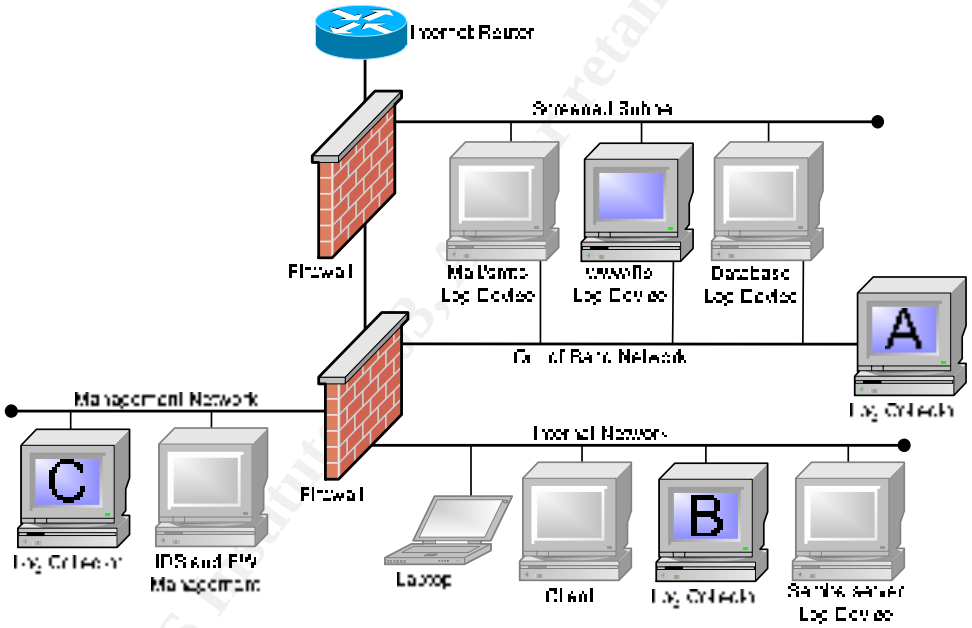



**Figure 3: Dual firewall network configuration depicting 3 possible log collector locations**

Network design that has security as a key requirement is very important. One design consideration is to use management and OOB (Out Of Band) networks. These networks provide protection because they are separated from data that flows over commonly used paths. No public network traffic should be seen on these networks.

A log collector would be protected in any of the three locations identified in the above diagram, lets take a look at the considerations of each location.

***Position A***

A log collector in this position is protected from being directly accessed from over the internet. This is a good position but relies on the security of each host within the screened subnet. If any of these hosts are compromised then it is only a matter of time before the attacker discovers the OOB network and try's to discover a way to further compromise your network.

Although I recommend placing filters on each interface for hosts within your perimeter they are not designed to be network filters and should not be relied upon for this function. Hosts on a screened subnet will have services that could have vulnerabilities discovered and therefore the system is at risk unless you patch or turn off the affected service. As long as you offer these services there is a level of risk. We have reduced this risk by placing these hosts on their own network segment but we should not rely on them to protect the log collector.

***Position B***

This looks like an even better idea. We now have a firewall that can reliably filter the traffic on the OOB network further before it enters our internal network. We still have a problem as many attacks are known to originate from the internal network. We have our log data situated in a compromising position, open to probing from internal users.

When considering the security policy for the internal firewall, I prefer to block all traffic from entering the internal network from any outside network, the firewall only allows traffic initiated from the inside. Having the log collector here would violate that good security policy.

***Position C***

Probably what could be considered the most secure position on your network is the management network. This network does not have user access, it does not have public access, only trusted administrators can access any of the systems on this network. The filtering into and out of this network should be very strict so if a host on the screened subnet is totally compromised the collector is still protected and users from the inside are unable to attack the collector.

Sending your data over the OOB to your management network still provides the ability for an attacker to sniff the traffic, but if the management network is correctly configured they will be able to do very little else.

Attacks can come from anywhere and you need to protect your valuable assets from all attack vectors internal or external. (I know I seem overly paranoid, but you need to consider everything). The eventual position you place your collector depends on your budget and aversion to risk. As has been stated many times the collector contains very important and confidential material, due diligence in protecting this machine the best way practical is required.

## Conclusion

There is a lot of material covering syslog and its implementation, hence the reason this paper is so long. Many people do not understand or even know many of the considerations when designing a networked environment. I hope I have opened your eyes to even one element that you had not considered before.

We have looked at ways to implement logging on your network, both local and network based. We covered ways to manage, store and secure logs, as well as the networking requirements with a central log collector. Finally we looked at the requirements to make a central log server more secure by using an alternate syslog daemon that provides RFC compliant reliable and secure logging.

A lot of material was covered and depending on your environment only certain aspects may be applicable. Hopefully you see logging in a new light and can make more informed decisions on how important it is to your company, including what needs to be done to secure a syslog implementation for maximum benefit.

Though there is much more that can be said about logging I hope I have covered the most essential elements, while also giving people who do not use syslog as their form of logging some ideas for use with what ever implementation they use.

## Appendix A: References

These are the reference that in one way or another helped form the content contained within this paper.

### *Books*

TCP/IP Illustrated, Volume 1 – The protocols
W. Richard Steven
Copyright © 1994 by Addison Wesley
ISBN 0-201-63346-9

BEEP: The Definitive Guide
Marshall T Rose
O'Reilly & Associates; 1st edition (March 2002)
(During the discussion on BEEP, some details were taken from the sample pages hosted on amazon.com
http://www.amazon.com/exec/obidos/tg/detail/-/0596002440/qid=1055727969/sr=8-5/ref=sr_8_5/002-0229526-3729618?v=glance&s=books&n=507846

### *Request for Comments*

RFC's can be located via http://www.rfc-editor.org

RFC 3080 “The Blocks Extensible Exchange Protocol Core”
M. Rose (Invisible Worlds, Inc.), March 2001

RFC 3164 “The BSD syslog Protocol”
C. Lonvick (Cisco Systems), August 2001

RFC 3195 “Reliable Delivery for Syslog”
D. New, M. Rose (Dover Beach Consulting, Inc.), November 2001

RFC 3227 “Guidelines for Evidence Collection and Archiving”
D. Brezinski and T. Killalea, February 2002

RFC 3339 “Date and Time on the Internet: Timestamps”
G. Klyne (Clearswift Corporation) and C. Newman (Sun Microsystems), July 2002

### *Internet-Draft*

Draft-ietf-syslog-sign-11.txt “Syslog-Sign Protocol”
J. Kelsey (Certicom) J. Callas (PGP Corporation), April 4, 2003
http://www.employees.org/~lonvick/index.shtml

### *CERT Coordination Centre*

Establish a policy and procedures that prepare your organization to detect signs of intrusion
http://www.cert.org/security-improvement/practices/p090.html

Identify data that characterize systems and aid in detecting signs of suspicious behaviour
http://www.cert.org/security-improvement/practices/p091.html

Manage logging and other data collection mechanisms

http://www.cert.org/security-improvement/practices/p092.html

Monitor and inspect system activities for unexpected behaviour
http://www.cert.org/security-improvement/practices/p095.html

Configuring and using syslogd to collect logging messages on systems running Solaris 2.x
http://www.cert.org/security-improvement/implementations/i041.08.html

***Internet sites***

Log Analysis Resources
Hosted by Counter Pane and maintained by Tina Bird
This site is highly recommended for further information on logging.
http://www.counterpane.com/log-analysis.html

***Internet based papers***

Presentation on Syslog from the ThaiCert "Thai Computer Emergency Response Team"
http://thaicert.nectec.or.th/event/itsec2002-material/logserver.pdf

Secure Remote Log Servers Using SCP by Kristy Westphal (February 2001)
http://www.securityfocus.com/infocus/1394

Complete Reference Guide to Creating a Remote Log Server by Eric Hines (August 2000)
http://www.linuxsecurity.com/feature_stories/remote_logserver-1.html

Advanced Log Processing by Anton Chuvakin (August 2002)
http://www.securityfocus.com/infocus/1613

Take Command: The System Logging Daemons, syslogd and klog by Michael A. Schwarz (July 2000)
http://www.linuxjournal.com/print.php?sid=4036

Linux Administrator's Security Guide - Log files and other forms of monitoring
by Kurt Seifried
http://www.windowsecurity.com/whitepapers/Linux_Administrators_Security_Guide__Log_files_and_other_forms_of_monitoring.html

Defending your log files by Ed Skoudis (September 2001_
http://www.informit.com/isapi/product_id~%7BF8B603B4-72C2-44EF-893C-58B6E9D1F5F1%7D/content/index.asp

Do you trust your system logs? By Alejo Sanchez (December 2001)
http://ezine.daemonnews.org/200112/log_protection.html

Rethinking UNIX System logging with SHARP by Matt Bing and Carl Erickson
http://www.csis.gvsu.edu/sharp/sharp.pdf

Securing Syslog on FreeBSD by Albert Mietus
http://eurobsdcon.org/papers/mietus_presentation.pdf

Syslog Overview v1.3 by Counterpane
http://www.counterpane.com/syslog-overview.pdf

### *Syslog Replacements*

SDSC Secure Syslog, an RFC3195 compliant syslog replacement
http://security.sdsc.edu/software/sdsc-syslog/

Msyslog (Modular Syslog)
http://sourceforge.net/projects/msyslog/

Syslog-ng (Syslog Next Generation)
http://freshmeat.net/projects/syslog-ng/

### *Red Hat 8.0 man pages*

This is a list of man pages that I referenced throughout the paper. Use the man command access these pages. For a description on how to use the man command type "`man man`" at the command prompt.

To look at the man pages listed here type "`man <number> <command>`" for example type "`man 3 syslog`" to view the syslog(3) man page.

| | | | |
|---|---|---|---|
| syslog(2) | klogd(8) | logrotate(8) | mkfifo(1) |
| syslog(3) | sysklogd(8) | logwatch(8) | |
| syslog.conf(5) | setlogmask(3) | logger(1) | |

## Appendix B: Red Hat 8.0 syslog.conf

```
# Log all kernel messages to the console.
# Logging much else clutters up the screen.
#kern.*                                                 /dev/console

# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none;cron.none                /var/log/messages

# The authpriv file has restricted access.
authpriv.*                                      /var/log/secure

# Log all the mail messages in one place.
mail.*                                                  /var/log/maillog

# Log cron stuff
cron.*                                                  /var/log/cron

# Everybody gets emergency messages
*.emerg                                                 *

# Save news errors of level crit and higher in a special file.

uucp,news.crit                                          /var/log/spooler

# Save boot messages also to boot.log
local7.*                                                /var/log/boot.log
```

## Appendix C: Explanation of tcpdump command and output

### Tcpdump command line options:

```
sudo /usr/sbin/tcpdump -s0 -X -vv -nn -i eth0
```

Command description

| | |
|---|---|
| `sudo` | This allows a normal user to execute commands as another user, in this case as root, `tcpdump` needs root privileges to open the network interface in promiscuous mode. |
| `/usr/sbin/tcpdump` | The actual executable that captures data from the network. The reason I use the full path is to ensure I am running the correct executable not another one that could be in my path, just a paranoid though good habit to be in. |
| `-s0` | Capture the whole packet. |
| `-X` | Tells `tcpdump` to print the packet in hex as well as ASCII. |
| `-vv` | Print very verbose information. |
| `-nn` | This is a command switch that stops `tcpdump` from resolving the names of IP address, protocol and port numbers as it collects data. This saves processing power and time. Besides, in my opinion, it is easier to work with numbers. If I need the name I will look it up via other means. |
| `-i eth0` | this command switch directs `tcpdump` to listen to a particular network interface 'eth0' |

### Tcpdump example of a UDP packet:

```
16:36:06.298930 192.168.201.20.514 > 192.168.201.10.514:  [udp sum ok] udp
44 (DF) (ttl 64, id 0, len 72)
0x0000   4500 0048 0000 4000 4011 2735 c0a8 c914        E..H..@.@.'5....
0x0010   c0a8 c90a 0202 0202 0034 6d77 3c31 3333        .........4mw<133
0x0020   3e44 4556 4943 4531 3a20 6d65 7373 6167        >DEVICE1:.messag
0x0030   6520 6c6f 6361 6c30 2e6e 6f74 6963 6520        e.local0.notice.
0x0040   6465 7669 6365 310a                            device1.
```

Output description:

| [Timestamp] | [Source IP and Port] | [Destination IP and Port] |
|---|---|---|
| `16:36:06.298930` | `192.168.201.20.514 >` | `192.168.201.10.514` |
| [Status of UDP checksum] | [protocol used] | [Data bytes in UDP body] |
| `[udp sum ok]` | `udp` | `44` |
| [Signal not to Fragment packet] | [Time to live] | [IP ID] |
| `(DF)` | `(ttl 64,` | `id 0,` |

[Total length of IP packet]
`len 72)`

| | [HEX output of packet] | [ASCII of packet] |
|---|---|---|
| `0x0000` | `4500 0048 0000 4000 4011 2735 c0a8 c914` | `E..H..@.@.'5....` |
| `0x0010` | `c0a8 c90a 0202 0202 0034 6d77 3c31 3333` | `.........4mw<133` |
| `0x0020` | `3e44 4556 4943 4531 3a20 6d65 7373 6167` | `>DEVICE1:.messag` |
| `0x0030` | `6520 6c6f 6361 6c30 2e6e 6f74 6963 6520` | `e.local0.notice.` |
| `0x0040` | `6465 7669 6365 310a` | `device1.` |

**Tcpdump example of a TCP packet:**

```
10:09:04.971538 207.228.225.5.511 > x.x.x.194.511: SF [tcp sum ok]
2118914133:2118914133(0) win 1028 (ttl 29, id 39426, len 40)
0x0000     4500 0028 9a02 0000 1d06 d1ec cfe4 e105          E..(............
0x0010     xxxx xxc2 01ff 01ff 7e4c 1055 7beb a83a          ........~L.U{..:
0x0020     5003 0404 c336 0000 0000 0000 0000               P....6........
```

Output description:

| [Timestamp] | [Source IP and Port] | [Destination IP and Port] |
|---|---|---|
| `10:09:04.971538` | `207.228.225.5.511 >` | `x.x.x.194.511:` |
| [TCP flags] | [Status of TCP checksum] | [TCP sequence numbers] |
| `SF` | `[tcp sum ok]` `2118914133:2118914133` | |
| [Bytes of data in TCP data] | [TCP window size] | [Time to live] |
| `(0)` | `win 1028` | `(ttl 29,` |
| [IP ID] | [Total length of IP packet] | |
| `id 39426,` | `len 40)` | |

| | [HEX output of packet] | [ASCII of packet] |
|---|---|---|
| `0x0000` | `4500 0028 9a02 0000 1d06 d1ec cfe4 e105` | `E..(............` |
| `0x0010` | `xxxx xxc2 01ff 01ff 7e4c 1055 7beb a83a` | `........~L.U{..:` |
| `0x0020` | `5003 0404 c336 0000 0000 0000 0000` | `P....6........` |

For a more comprehensive description of `tcpdump` please refer to the `tcpdump` man page by typing '`man tcpdump`' on a *NIX host that has the product running or visit http://www.rt.com/man/tcpdump.1.html for an online version taking particular note of the section labels 'Output Format'.

## Appendix D: Explanation of hping2 command

```
sudo ./hping2 -I eth0 -c 1 -a 192.168.201.20 -y -2 -s 514 -p 514 \
-d 54 -E ../udpbody 192.168.201.10
```

Command description

| | |
|---|---|
| `sudo` | this allows a normal user to execute commands as another user, in this case as root, hping2 needs root privileges to open a raw network socket. |
| `./hping2` | the executable that generates the packet. |
| `-I eth0` | send the packet out this interface. |
| `-c 1` | creates only one packet and finish. |
| `-a 192.168.201.20` | use this as the source address (spoof IP address). |
| `-y` | Don't fragment the packet. |
| `-2` | Use the UDP protocol. |
| `-s 514` | Use source port 514. |
| `-p 514` | Use destination port 514. |
| `-d 54` | Use a payload size of 54 bytes for the UDP packet. |
| `-E ../udpbody` | Use the contents of file as the UDP payload. |
| `192.168.201.10` | Send packet to this address. |

## Appendix E: PRI matrix

Within a syslog packet there is a PRI field specially calculated from the numerical value of the facility and the numerical value of the severity. According to RFC 3164 (4.1.1):

> The priority value is calculated by first multiplying the Facility number by 8 and then adding the numerical value of the severity.

This table is designed to make it easy to determine what Facility and Severity a message is when all you have is the calculated number, this could be the case if you are debugging a problem while sniffing traffic over the network. For example a Red Hat 8.0 syslog packet as it traverses the network may look like:

```
<13>shutdown:.shutting.down.for.system.reboot.
```

The PRI value is 13; looking at the matrix we can see this was a message for the user facility with a severity of notice (user.notice)

| | | Severity | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | **Emerg** | **Alert** | **Critical** | **Error** | **Warn** | **Notice** | **Info** | **Debug** |
| **Facility Level** | | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| **Kernel** | **0** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| **User** | **1** | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| **Mail** | **2** | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| **System** | **3** | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| **Security** | **4** | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 |
| **Syslog(d)** | **5** | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| **Line printer** | **6** | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 |
| **Network news** | **7** | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |
| **UUCP** | **8** | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 |
| **Clock** | **9** | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 |
| **Security** | **10** | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 |
| **FTPd** | **11** | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 |
| **NTPd** | **12** | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 |
| **Log audit** | **13** | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 |
| **Log alert** | **14** | 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119 |
| **Clock daemon** | **15** | 120 | 121 | 122 | 123 | 124 | 125 | 126 | 127 |
| **Local 0** | **16** | 128 | 129 | 130 | 131 | 132 | 133 | 134 | 135 |
| **Local 1** | **17** | 136 | 137 | 138 | 139 | 140 | 141 | 142 | 143 |
| **Local 2** | **18** | 144 | 145 | 146 | 147 | 148 | 149 | 150 | 151 |
| **Local 3** | **19** | 152 | 153 | 154 | 155 | 156 | 157 | 158 | 159 |
| **Local 4** | **20** | 160 | 161 | 162 | 163 | 164 | 165 | 166 | 167 |
| **Local 5** | **21** | 168 | 169 | 170 | 171 | 172 | 173 | 174 | 175 |
| **Local 6** | **22** | 176 | 177 | 178 | 179 | 180 | 181 | 182 | 183 |
| **Local 7** | **23** | 184 | 185 | 186 | 187 | 188 | 189 | 190 | 191 |

Last Updated: September 17th, 2019

# Upcoming SANS Training

**Click here to view a list of all SANS Courses**

| | | | |
|---|---|---|---|
| SANS San Francisco Fall 2019 | San Francisco, CAUS | Sep 23, 2019 - Sep 28, 2019 | Live Event |
| SANS Dallas Fall 2019 | Dallas, TXUS | Sep 23, 2019 - Sep 28, 2019 | Live Event |
| SANS London September 2019 | London, GB | Sep 23, 2019 - Sep 28, 2019 | Live Event |
| SANS Kuwait September 2019 | Salmiya, KW | Sep 28, 2019 - Oct 03, 2019 | Live Event |
| SANS Tokyo Autumn 2019 | Tokyo, JP | Sep 30, 2019 - Oct 12, 2019 | Live Event |
| SANS Cardiff September 2019 | Cardiff, GB | Sep 30, 2019 - Oct 05, 2019 | Live Event |
| SANS Northern VA Fall- Reston 2019 | Reston, VAUS | Sep 30, 2019 - Oct 05, 2019 | Live Event |
| SANS DFIR Europe Summit & Training 2019 - Prague Edition | Prague, CZ | Sep 30, 2019 - Oct 06, 2019 | Live Event |
| Threat Hunting & Incident Response Summit & Training 2019 | New Orleans, LAUS | Sep 30, 2019 - Oct 07, 2019 | Live Event |
| SANS Riyadh October 2019 | Riyadh, SA | Oct 05, 2019 - Oct 10, 2019 | Live Event |
| SANS Baltimore Fall 2019 | Baltimore, MDUS | Oct 07, 2019 - Oct 12, 2019 | Live Event |
| SANS October Singapore 2019 | Singapore, SG | Oct 07, 2019 - Oct 26, 2019 | Live Event |
| SANS Lisbon October 2019 | Lisbon, PT | Oct 07, 2019 - Oct 12, 2019 | Live Event |
| SANS San Diego 2019 | San Diego, CAUS | Oct 07, 2019 - Oct 12, 2019 | Live Event |
| SIEM Summit & Training 2019 | Chicago, ILUS | Oct 07, 2019 - Oct 14, 2019 | Live Event |
| SANS Doha October 2019 | Doha, QA | Oct 12, 2019 - Oct 17, 2019 | Live Event |
| SANS Seattle Fall 2019 | Seattle, WAUS | Oct 14, 2019 - Oct 19, 2019 | Live Event |
| SANS SEC504 Madrid October 2019 (in Spanish) | Madrid, ES | Oct 14, 2019 - Oct 19, 2019 | Live Event |
| SANS Denver 2019 | Denver, COUS | Oct 14, 2019 - Oct 19, 2019 | Live Event |
| SANS London October 2019 | London, GB | Oct 14, 2019 - Oct 19, 2019 | Live Event |
| SANS Cairo October 2019 | Cairo, EG | Oct 19, 2019 - Oct 24, 2019 | Live Event |
| SANS Santa Monica 2019 | Santa Monica, CAUS | Oct 21, 2019 - Oct 26, 2019 | Live Event |
| Purple Team Summit & Training 2019 | Las Colinas, TXUS | Oct 21, 2019 - Oct 28, 2019 | Live Event |
| SANS Training at Wild West Hackin Fest | Deadwood, SDUS | Oct 22, 2019 - Oct 23, 2019 | Live Event |
| SANS Orlando 2019 | Orlando, FLUS | Oct 28, 2019 - Nov 02, 2019 | Live Event |
| SANS Houston 2019 | Houston, TXUS | Oct 28, 2019 - Nov 02, 2019 | Live Event |
| SANS Amsterdam October 2019 | Amsterdam, NL | Oct 28, 2019 - Nov 02, 2019 | Live Event |
| SANS DFIRCON 2019 | Coral Gables, FLUS | Nov 04, 2019 - Nov 09, 2019 | Live Event |
| Cloud & DevOps Security Summit & Training 2019 | Denver, COUS | Nov 04, 2019 - Nov 11, 2019 | Live Event |
| SANS Paris November 2019 | Paris, FR | Nov 04, 2019 - Nov 09, 2019 | Live Event |
| SANS Sydney 2019 | Sydney, AU | Nov 04, 2019 - Nov 23, 2019 | Live Event |
| SANS Mumbai 2019 | Mumbai, IN | Nov 04, 2019 - Nov 09, 2019 | Live Event |
| SANS Bahrain September 2019 | OnlineBH | Sep 21, 2019 - Sep 26, 2019 | Live Event |
| SANS OnDemand | Books & MP3s OnlyUS | Anytime | Self Paced |