Global Information Assurance Certification Paper

Central Logging with a Twist of COTS in a Solaris Environment

Kent Stout
SANS GSEC v1.3
March 25, 2002

II. Introduction

Logging can be a security administrator's best friend. It's like an administrative partner that is always at work, never complains, never gets tired, and is always on top of things. If properly instructed, this partner can provide the time and place of every event that has occurred in your network or system. However, it is up to you as the security administrator to properly instruct and use this potentially invaluable partner, and then take action on the information that it provides.

In this paper we will take a look at how to set up a central logging system for a network of Solaris machines, some of which will be hosting COTS products. The Solaris version that will be assumed for this paper is Solaris 8. The first area that will be covered is the configuration of a central log server. This will include the steps needed to prepare the log server to properly process and protect the various types of log messages that it will be receiving. Next, we will step through the configuration process for each type of logging client, preparing them to send all relevant log messages to the central log server. Parts of this process will be different for each given client based upon what applications it is hosting. The COTS products that will be used as examples in this paper include the iPlanet Web Server version 6.0 and iPlanet Directory Server (LDAP) version 5.0. Each logging client will also be configured to run the infamously verbose SunSHIELD Basic Security Module logging system, or BSM as we will call it. Log analysis is beyond the scope of this paper and will not be discussed.

III. Central Log Server

A. Machine Specifications

The exact specifications of a machine that is going to act as a central log server are obviously dictated by the number of logging clients and the amount of log messages that these clients are going to generate, but there are a couple of general rules that need to be followed when selecting a machine for the log server.

The first rule is that the log server needs to have a substantial amount of disk storage in order to hold all of the log messages that it is going to receive. Log files will need to be rotated and archived to some type of write-once medium, such as a WORM drive, on a regular basis to keep them manageable. This rotation scheme is a contributing factor when deciding how much disk storage is needed on the actual machine. If the rotation scheme is lethargic (once a month), then the log server will generally need a larger amount of disk

storage. On the other hand, if the rotation scheme is very aggressive (daily), then the amount of disk storage needed on the log server can be somewhat reduced. For example, the log server in this paper will have an 18 Gigabyte hard drive with 10 Gigabytes devoted to the /var partition. This is the partition that Solaris inherently logs to out of the box, and we will create and use the /var/log directory to store our logs. We will also use the default log file rotation scheme used by Solaris. This rotation scheme is described in a later section.

Another general rule to follow is that the log server should be a high performance machine that can be easily upgraded. Again, the performance level of the log server is going to be dictated by the amount of logging that is being done, and it is always better to be safe than sorry. Also, remember that networks can change. If the log server that is chosen now will have to operate near peak performance just to handle the current logging clients, what will happen when a new logging client or application is added to the network in the future? For our example, we will use a Sun Enterprise 220R with two 450MHz CPUs and 2 GB of memory. I know you are thinking this is overkill now, but think of the comfort you'll feel when management tells you that the network is going to be expanded by adding 10 new machines, each of which will be running BSM.

B. Configuration

Before we dive into the step-by-step process of configuring the log server, I would like to say a word or two about hardening it. The level to which you harden the log server is really dependent on your environment and situation. Obviously, if you are planning on placing your log server in the DMZ of your network for some ridiculous reason, you are going to want to lock it down as tight as possible. On the other hand, if you are planning on placing it in the back end or internal part of your network, you can probably be a little more liberal when hardening it. However, it is always good practice to shut down all the services on the machine that you know you are not going to be using. The actual process of hardening the Solaris operating system is beyond the scope of this paper, but there are many great tools, such as YASSP, and resources on the web to assist in this area. For our example, we will assume that the log server is located on the backend or internal part of the network, and that all of the logging clients are also located on the internal part of the network. We will also assume that the sole purpose of the log server is just that. The log server is not performing any other services or duties for the network.

Now that we have selected a machine for the central log server and placed it in the network, it's time to take a look at some of the steps involved with the configuration process. First we need to assign the log server its identity. To do this, the /etc/hosts file must be edited to associate the label "loghost" with the IP address assigned to the log server. A line similar to the following should be found in the /etc/hosts file to accomplish this task:

192.168.228.1 log-server loghost

Here we have the IP address assigned to the log server, which has been given the hostname “log-server”, and then we assigned this IP address and hostname to be the “loghost”. The keyword “loghost” should only be assigned to one IP address in the /etc/hosts file of any Solaris machine. If you see it assigned to any other address in the /etc/hosts file, remove it. Make sure to save this file before you close it. Next, we need to take a look at the /etc/syslog.conf file. Open this file in some text editor, and add the following lines to the end of the file:

```
auth.debug              /var/log/authlog
daemon.debug                    /var/log/daemonlog
lpr.debug               /var/log/lprlog
news.debug              /var/log/newslog
uucp.debug              /var/log/uucplog
user.debug              /var/log/userlog
kern.debug              /var/log/kernlog
mail.debug              /var/log/maillog
cron.debug              /var/log/cronlog
local0.debug            /var/log/ldap_logs
local1.debug            /var/log/web_server_logs
local2.debug            /var/log/bsm_logs
```

When adding these lines, make sure to use tabs not spaces to separate the fields. The above lines allow us to separate the various logs received by the log server into the separate files listed on the right above based upon the facility that is associated with the log message. To gain an understanding of what the above lines are doing, let’s take a look at the local0.debug. This line instructs the syslog daemon to place any syslog message logged using the local0 facility in the /var/log/ldap_logs file. Every syslog message that is generated has a facility.level pair associated with it [9]. This facility.level pair gives a brief insight as to what entity generated the message, and it gives a means by which to separate or group these messages. To see a complete listing of all facilities and levels that can be used to log messages, and to gain a better understanding of what they mean, refer to the syslog.conf man page [9].

I would like to discuss the three lines above that are using the local0-local2 facilities. Solaris provides 8 local facilities, local0-local7, for use, and none of them are inherently used by Solaris. As you can see above, we are using the first three local facilities to separate our COTS and BSM logs. However, caution must be taken when using these facilities because there are a few products that do use them. For instance, many Cisco devices, such as their routers and switches, inherently log messages using the local7 facility. Now that the syslog.conf file has been properly edited, save and close it. Before the syslog daemon is notified of the changes, the files that are going to be used by the above lines need to be touched. If the /var/log directory does not exist, then create it with

the “mkdir /var/log” command. Now execute a “chmod 600” on the /var/log directory to allow read/write privileges for the root user only. Cd to the /var/log directory and touch each of the following files: authlog, daemonlog, lprlog, newslog, uucplog, userlog, kernlog, maillog, cronlog, ldap_logs, web_server_logs, and bsm_logs. The syslog daemon must now be restarted for the changes in the syslog.conf file to take place. Execute the following command to do this (the # represents the command prompt):

kill –HUP `cat /etc/syslog.pid`

We can test to make sure that the proper changes have taken place by executing the following commands:

logger –p auth.debug TEST
logger –p daemon.debug TEST
logger –p lpr.debug TEST
logger –p news.debug TEST
logger –p uucp.debug TEST
logger –p user.debug TEST
logger –p kern.debug TEST
logger –p mail.debug TEST
logger –p cron.debug TEST
logger –p local0.debug TEST
logger –p local1.debug TEST
logger –p local2.debug TEST

There should now be a syslog message containing the TEST string in each of the above files that we just touched located in the /var/log directory . Our central log server is now ready to receive the various types of log messages that the logging clients will be generating. There is one more issue that needs to be covered before we leave the log server, and this is the rotation of the log files. In order to keep the log files on the log server manageable, they need to be rotated on a regular basis. The common rotation scheme used by Solaris for the /var/adm/messages log file is to rotate it every night at 3:10am. Four archived messages files are kept on the machine at any given time. This means that for the “messages” log file located in the /var/adm directory, there is also a “messages.0”, “messages.1”, “messages.2”, and “messages.3” log file where the greater the number, the older the file. Every night at 3:10am, “messages.2” gets moved to “messages.3”, “messages.1” gets moved “messages.2”, “messages.0” gets moved to “messages.1”, and “messages” gets moved “messages.0”. A new “messages” file is then created to receive the current logs. As you can see, the logs located in the “message.3” file get overwritten, and thus are lost forever. In order to prevent this, each night before 3:10am, “messages.3” needs to be written to a write-once medium for storing.

The length of time that these log files are kept should be dictated by your security policy. The script that does this rotation is /usr/lib/newsyslog. Since this script is already executing every night and rotating the /var/adm/messages file, we can modify it to rotate

the new log files in a similar fashion. We will only keep two archived files of each log file instance instead of four as used in the "messages" case. Open /usr/lib/newsyslog in a text editor, and add the following lines directly before the "kill –HUP `cat /etc/syslog.pid`" line:

```
cd /var/log
LOG=ldap_logs
test –f $LOG.0 && mv $LOG.0 $LOG.1
mv $LOG   $LOG.0
cp /dev/null $LOG
chmod 600    $LOG

LOG=web_server_logs
test –f $LOG.0 && mv $LOG.0 $LOG.1
mv $LOG   $LOG.0
cp /dev/null $LOG
chmod 600    $LOG

LOG=bsm_logs
test –f $LOG.0 && mv $LOG.0 $LOG.1
mv $LOG   $LOG.0
cp /dev/null $LOG
chmod 600    $LOG

LOG=authlog
test –f $LOG.0 && mv $LOG.0 $LOG.1
mv $LOG   $LOG.0
cp /dev/null $LOG
chmod 600    $LOG

LOG=daemonlog
test –f $LOG.0 && mv $LOG.0 $LOG.1
mv $LOG   $LOG.0
cp /dev/null $LOG
chmod 600    $LOG

LOG=lprlog
test –f $LOG.0 && mv $LOG.0 $LOG.1
mv $LOG   $LOG.0
cp /dev/null $LOG
chmod 600    $LOG

LOG=newslog
test –f $LOG.0 && mv $LOG.0 $LOG.1
mv $LOG   $LOG.0
```

```
cp /dev/null $LOG
chmod 600    $LOG

LOG=uucplog
test –f $LOG.0 && mv $LOG.0 $LOG.1
mv $LOG   $LOG.0
cp /dev/null $LOG
chmod 600    $LOG

LOG=userlog
test –f $LOG.0 && mv $LOG.0 $LOG.1
mv $LOG   $LOG.0
cp /dev/null $LOG
chmod 600    $LOG

LOG=kernlog
test –f $LOG.0 && mv $LOG.0 $LOG.1
mv $LOG   $LOG.0
cp /dev/null $LOG
chmod 600    $LOG

LOG=maillog
test –f $LOG.0 && mv $LOG.0 $LOG.1
mv $LOG   $LOG.0
cp /dev/null $LOG
chmod 600    $LOG

LOG=cronlog
test –f $LOG.0 && mv $LOG.0 $LOG.1
mv $LOG   $LOG.0
cp /dev/null $LOG
chmod 600    $LOG
```

Make sure to save and close the file.

The log server should now be configured to rotate each of the log files that we created every night at 3:10am. Remember that the "*.1" archives of each log file will be overwritten each night, so they will all need to be written to a write-once medium before rotation. Also, notice that we chmod 600 each current log file during the rotation process, this gives read/write privileges to root only. This ensures that only those with root privileges can read or modify the log files.

IV. Logging Clients

In this section we will examine the steps needed to configure each logging client to send the proper log messages to the central log server. We will start with instructions that apply to all of the logging clients, and then we will dive into the specialized instructions on a per client basis.

A. Configuration for all clients

The first thing that we need to do on each and every logging client is edit the /etc/hosts file in order to allow the client to be able to identify the central log server. This can be accomplished by adding the following line to the file:

192.168.228.1 log-server loghost

Remember that this is the IP address and hostname that we previously assigned to the log server. Once again make sure that the "loghost" string does not appear anywhere else in the hosts file. Make sure to save and close the file. The next thing we need to do is add the following line to the top of the /etc/syslog.conf file right below the initial commented section [6]:

*.debug @loghost

Remember to separate the fields using tabs only. The above line instructs the syslog daemon to send every syslog message to the central log server no matter what facility.level pair is used. The syslog daemon now needs to be restarted in order for the above changes to take place. Execute the following command to do this:

kill –HUP `cat /etc/syslog.pid`

To test each logging client to make sure that the changes have taken place, execute the following command on each logging client:

logger –p daemon.notice TEST

This command will generate a syslog message using the "daemon" facility and "notice" severity level. According to the syslog.conf file on each logging client, this message should not only be written to the local /var/adm/messages file, but it should also be forwarded to the central log server. To verify this, go to the log server and examine the /var/log/daemonlog file. In this file we should see a syslog message containing the TEST string from each of the IP addresses or hostnames of our logging clients (depending on whether there is an entry in the /etc/hosts file associating the client's IP address with a hostname). If there is, then we know that each of the logging clients is successfully logging to the log server.

The next thing that we need to do on each logging client is activate BSM. BSM is a kernel

level auditing process that is packaged with Solaris, but is not inherently activated. It qualifies as a C2 security level auditing feature as defined in the Trusted Computer System Evaluation Criteria (TCSEC) [1]. It can be an extremely useful tool for host-based intrusion detection and analysis, but if not configured properly, it could quite possibly perform a DOS attack on your central log server, and cause many syslog messages to be dropped. The intricacies of configuring BSM are beyond the scope of this paper, but there are some very useful links listed at the end of this paper that can help greatly. For the sake of our example logging system, we will go through a very brief configuration example. We will start by editing the /etc/security/audit_control file. This file acts as the main configuration file for the BSM audit daemon. Open this file in a text editor and modify the following lines as instructed below:

minfree:20
flags:
naflags:

Change the above lines to the following:

minfree:50
flags:lo,ex
naflags:lo

Save and close the file. In the flags line you can see that we added the "lo" and "ex" classes. The "lo" class instructs the audit daemon to record the "login" events for all users, and the "ex" class instructs the audit daemon to log every exec system call made by every user (basically log every command executed by every user) [2]. The naflags field specifies events that are not mapped to a particular user [2]. We will discuss the minfree field below. The other change that we need to make is to the /etc/security/audit_startup file. Open this file with a text editor, and add the following line to the end:

auditconfig –setpolicy +argv

Save and close the file. The audit_startup file is also read by the BSM audit daemon upon startup, and the above line instructs the daemon to record all command line arguments associated with any give exec system call. We are now ready to actually start the BSM audit daemon. Execute the commands below to do this [1]:

init s

NOTE: The above command instructs Solaris to enter single user mode. Do not enter this command if you are logged into the client machine remotely. You will be disconnected.

```
# /etc/security/bsmconv
# init 6
```

NOTE: The above "init 6" command instructs the Solaris machine to reboot.

After the machine has finished rebooting, the BSM audit daemon should be running. To verify this execute the following command:

```
# ps –ef | grep auditd
```

The auditd process should be displayed. Further verification can be made by taking a look at the /var/audit directory. Cd to this directory, and perform an "ls" command on it. A file should be listed that looks similar to the filename below.

20020328101545.not_terminated.logging_client1

The fields in this filename are as follows:

start-timestamp.not_terminated.*machine-name*

This is the file that the BSM audit daemon writes to. This file is written in binary format, and can only be viewed with the "praudit" command. Unfortunately, the BSM audit daemon does not have built in syslog functionality. However, I have attached a simple perl script to the end of this paper that will generate syslog messages containing the output of this file. The script runs as a daemon that tails the audit file and pipes this tail output into the "praudit –l" command. The "-l" flag tells the "praudit" command to output the audit record in a single line. The script then takes the output of the "praudit" command, which is plain text, and sends it to the log server using the perl Net::Syslog module, which can be found at www.cpan.org. This perl module is very nice because it not only bypasses the local syslog daemon on the Solaris logging client, but it also allows you to specify directly the IP address that you want to send the syslog message to [5]. See the Net::Syslog man page for more information on the options made available by this module [5].

This may sound like a somewhat complicated description, but the code is very simple to look at and understand. The name of the script is BSM_daemon.pl. To ensure that this perl daemon starts upon boot up of the machine, we need to add a script, which we will call S99audit_start, to /etc/rc3.d directory. First place BSM_daemon.pl in the /etc/init.d directory. Now create a file called S99audit_start, and type the following lines in it:

```
#! /bin/sh

sleep 5
/etc/init.d/BSM_daemon.pl
```

Save and close the file. Place the file in the /etc/rc3.d directory, and then execute a "chmod 700" on the file. This should start the perl daemon upon boot up of the machine. The reason that we are using a startup script rather than a symbolic link is scalability. We are going to add a command to this startup script later in this paper to start another perl daemon. We should also ensure that the perl daemon is killed gracefully any time that the machine is powered down. To do this, create a file called K99audit_stop, and add the following lines to it:

```
#! /bin/sh

kill –TERM `cat /etc/BSM_daemon.pid`
```

Save and close the file. Place the file in the /etc/rc0.d directory, and then execute a "chmod 700" on the file. This will now gracefully kill the perl daemon any time that the machine is powered off.

There is one more thing that needs to be discussed concerning BSM, and that is the rotation of the BSM log files. These files can become very large, and thus need to be periodically rotated and removed to prevent the filling up of the /var/partition. To aid us in accomplishing this, we will employ the use of the /etc/security/audit_warn script. Recall the "minfree:" line from the audit_control file. Remember that we changed the value from 20 to 50. This field specifies the percentage of free disk storage needed in the file system containing the current audit file. When this threshold is reached /etc/security/audit_warn script is invoked. So when the file system reaches a point when only 50 percent of its allocated disk storage on the /var partition is free, the audit_warn script will be executed. We are going to modify this script to rotate and delete the current BSM audit file when this happens. To do this, add the following lines to the top of the script before the "# Check usage" comment:

```
audit –n
kill –TERM `cat /etc/BSM_daemon.pid`
/etc/init.d/BSM_daemon.pl
```

Save and close the file. The audit –n command instructs the audit daemon to close and timestamp the current audit file, and create a new one. The kill command is for the daemon that was mentioned above, and the line below that restarts the daemon. The daemon has to be killed and restarted in order to tail the new audit file. When the daemon is restarted it also deletes any old audit files. We have now taken care of our rotation and removal problem.

B. Configuration for Specialized Logging Clients

Most COTS products on the market today provide some type of logging. Usually, these

products will create a file or two in some obscure directory, and then will begin writing plain text messages to them. Most of them do not offer the ability to forward these log messages to a remote location. This means that if we are going to use a central logging system, then we have to find a way to get these log messages to our central log server. Two such products that fit this mold are the iPlanet Web Server and the iPlanet Directory Server (LDAP). Both products behave in similar manner with respect to logging. When installed both products support multiple instances of each type of server. The log files for each instance of each type of server can be found in the product's <server-root-directory>/<instance>/logs directory. Below is an example:

/products/iws-6.0/http-level1/logs/access

The "iws-6.0" stands for iPlanet Web Server version 6.0. Looking at the above example, the server root directory is "/products/iws-6.0", and the instance is "http-level1". Under this directory is /logs directory, and "access" is the name of the log file [3][4]. Both of these products also generate an "errors" log file in the same directory [3][4]. I have attached a daemonized perl script named Web_Server_daemon.pl that tails both the "access" and "errors" log file for a given instance of the web server and generates syslog messages containing messages written to either of these files. Again, the perl Net::Syslog module is used to generate and send each syslog message. This perl daemon can be killed at any time with the following command:

kill –TERM `cat /etc/Web_Server_daemon.pid`

It can easily be modified to work for a given instance of LDAP server, and it provides a template for any other type of COTS product for which you might want to centralize logs. Both of the previously mentioned iPlanet products provide a rotation mechanism for their respective log files. Whenever this rotation mechanism is used, the corresponding perl daemon must be killed and restarted so that it monitors the current log file.

We now need to ensure that the previously mentioned perl daemon is start upon boot up of the machine that it is located on. First place Web_Server_daemon.pl in the /etc/init.d directory. Now recall the S99audit_start script that we created earlier for the BSM_daemon.pl perl daemon. This script should be located on all logging clients since we are running BSM on every one of them. Open the S99audit_start script, which is located in the /etc/rc3.d directory, and add the following line to the end of it:

/etc/init.d/Web_Server_daemon.pl

Save and close the file. This should ensure that our perl daemon is executed upon boot up of the machine. We also need to make sure that the perl daemon is killed gracefully any time that the machine is powered down. To do this, we need to add the following line to the end of the /etc/rc0.d/K99audit_stop script:

kill –TERM `cat /etc/Web_Server_daemon.pid`

Save and close the file. This will kill the perl daemon gracefully any time that the machine is powered down.

V. Conclusion

We have now successfully set up a central logging system incorporating all Solaris operating system logs, BSM logs, and logs from a couple of COTS products. This should make the administrator's job of analyzing the logs much simpler due to the fact that all of the various types logs are now consolidated in one place. The daemonized perl scripts that have been attached to this paper have been run under perl5.005_03.

Resources

1. Osser, William; Noordergraaf, Alex. "Auditing in the Solaris 8 Operating Environment" February 2001. URL: http://www.sun.com/blueprints/0201/audit_config.pdf

2. Sun Microsystems, Inc. "SunSHIELD Basic Security Module Guide" 2000. URL: http://docs.sun.com/ab2/coll.47.11/SHIELD/

3. Sun Microsystems, Inc.; Netscape Communications Corporation. "iPlanet Directory Server Administrator's Guide version 5.0" April 2001. URL: http://docs.iplanet.com/docs/manuals/directory/50/pdf/ag/admin.pdf

4. Sun Microsystems, Inc.; Netscape Communications Corporation. "iPlanet Web Server, Enterprise Edition Administrator's Guide version 6.0" May 2001. URL: http://docs.iplanet.com/docs/manuals/enterprise/50/ag/es60.pdf

5. Howard, Les. "Net::Syslog" man page. March 2000. URL: http://search.cpan.org/doc/LHOWARD/Net-Syslog-0.03/Syslog.pm

6. Carnegie Mellon University. "Configuring and using syslogd to collect logging messages on systems running Solaris 2.x" 29 Jan 2001 last revision. URL: http://www.cert.org/security-improvement/implementations/i041.08.html

7. Wall, Larry; Christiansen, Tom; Orwant, Jon. Programming Perl. O'Reilly & Associates, Inc. 3rd Edition July 2000

8. Pitts, Donald. "Basic Security Module: Solaris Kernel-based Auditing" 23 May 2001. SANS IDS GIAC Practical.

9. Sun Microsystems, Inc. "Solaris 8 Reference Manual Collection" - syslog.conf man page. 22 Jan 1997. URL: http://docs.sun.com/ab2/coll.40.6/

Web_Server_daemon.pl

```
#!/bin/perl

#####################################################
#
#  Author:  Kent Stout
#  Title:   Web_Server_daemon.pl
#  Funtion: This is a daemonized script that tails
#           the access and errors log files generated
#           by the given web server.  Using Net::Syslog,
#           this script forwards any log messages
#           written to the afore mentioned files
#           to the log server to be processed.
#
#####################################################

my $pid = fork;
exit if $pid;
die "Couldn't fork: $!" unless defined($pid);

use POSIX;
use Net::Syslog;
use IO::Select;

POSIX::setsid() or die "Can't start a new session: $!";

my $time_to_die = 0;
my $grpid = getpgrp;
open (PID, ">Web_Server_daemon.pid") ||
   die "ERROR: Couldn't open Web_Server_daemon.pid file: $!";
print PID "$grpid";
close(PID);

sub signal_handler {
   $time_to_die = 1;
   kill 9 => -$grpid;
}

sub system_error_handler {
   my($code, $command_string) = @_;
   $code = $code >> 8;
   print "System call Error for $command_string: $code\n";
   exit;
}

$SIG{INT} = $SIG{TERM} = \&signal_handler;
$SIG{PIPE} = 'IGNORE';

until ($time_to_die) {
   my $web_message = new Net::Syslog(Facility => 'local1', Priority
=> 'debug', SyslogHost => 'loghost');
   my $web_access_file = "/products/iws-6.0/http-level1/logs/access";
   my $web_errors_file = "/products/iws-6.0/http-level2/logs/errors";

   open(WEB_ACC_OUT, "tail -f $web_access_file |") ||
      die "ERROR: Couldn't open tail -f $web_access_file: $!";
   open(WEB_ERR_OUT, "tail -f $web_errors_file |") ||
      die "ERROR: Couldn't open tail -f $ldap_access_file: $!";
```

```
   my $sel = new IO::Select(\*WEB_ACC_OUT, \*WEB_ERR_OUT);
   my $fh = "";
   my @ready = ();
   while(@ready = $sel->can_read) {
      foreach $fh (@ready) {
         my $line = <$fh>;
         if ($fh == \*WEB_ACC_OUT) {
            $web_message->send("Access: ".$line);
         }
         else {
            $web_message->send("Errors: ".$line);
         }
      }
   }
}
```

BSM_daemon.pl

```
#!/bin/perl

#####################################################
#
#  Author:  Kent Stout
#  Title:   BSM_daemon.pl
#  Funtion: This is a daemonized script that tails
#           the current bsm log output file, which is
#           signified with the not_terminated tag,
#           in the /var/audit/ directory.
#           Using Net::Syslog, this script forwards
#           any log message written to the above file
#           to the log server so that the messages
#           can be properly processed.
#
#####################################################

my $pid = fork;
exit if $pid;
die "Couldn't fork: $!" unless defined($pid);

use POSIX;
use Net::Syslog;

POSIX::setsid() or die "Can't start a new session: $!";

my $time_to_die = 0;
my $grpid = getpgrp;
open (PID, ">BSM_daemon.pid") ||
   die "ERROR: Couldn't open BSM_daemon.pid file: $!";
print PID "$grpid";
close(PID);

sub signal_handler {
   $time_to_die = 1;
   kill 9 => -$grpid;
}
```

```
sub system_error_handler {
   my($code, $command_string) = @_;
   $code = $code >> 8;
   print "System call Error for $command_string: $code\n";
   exit;
}

$SIG{INT} = $SIG{TERM} = \&signal_handler;
$SIG{PIPE} = 'IGNORE';

until ($time_to_die) {
   opendir(DIR, "/var/audit/") ||
      die "ERROR: Couldn't open /var/audit/ directory: $!";
   my @files = readdir DIR;
   closedir DIR;
   my $bsm_file = "";
   foreach $file (@files) {
      if ($file =~ /not_terminated/) {
         $bsm_file = "/var/audit/$file";
      }
      elsif ($file !~ /^\./){
         system("rm /var/audit/$file") == 0
            || &system_error_handler($?, "rm /var/audit/$file");
      }
   }

   my $bsm_message = new Net::Syslog(Facility =>
'local2',Priority=>'debug', SyslogHost => 'loghost');

   open BSM_OUT, "tail -f $bsm_file | praudit |" ||
      die "ERROR: Couldn't open tail -f $bsm_file: $!";
   while(<BSM_OUT>) {
      $bsm_message->send($_);
   }
}
```