Global Information Assurance Certification Paper

Secure Logging of Hosts Across a Firewall

Robert Early
July 9, 2002
GSEC Practical Assignment v1.4, Option 2

Introduction

This paper describes a method in which centralised logging with a single server can be achieved across a network which has segments separated by a firewall. It is shown how a typical security policy in an organisation can result in firewall rules preventing syslog traffic from crossing the firewall. The solution chosen is one suitable for a small organisation and is based on freely available tools and applications. The result is a secure method for the consolidation of logs across firewalled networks.

Description of the Problem

Accurate and reliable system logging is essential for any computing environment for ensuring the three main principles of security, namely Confidentiality, Integrity and Availability. In terms of confidentiality, logging helps ensure that only those who should have access to systems and data actually do. Logging of failed attempted access warns that a confidentiality attack has or is taking place for example. System integrity cannot be guaranteed without accurate and detailed logging. If a system were broken into, it may not be detected if adequate system logging is in place. It is possible to measure system availability through logging. For example logging when services are malfunctioning would be an example of assuring availability through the use of logging.

When choosing a method for system logging, one must consider syslog as it has universal acceptance across every flavour of Unix. The level and range of logging is highly configurable:

> The syslog system is one of the most delightful things about Unix. Unlike some operating systems that force you to use the limited range of logs that they condescend to provide, Unix allows you to log almost anything, at almost any level of detail. While system logging hooks are provided for the most common Unix resources, administrators can choose a logging configuration that meets their needs (Lucas).

This versatility has resulted in syslog becoming the de-facto standard for system logging on Unix hosts. It is even available for Windows systems as a third-party add-on. Winsyslog (http://www.winsyslog.com/en/Product/) is an example of this. For these reasons I have chosen to focus on this system for the purposes of this paper.

The standard syslogd system however is inflexible when it comes to logging across secured networks. In the case of a common corporate LAN configuration, where there is a firewall or firewalls separating the public Internet connection, the internal network/networks and a DMZ which contains the public-facing servers such as Web, Ftp, email, etc., syslog often cannot be used as-is. Security policy in a corporation often defines that port 514 is blocked, amongst others. In some cases the entire UDP protocol is blocked. This is due to the large number of exploits using the UDP protocol. The disadvantage due to the loss of UDP is usually outweighed by the gain in security. Most services can operate on TCP and for those that cannot, the servers with UDP services can be placed on the DMZ. An example of this is the DNS service. Corporations that require public-facing DNS services often place their DNS servers outside the firewall on the DMZ LAN. Since syslog uses UDP on port 514, it cannot operate across the firewall in this situation. This makes a single syslog server collecting the logs from all the servers in the corporation impossible without devising some sort of system to overcome this.

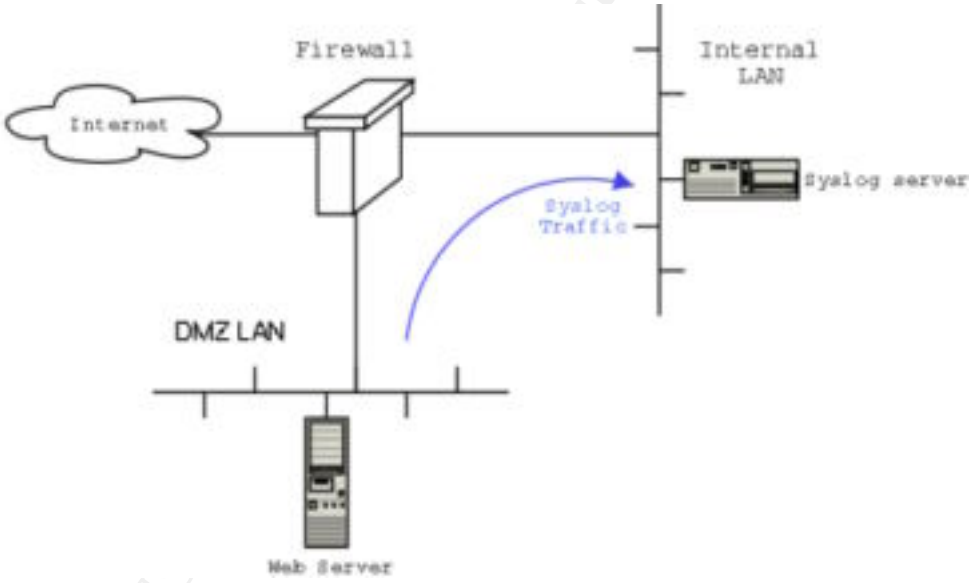

Fig 1

Here in Fig 1 we can see a graphic representation of the LAN configuration. The Firewall in this case is a machine which has three network interfaces and runs a firewall application to filter and pass traffic between them. This has the effect of creating three separate logical networks between which traffic is exchanged. In our case all inward traffic from the Internet to the DMZ network is filtered by port number and protocol to only allow the provision of web services, FTP, email, etc. The firewall disallows all traffic directly inward to the internal LAN from the Internet. Any traffic initiated from the internal LAN to either the DMZ or Internet is allowed by default.

Analysis of Possible Solutions

One possible solution to the problem would be to set up a separate syslog server on the DMZ LAN to serve the machines there. The problem with this is that it could possibly leave the log server on the DMZ LAN more exposed to attack than the syslog server on the internal LAN. This is due to the less restrictive policy allowing certain types of traffic on to the DMZ LAN from the Internet. A denial of service attack on the syslog server on the DMZ LAN could hide an intrusion on the other machines on the same network. Another factor to take into account is that some organisations may not be able to afford a second (or third, depending on the number of separated networks) server to perform logging on each network. There is also an administrative overhead in setting the machines up and configuring monitoring for the logs generated on each one.

Another possible solution would be to set up VPNs or tunnels through the firewall between the syslog server located on the internal LAN and the machines being monitored on the DMZ. Such a solution is proposed by James Hunter in his paper, "Central Logging Security" (http://rr.sans.org/unix/logging_sec.php).
This however is undesirable in our case, since the syslog server could still be subject to a DoS attack if a monitored host is compromised. This is due to the fact that a tunnel allows traffic in either direction. If a monitored host on the DMZ LAN were compromised, it would be easy to send data to the port on the monitored host that could overload the syslog server or fill up it's disks. Such a scenario would be possible for anyone with sufficient privileges on the monitored host. Another reason why this may be unsuitable would be the fact that it contravenes defined security policies as it bypasses firewall controls and any logging of content. If an encrypted tunnel were used, the firewall could not "see" the contents of the traffic. If a network-based IDS were in operation, the system could not detect any attacks via the tunnel as the data would be encrypted and hidden between the hosts. The option of leaving the tunnel data unencrypted would mean that it would be vulnerable to spoofing. This would reduce the usefulness of the log files, as they could not be completely trusted.

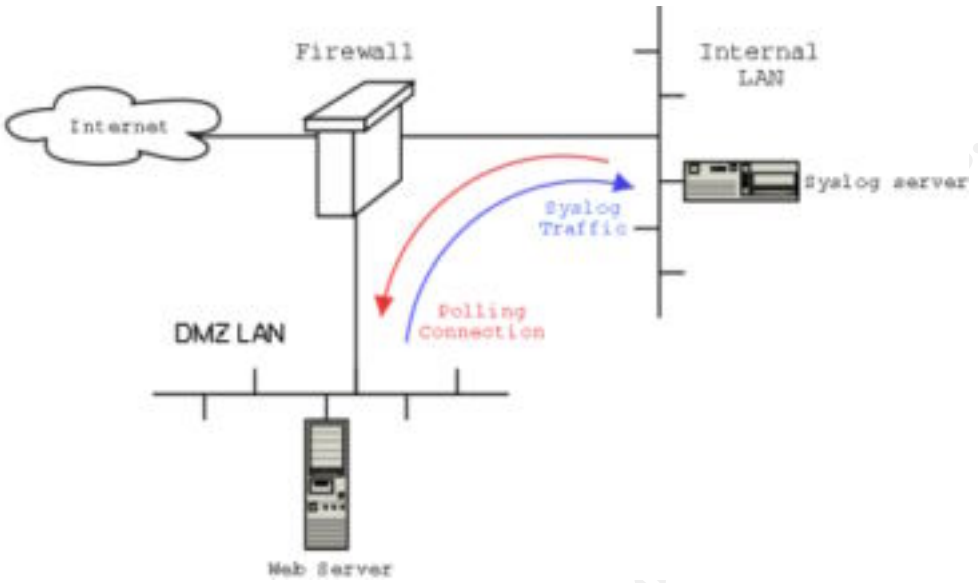

Fig 2.

The last suggestion, and the one which was chosen for implementation in this case, was to have the syslog server initiate connections to the monitored host. This would get around the problem of not being able to directly log to the log server. This connection could be encrypted, to get around the problem of the possibility of spoofed messages or having the stream otherwise tampered with. This could be configured to connect to the monitored host periodically and the log entries generated since the last connection would be downloaded and the connection terminated. The frequency of this polling could be adjusted to whatever was deemed appropriate for the situation. We can see in Fig 2 how this is represented graphically. The polling connection is represented by the red arrow and the direction of the data transfer in blue. SSH was used for the connection, thereby satisfying the need for security and strong authentication in one.

Implementation

The process of logging and the analysis of the logs are broken into three areas. Firstly the method of extracting the logs from the monitored host is examined. This is dealt with under the section entitled "Logtail." Next, the nature and method of the connection used to retrieve the logs is detailed under "OpenSSH" and finally the method of analysing the consolidated logs is looked at in "Swatch."

Logtail:

In considering what method to use to get the data from the monitored host to the logging server, probably the most simplest method would be to copy the entire log file from the monitored host to the syslog server on every poll. This however

causes problems on the second and subsequent occasions this is done. This is because the logging daemon on the monitored host writes continuously to a file until it is rotated out. If we were to copy the entire log file every time, the entries copied up until the first point would be duplicated in the second retrieval and so on. What is needed is a mechanism to remember the last position of the file and extract the lines that have been written since the last access time.

Rather than writing an application specifically for this purpose, one was found which does this. Psionic Software's LogSentry package comes with a program called logtail, which performs this exact task and is freely available under the GNU General Public License. This package was obtained from Psionic's website at http://www.psionic.com/products/logsentry.html. Once this is downloaded, gunzip and untar the package to extract the source. Then change into the *logsentry-1.x/src* directory. It is only necessary to compile the logtail.c source file, so simply execute:

```
gcc logtail.c -o logtail
```

This example uses gcc, but your favourite C compiler can be used instead. The resultant binary is called logtail. This binary is then installed on the monitored machine in an appropriate directory, e.g. /usr/local/bin.

OpenSSH:

To provide the encryption and authentication portions of the connection, OpenSSH was chosen since it is a commonly used and very reliable application for secure connections. The source of OpenSSH was downloaded from http://www.openssh.org and built for each of the systems and installed.

In our configuration, we needed to build OpenSSH for RedHat Linux and HPUX 11.00 platforms. The version of the source that was obtained was the portable version for this reason. This is denoted by the "p" in the file version. In this case it was openssh-3.4p1. Building the source was straightforward, simply running the following commands to build the source with the default options.

```
configure
make
make install
```

The binaries were installed in /usr/local/bin and /usr/local/sbin as a result, with the other associated files being placed in their respective directories. Once this was done, the sshd daemon was started on the monitored hosts so that connections could be made to them.

For the purposes of this project, we needed to have the host system to log into the monitored hosts periodically and non-interactively to run the logtail command. To allow this to happen, a trust relationship was required between the machines. This allows a connection to be made without the use of a passphrase, in a similar

manner to the "r" tools in Unix: rlogin, rsh etc..
The method for setting this up is described in the article "SSH: The Secure Shell (The Definitive Guide)" at http://www.snailbook.com/faq/trusted-host-howto.auto.html. To allow this, the public key of the logging host is copied to the monitored hosts and placed in the known hosts file there. This defaults to /etc/ssh/ssh_known_hosts2 for standard installations.

The value "HostbasedAuthentication yes" also has to be set on both machines in the /etc/ssh/sshd_config file. Once this is configured, the setup can be tested by trying to run ssh to connect to the other machine. If the login is successful then the configuration is correct. An example of the expected output is given below:

```
[root@loghost files]# ssh remotehost
Last login: Tue Jun 25 15:15:48 2002 from otherserver

[root@remotehost user1]#
```

If host based authentication is configured correctly, the user should not be prompted for a passphrase, in this case for the root user. In the case of our configuration, the ssh port was initially blocked by the firewall by default for all outgoing connections. This was enabled by allowing TCP port 22 traffic from the internal network to the DMZ. This was restricted by IP address to allow only the syslog server to connect to only the monitored hosts.

A one-line script was then written on the syslog host to run SSH and call the logtail application on the remote hosts. The output of this was then piped to logger to insert the log lines into the local syslog system. The resultant line is given below:

```
ssh root@remotehost logtail /var/log/messages | logger
```

This was configured to be run by cron every five minutes. The log entry was thus:

```
*/5 * * * * ssh root@remotehost logtail /var/log/messages |
logger
```

The frequency of the connection can be set to whatever is deemed appropriate for the situation. For higher volume applications a shorter time interval would be required. If logging was only occasional then a longer time period may be desired. Since the output is passed to logger, the log line is preceded by the time at which the log was inserted into the syslog server logs. This is useful for correlating the time the entry was made and when it was initially recorded. An example of the log entries are given below:

```
Jun 27 13:30:01 loghost logger: Jun 27 13:42:35 logged-
server : su : + 2 user1-root
Jun 27 13:30:01 loghost logger: Jun 27 13:42:41 logged-
server syslogd: going down on signal 15
Jun 27 13:30:01 loghost logger: Jun 27 13:45:26 logged-
server xntpd[641]: xntpd version  3.5f: Sat Apr 14 15:04:53
IST 2001  PHNE_23697
Jun 27 13:30:01 loghost logger: Jun 27 13:45:26 logged-
server xntpd[641]: tickadj = 625, tick = 10000, tvu_maxslew
= 61875
Jun 27 13:30:01 loghost logger: Jun 27 13:45:26 logged-
server xntpd[641]: precision = 6 usec
Jun 27 13:35:01 loghost logger: Jun 27 13:45:36 logged-
server sshd[714]: Server listening on 0.0.0.0 port 22.
```

Swatch:
Now that we have the log entries integrated into the syslog system of the logging host, the final task is to process these logs to produce automatic warnings and alerts. Swatch, another free package which is very popular for analysing and monitoring log files, was chosen to do this. Swatch is produced by Todd Atkins and is also available under the GNU General Public License also.

The source file was obtained from ftp://ftp.stanford.edu/general/security-tools/swatch/swatch-3.0.4.tar.gz. Since the file obtained was a plain tar file and had no built-in integrity checking, an md5sum check was performed to ensure it's cryptographic integrity. This was matched with the md5 signature as given on the Swatch home page. The tar file was gunzipped and untared into a working directory

```
[myhost user1]$ md5sum swatch-3.0.4.tar.gz
ce290dd2cae6ce834f59e24d97a30d3b  swatch-3.0.4.tar.gz
```

The installation instructions differed from the OpenSSH setup slightly and resemble building a Perl CPAN module. To build the source, the following commands were issued as per the supplied instructions:

```
perl Makefile.PL
make
make test
make install
```

The initial compilation failed due to some essential modules that were missing namely Date/Calc.pm, Date/Parse.pm, File/Tail.pm and Time/HiRes.pm. These were obtained from http://www.cpan.org, compiled and installed.

Once Swatch was compiled and installed, the next step was to configure the triggers so that alerts are given according to network events. A couple of triggers were added initially, which could be expanded later on. Below is an excerpt from the /export/swatch.conf file:

```
#Some keywords to watch for, mail root on match
watchfor /FAILURE/
        exec=mail root

watchfor /REFUSED/
        exec=mail root

watchfor /ILLEGAL/
        exec=mail root

watchfor /attackalert/
        exec=mail root

watchfor /BAD/
        exec=mail root
```

Additional configuration is necessary to start up Swatch automatically and also to restart the daemon when the log files are rotated. Scripts to do these were obtained from:
http://www.cert.org/security-improvement/implementations/i042.01.html.

Observations

One peculiarity of the system was that a log entry was written to syslog every time there was a connection made by SSH. This included every time the logging server connected to the monitored host. The result being if one were to configure swatch to trigger on a successful connection by SSH, it would unfortunately trigger every time the log was retrieved. So it would produce an alert every five minutes in this configuration. This however was not a problem in our case since we were not interested in successful SSH connections, only failed ones. Swatch was not configured to trigger on successful SSH connections. This behaviour can be switched off in the OpenSSH configuration however if it presented a problem. In a low-volume log, real log entries could become swamped by the automatic SSH connects generated by the syslog system.

End Result

The resultant system is one that allows the monitoring of hosts across networks which are separated by a firewall. It also has the additional protection that the connection between the two hosts cannot be easily used to compromise the syslog server if a monitored host is compromised. One potential weakness of the system is that if the log file on the monitored host filled too quickly, the syslog server could potentially loose messages or the partition holding the logs could fill up. This however is also a potential problem with syslog in general as it does not do any authentication with the source host. Anyone with access to the syslog port on the log server could flood the server with messages to overwhelm it. In our configuration, an attacker would have to compromise the logged host first.

To overcome this weakness, it would be possible to modify the logtail application or make a more intelligent script for retrieval that would detect and prevent log entries being transferred too rapidly, as an extra layer of defence.

This also may not be suitable for situations that require real-time log analysis as the retrieval script only runs periodically. The smallest interval that can be specified on the standard Vixie cron is once every minute. If this window is too large then a solution involving a permanently open connection may be more suitable.

The result achieved increased the security of the network by:

a) providing a mechanism by which hosts in the DMZ LAN can be monitored

b) not requiring any new ports in the inward direction to be opened in the firewall

c) requiring that all connections be made outwards from the syslog host, thereby reducing the possibility of an attack being launched from a compromised host on the DMZ network.

Summary

Although such a system could be built using commercial tools, we have set up a system using freely available software. It is not designed with high performance or a high degree of scalability in mind but should work for organisations with up to tens of hosts to be monitored. For those who would like to set up a single syslog server, logging hosts across firewalls with freely available tools then this would be one possible solution.

References

Barrett & Silverman "How do I get trusted-host (SSH-2 hostbased, SSH-1 "RhostsRSA") authentication working?" SSH: The Secure Shell (The Definitive Guide). 2 June 2001. URL: http://www.snailbook.com/faq/trusted-host-howto.auto.html (18 May 2002)

Lucas, Michael. "System Logging." Big Scary Daemons. 17 May 2001. URL: http://www.onlamp.com/pub/a/bsd/2001/05/17/Big_Scary_Daemons.html (20 May 2002)

Hunter, James. "Central Logging Security." November 25, 2000. URL: http://rr.sans.org/unix/logging_sec.php (8 June 2002)

"Configuring and using syslogd to collect logging messages on systems running Solaris 2.x." 29 January 2001. URL: http://www.cert.org/security-improvement/implementations/i041.08.html (28 May 2002)

"Installing, configuring, and using swatch 2.2 to analyze log messages on systems running Solaris 2.x." 15 Mar. 2000. URL: http://www.cert.org/security-improvement/implementations/i042.01.html (17 June 2002)

Shaul, Matthew. "Using Swatch to Utilize Your Logs." 7 May 2001. URL: http://rr.sans.org/sysadmin/swatch.php (15 June 2002)

Last Updated: October 19th, 2017

Upcoming Training

SANS vLive - SEC401: Security Essentials Bootcamp Style	SEC401 - 201710,	Oct 23, 2017 - Nov 29, 2017	vLive
SANS San Diego 2017	San Diego, CA	Oct 30, 2017 - Nov 04, 2017	Live Event
San Diego Fall 2017 - SEC401: Security Essentials Bootcamp Style	San Diego, CA	Oct 30, 2017 - Nov 04, 2017	vLive
SANS Seattle 2017	Seattle, WA	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS Gulf Region 2017	Dubai, United Arab Emirates	Nov 04, 2017 - Nov 16, 2017	Live Event
Community SANS Vancouver SEC401^	Vancouver, BC	Nov 06, 2017 - Nov 11, 2017	Community SANS
Community SANS Colorado Springs SEC401~	Colorado Springs, CO	Nov 06, 2017 - Nov 11, 2017	Community SANS
SANS Miami 2017	Miami, FL	Nov 06, 2017 - Nov 11, 2017	Live Event
SANS Sydney 2017	Sydney, Australia	Nov 13, 2017 - Nov 25, 2017	Live Event
SANS Paris November 2017	Paris, France	Nov 13, 2017 - Nov 18, 2017	Live Event
SANS San Francisco Winter 2017	San Francisco, CA	Nov 27, 2017 - Dec 02, 2017	Live Event
Community SANS St. Louis SEC401	St Louis, MO	Nov 27, 2017 - Dec 02, 2017	Community SANS
SANS London November 2017	London, United Kingdom	Nov 27, 2017 - Dec 02, 2017	Live Event
Community SANS Portland SEC401	Portland, OR	Nov 27, 2017 - Dec 02, 2017	Community SANS
SANS Khobar 2017	Khobar, Saudi Arabia	Dec 02, 2017 - Dec 07, 2017	Live Event
SANS Austin Winter 2017	Austin, TX	Dec 04, 2017 - Dec 09, 2017	Live Event
SANS Munich December 2017	Munich, Germany	Dec 04, 2017 - Dec 09, 2017	Live Event
Community SANS Ottawa SEC401	Ottawa, ON	Dec 04, 2017 - Dec 09, 2017	Community SANS
SANS vLive - SEC401: Security Essentials Bootcamp Style	SEC401 - 201712,	Dec 11, 2017 - Jan 24, 2018	vLive
SANS Bangalore 2017	Bangalore, India	Dec 11, 2017 - Dec 16, 2017	Live Event
SANS Cyber Defense Initiative 2017	Washington, DC	Dec 12, 2017 - Dec 19, 2017	Live Event
SANS Cyber Defense Initiative 2017 - SEC401: Security Essentials Bootcamp Style	Washington, DC	Dec 14, 2017 - Dec 19, 2017	vLive
Community SANS Nashville SEC401^	Nashville, TN	Jan 08, 2018 - Jan 13, 2018	Community SANS
Community SANS Hawaii SEC401	Honolulu, HI	Jan 08, 2018 - Jan 13, 2018	Community SANS
SANS Security East 2018	New Orleans, LA	Jan 08, 2018 - Jan 13, 2018	Live Event
Mentor Session - SEC401	Memphis, TN	Jan 09, 2018 - Mar 13, 2018	Mentor
Northern VA Winter - Reston 2018	Reston, VA	Jan 15, 2018 - Jan 20, 2018	Live Event
SANS Amsterdam January 2018	Amsterdam, Netherlands	Jan 15, 2018 - Jan 20, 2018	Live Event
Mentor Session - SEC401	Minneapolis, MN	Jan 16, 2018 - Feb 27, 2018	Mentor
SANS Las Vegas 2018	Las Vegas, NV	Jan 28, 2018 - Feb 02, 2018	Live Event
Las Vegas 2018 - SEC401: Security Essentials Bootcamp Style	Las Vegas, NV	Jan 28, 2018 - Feb 02, 2018	vLive