Global Information Assurance Certification Paper

A Centralized Logging and Alerting Solution Using Logwatcher/Logcatcher and Swatch

SANS GIAC Security Essential Certification
Practical Assignment
Version 1.4b – Option 2
06 July 2004

Daniel J. Wittig

Abstract

You have just completed a long project and spent a lot of time and money implementing reasonable security controls to appropriately protect the company GIAC Enterprise Resource Planning (ERP) system. You have spent your time in the trenches and have gone through the complete development process from initial concept through design and construction to user acceptance testing and then finally implementation and rollout. Now you start your next phase of the project, but it turns out that the project was over budget and all outside development or purchases are on hold until first quarter of the next fiscal year. To make matters worse, change management is non-existent at best. This paints a pretty grim picture for the Security Engineer responsible to design, develop and implement a logging and alerting solution for the GIAC ERP System that was just rolled out. This is not an uncommon scenario for many security departments. Many Project Managers seem to overlook the ongoing management of the system and focus all attention on rolling out the solution to the business owners and end users on time.

Basically, your Security Director just tasked you to develop a logging and alerting strategy within a couple months, but with a very limited budget and without outside/contractor development resources.

This practical will illustrate our teams approach to implementing a reasonable logging and alerting solution with freely available open source software. We decided to concentrate on two freely available and open source products. The first tool, Logwatcher/Logcatcher, was developed by Luke Kanies and provides ease of log management and improved integrity of the logs through centralized logging. The second tool, Swatch monitoring and alerting tool developed by Todd Atkins provides real time alerting and notification of significant events that may indicate possible abuse or misuse of the system. Both of these functions, centralized logging and alerting, are critical aspects of a layered defensive posture to monitor the current security state of the Company's ERP system. The solution presented is not meant to be a complete solution, but it is a solution that can be implemented quickly and with limited resources. It is also a solution that can be built upon as additional resources, time and money, become available.

1. Pre-existing Scenario

After reviewing the current operating environment it became apparent that the management of the logs and the data generated by system, network, application, and user activities was inadequate at best. System administrators were randomly deleting system logs, application logs were unreliable due to file permissions and administrators were too busy with basic system administrative tasks to review the logs. We performed a risk assessment of the current operating environment to determine the extent of our problems. The following risks were noted as result.

- Application log files are currently set to a permissions level that makes it impossible to guarantee the integrity of the logs and to prevent or detect unauthorized disclosure, deletion or access.

- System administrators are randomly deleting the logs as disk space nears capacity. The deletion of the logs makes subsequent review, analysis and problem resolution impossible. Therefore, it will not be possible to detect actual or attempted inappropriate access for inquiry or modification of application data or system configuration.

- No procedures/processes exist to review system-created audit logs for possible system compromise. Unauthorized abuse or misuse could go undetected or unaccounted because administrators or security personnel were not routinely reviewing the logs.

- The security team was not being notified of system events or changes to configuration that adversely impact the security state of the Company's ERP System. The following are an example of the events or changes:

    - Application security disabled without security team approval.
    - New security programs migrated to production environment before security/code review with without security team approval.
    - Unauthorized file permission changes.
    - Changes to security configuration outside normal change control process.
    - Unauthorized migration to production of security authorization changes.

Logging is only one aspect of a layered defensive posture, which begins with the establishment of appropriate and effective security policies, but it's probably the most overlooked.

2. The Plan

Now that we understand the risks, our next task was to attack the problem and develop a solution that we could implement within two months, without a budget and limited developer resources. We met as a team to discuss the current scenario and decided to focus on journaling the logs to a remote server and automating the review and alerting process. Furthermore, Because of the limited budget and resources, we knew we knew that we had to look to freely available open source software.

3. Our Objectives

The next step was to set the objectives of what we wanted to accomplish. We came up to two main objectives.

- The integrity, trust and supportability of the system and application logs will be greatly enhanced if logs are journaled to a secure server interactively. The journaling of the log files to a secure server interactively will protect the log files from being accessed, modified, or deleted by authorized or unauthorized users. Furthermore, remote journaling will help mitigate disk space issues that may occur during heavy utilization.

- Active real time monitoring tools should be configured to actively monitor system logs to detect suspicious or abnormal activity or events while they are occurring. Teams with key roles in responding to events need to be notified whenever there are indications that something suspicious, unusual or abnormal is occurring or has occurred. Teams within the organization cannot execute their responsibilities if they are not notified in a timely manner that an event is occurring or has occurred. As a result, the ERP System and the data may suffer greater damage (loss of confidentiality, integrity, availability) than if all those who needed to be involved had been informed in a timely manner.

4. The Approach

The steps we took to implement to solution are divided into three tasks. The first task was to identify the type of data or the logs we want to collect. The next step was to install and configuration of logwatcher and logcatcher to journal the collected data to the secure central logging server. The final step was to configure Swatch to automatically notify the appropriate teams upon detection of matched patterns or configured events.

4.1 Identify the Data to Collect

Before we discuss the technical details of tool we will use to collect and journal the logs, we need to identify the data we want to collect. The main issue with collection of data is that it is not too difficult to enable logging mechanisms to collect the data, but rather that it's altogether too easy to collect an overwhelming amount of it. It is commonly acknowledged that it is important to log as much information as possible because it makes it more difficult or an attacker to hide all of the evidence, but the collection of a large amount of data places considerable strain on processing and storage facilities. Furthermore, a considerable amount of time must be spent, either manually, or aided by tools, sifting through the logs to detect suspicious or abnormal events. Therefore, It is a balancing act between logging too much, and not being able to manage the collected data, and logging too little to be able to ascertain whether indeed an abuse or misuse of the system occurred. Our decision on how much data to collect on the remote server

was not based on the time spent sifting through the logs, but the fact we had limited disk space to actually store the logs as well as shortage of manpower resources to manage the data.

The following table identifies the type of data collected in the location of the information i.e. log files. The data in the logs identified in the following table is the data required to determine abnormal or out of tolerance conditions or system events that have occurred. We decided against journaling binary log files because the issues of converting to plain text. However, there are tools available to convert binary logs, but our focus for this project was text files only.

Types of Data	Location
The use of the switch user (SU) command	/var/adm/sulog
Users last login information	/etc/security/lastlog
Super user (SUDO) logging information. Logs users use of SUDO to run command as root.	/var/log/sudolog
System activity information	/var/log/messages
Apache diagnostic information and records of errors encountered in processing requests	/var/log/http/error_log
User requests processed Apache.	/var/log/http/access_log
Messages from the ERP application server. The following types of data: • Version of the server • Who started or stopped the server, and when • Timestamp error messages • CPU statistics • Processing statistics	$GIACDIR/sytem/latm.log
Messages from the ERP batch server. The following types of data • Version of the job server • Who started the server, and when • Who stopped the server • Parameter settings • Processing statistics	$GIACDIR/sytem/lajs.log
Log changes to the security status (security on or off)	$GIACDIR/system/sec.log
Log the use of dump and load programs which is run, by whom, and when. Also log information about security on creation, for example, when a user creates a new product line, system code, or form ID, and when access to that object is added to the user's security class.	$GIACDIR/system/secadmin.log.
Log failed attempts to access environment categories or forms.	$GIACDIR/system/secvio.log.

Figure 1

4.2 Task 2 – Journal the logs to the remote server.

Collecting and journaling logs on a remote server not only provide ease of log management but also provides for improved integrity of the logs. The integrity and maintenance of the logs is critical to the Security Team's ability to monitor the current operating environment of the ERP Systems. Logwatcher/Logcatcher developed by Luke Kanies is application we used to facilitate the remote collection of the systems logs in a centralized location. The two components of the application are Logwatcher and Logcatcher. The Logwatcher daemon runs on the loghost and "watches" for logging activity and then transfers that captured data to the remote server via Logcatcher. Logcatcher resides on the remote server and "catches" the log data forwarded by the Logwatcher to the remote centralized logging server.

Logwatcher residing on the loghost performs the following functions:

- Watches – Watches the log files specified in the logwatcher configuration file (/usr/local/etc/logwatcher.cfg)
- Collects – Captures the log data
- Connects – Initiates a connection with the remote server
- Transfers – Journals, in real time, the captured data to the remote server

Logcatcher resides on the remote server and performs the following:

- Listens – Listens on port 2000 for connection initiated by logwatcher
- Writes – Writes all logging data transferred to a centralized dump file – /var/log/logcatcher.dump
- Organizes – Writes logging data to /log directory in addition to the dump file, but in a more organized fashion.

A graphical representation of Logwatcher/Logcatcher is shown below.

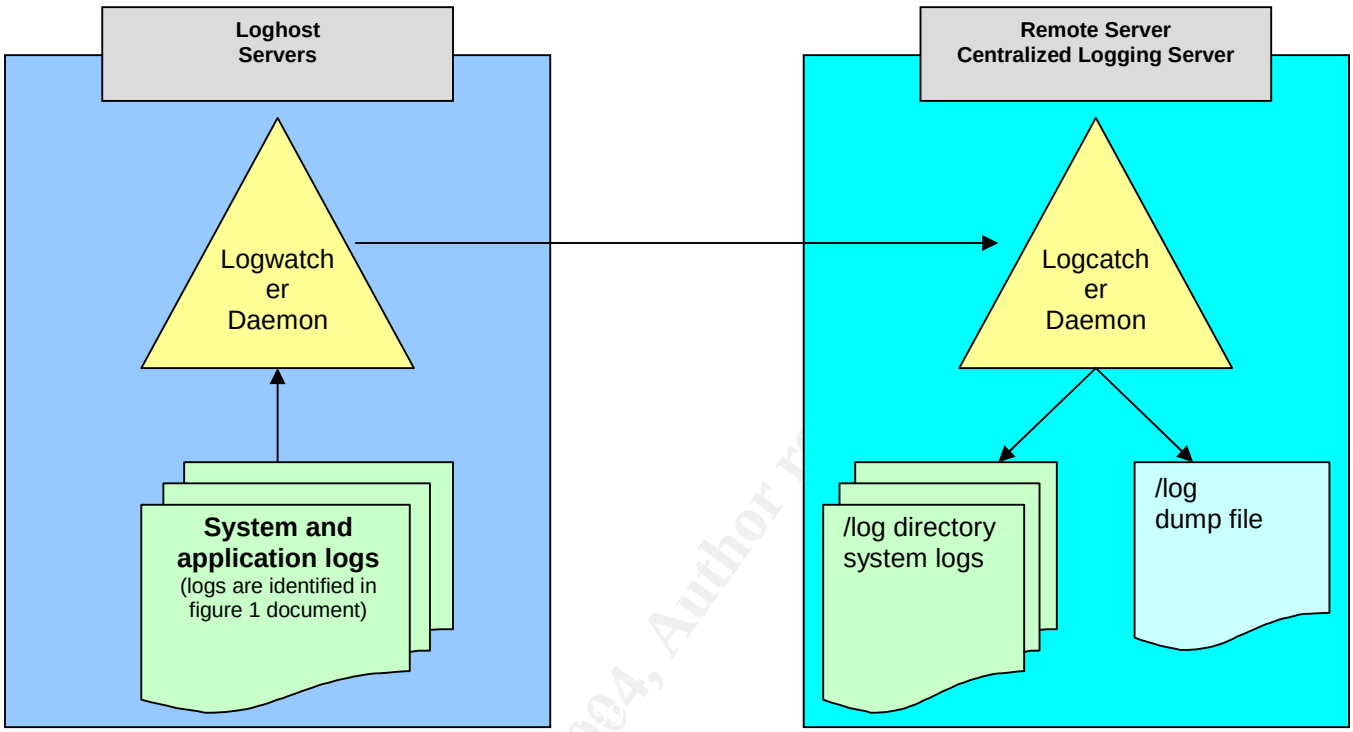


Figure 2

4.2.1 Installation

Our System Administration team completed the installation of Logwatcher/Logcatcher. Therefore, this practical does not discuss that actual installation of the application. The installation consisted of downloading the Logwatcher/Logcatcher program forwarded by the developer and installing it in the right directory. We will discuss the actual configuration of the application, the startup process, and the organization of the data on the remote centralized logging server.

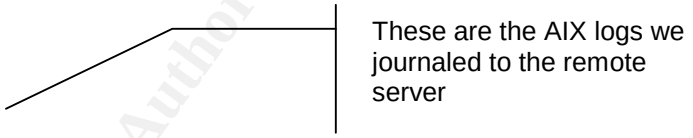
4.2.2 Logwatcher configuration

The logwatcher configuration file is located in /usr/local/etc/logwatcher.cfg. This is where you tell logwatcher what log files to "watch". The log files are listed under the specified group within the configuration file. This is the only file we had to update after the System Administrator completed the installation of Logwatcher/Logcatcher. We added the log files identified in figure 1 of this document to the specified groups. AIX

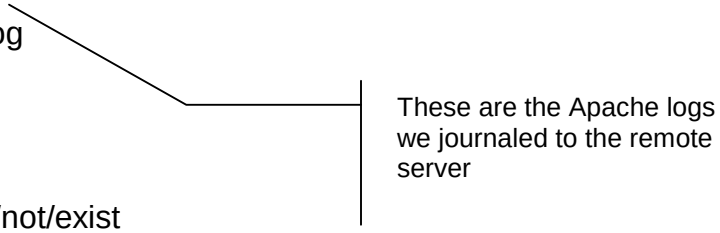
logs where added to the aix group, Apache logs where added to the http group, and ERP logs were added to the env group. Logwatcher is very flexible, because it allows you to make as many groups as you want in the configuration file and then just load them in logwatcher

Logwatcher configuration file

```
# this is just used for defining which groups to parse
# you can find out more about this config file format by
# running 'perldoc Config::IniFiles' and you can find out more
# about the details of the log file itself by running
# 'perldoc /usr/local/scripts/logging/logwatcher'

# $Id: logwatcher.cfg,v 1.5 2004/03/31 19:19:18 wzd4845 Exp $

# all aix logs to watch
# these are opened by adding '--group aix' to the command
[aix]
Logfiles=<<EOT
/var/adm/sulog
/etc/security/failedlogin
/etc/security/lastlog
/var/log/sudolog
/var/log/messages
/var/log/Cfengine
EOT
```

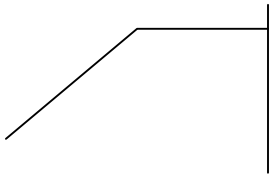



```
# http logs
# these are opened by adding '--group http' to the command
[http]
Logfiles=<<EOT
/var/log/HTTP/error_log
/var/log/HTTP/access_log
EOT

# erp-specific logs
#[erp]
#Logfiles=/this/log/does/not/exist
```




```
# per-environment erp logs
# these logs are opened in every environment specified to logwatcher, and
# to open these you must specify an erp environment with '--env <environment>'
[erpenvlogs]
ERPlogfiles=<<EOT
```

${GIACDIR}/system/latm.log
${ GIACDIR}/system/lajs.log
${ GIACDIR}/system/secadmin.log
${ GIACDIR}/system/secvio.log
${ GIACDIR}/system/sec.log
EOT



4.2.3 Starting Logwatcher

Starting Logwatcher on the loghost is very simple. You just execute the script downloaded by the administrators and specify the groups you want to watch and the server you want to journal the logs to. You specify the groups with --group and the server you want to write to with --server.

> Example /usr/local/scripts/logging/logwatcher --group aix --group http --server my_remote_server
>
> In the example above we are watching the logs in the aix and http groups specified in our configuration file (/usr/local/etc/logwatcher.cfg) and journaling them to the server my_romote_server.

Logwatcher will read the logwatcher configuration file to determine which logs are in the aix and http groups, watch for activity in those logs, and then write the logs entries in real time to my_remote_server.

> Note: Logwatcher cannot write logs to a remote server unless Logcatcher is running on that server. Logwatcher will not be able to initiate the connection if this is the case.

4.2.4 Logwatcher Process

Logwatcher should start one process after the script is executed. Check to make sure the process started correctly by executing the command ps -ef.

> ps –ef|grep logwatcher results:
>
> root 909460 1 3 Jul 03 - 48:36 perl /usr/local/scripts/logging/logwatcher --group aix --group http --server my_remote_server

4.2.5 Logcatcher Configuration

Logcatcher is a simple POE event passer. It listens on one port and passes all events to another port. The only configuration task with Logcatcher was to review of the default

options to determine if they would work in our environment. The following are the default values:

logfile
Where to write the log messages. Defaults to /var/log/logcatcher.log.

ports
The port on which to listen for connections. Defaults to 2000.

stats
How often to print a statistics message to the log file. Defaults to every minute.

store
Where to store the logs. Defaults to /var/tmp/logcatcher.

After a review of the default configuration, we decided to change only the storage location of the logs.

We changed the following in /usr/local/scripts/logcatcher

- Storage location changed to /log
  - $store ||= '/log';

We kept the defaults for the following in /usr/local/scripts/logcatcher

- $port ||= 2000;
- $stats ||= 5;
- $logfile ||= "/var/log/logcatcher.log";

4.2.6 Starting Logcatcher

One of the major selling points of Logcatcher was the option to “dump” all data received from Logwatcher to a central file. Also, if you remove the file Logcatcher is writing to, it will immediately reopen the file. We saw an opportunity with Swatch to monitor the dump file containing all logging data from all hosts writing to the remote logging server. We discuss this further down in the practical.

Start Logcatcher by executing the script downloaded by the administrators and specify the location of the “dump” file. You specify the “dump” file with a --dump.

Example /usr/local/scripts/logging/logcatcher --dump /log/logcatcher.dump

In the example above, Logwatcher will "catch" all data forwarded by Logwatcher and write it to two locations

1. It will write the data to the default store location – /log
2. It will also write the data to wherever you specified with the --dump variable – /log/logcatcher.dump

Caution – Keep in mind that all data is being written to the "dump" file. Make sure you have the storage capacity or disk space to support this. This is critical in our solution because Swatch monitors the "dump" file and Logcatcher will fail to write to the file if it reaches space capacity. What does this mean? No alerts!

4.2.7 Logcatcher Process

Logcatcher should start one process after the script is executed. Check to make sure the process started correctly by executing the command ps -ef.

ps –ef|grep logcatcher should indicate:

root 6184986 1 3 00:03:13 - 2:22 perl -w /usr/local/scripts/logging/logcatcher --dump /log/logcatcher.dump

4.2.8 Log storage and organization

Logcatcher not only writes to the dump file, but also writes to the location specified in "store" variable of the Logcatcher script, but in a more organized fashion. A subdirectory is created for each loghost journaling logs to the remote centralized logging server.

The following is an example of the directory structure of the directory specified in the "store" option of Logcatcher program.

- /log – all log data is stored in /log
- /log/loghost – each loghost writing to the remote centralized server has a directory
- /log/loghost/log_name – every individual log for the loghost has a directory (example - /var/log/messages from myloghost)
- /log/loghost/log_name/year – Logs are organized by year
- /log/loghost/logname/year/month – logs are further organized by month
- /log/loghost/logname/year/month/day – final location of the actual data is the day

Example – You are looking for the sudo logs from June 17th, 2004 logs on remote centralized logging server from server na123abc

The path on the remote centralized logging server would be - /log/na123abc/ var__log__sudolog/2004/6/17

- UNIX command - cd /log/na123abc/ var__log__sudolog/2004/6/17
- UNIX command - more 17

We are not monitoring these files with Swatch, but they have been very useful for problem resolution.

4.3 Automated monitoring, detection and alerting

The monitoring of the data collected and the detection of certain events was a major concern of the Security Team as well as the System Administrators. Both these teams have the assigned responsibility to act upon notification or detection of certain events, but neither team had the resources to sift through the vast amount of data for signs of suspicious or abnormal activity.

We automated the sifting though that vast amount of data by implementing Swatch on the remote logging centralized logging server. Swatch is a simple program written in the Perl programming language that is designed to monitor log files. Swatch allows us to automatically scan the log files collected on the remote centralized logging server by Logcatcher. Swatch will search for particular entries or patterns and then take appropriate action, such as sending an alert email or alert. Additionally, Swatch is widely used, is relatively simple to implement, and is freely available open source software.

One of the features of Logcatcher is that it was developed with a --dump=<file> function. The means that Logcatcher writes all data collected from Logwatcher to a dump file as well as individual files based on hostname and log. We configured Swatch to monitor the dump file and send alerts based on the configuration file or filter developed by the Security team. By configuring Swatch to monitor the dump file, we search for patterns in all logs from all hosts journaling logs to the Centralized Logging Server.

There are three components of Swatch. The first component is the swatch tool itself written in the Perl programming language. The second component is the startup or command line options developed by the Security team. The startup script for the most part defines the logs to be monitored, the mode of operation and the configuration file containing filter or pattern we are monitoring. The third component is configuration file or

filter developed by the Security team. The configuration file contains the patterns to look for and the actions to perform when a specific pattern is found.

The preceding sections contain the detailed steps of installing and/or configuring the three components of swatch.

- The Swatch Perl program and the prerequisite Perl modules
- The Swatch startup script
- The Swatch filter or configuration file

4.3.1 Download and Install Prerequisite Perl modules

Repeat the following steps for each perl module:

Date::Calc

Time::HiRes

Date::Format

1) download the perl module from www.cpan.org to the local machine
2) scp the perl module from the local machine to your home directory $HOME
3) mv the downloaded gzip file to /usr/lib/per15
4) untar the gzip file by executing tar –xzvf <gzip file>
5) cd into the directory created during untarring/unzipping and type "perl Makefile.PL" – this will create a "Makefile" with the appropriate parameters
6) while in the directory type "Make" – this will compile the module and create the dynamically linkable library file that will be linked to Perl.
7) while in the directory type "make test" – should indicate All tests successful at the end if everything is okay
8) while in the same directory type "make install" – this completes the install of the Perl module.

4.3.2 Download and Install Swatch

1) download swatch from http://swatch.sourceforge.net/ to you local machine
2) scp the downloaded swatch gzip file from your local machine to your home directory
3) mv swatch downloaded swatch gzip file from the home directory to /usr/local/bin

4) untar the downloaded swatch gzip file by executing “tar –xzvf <gzip file>”. This will create the swatch directory /usr/local/bin/swatch-3.0.8

5) cd into /usr/local/bin/swatch-3.0.8 and type “perl Makefile.PL” - this will create a “Makefile” with the appropriate parameters and will also check the system to determine if prerequisite perl modules are installed. The following is returned if all is okay

Screen shot

```
Checking if your kit is complete...
Looks good
Writing Makefile for swatch
```

6) while in /usr/local/bin/swatch-3.0.8 type “make” to compile swatch.

Screen Shot:

```
cp swatch_oldrc2newrc blib/script/swatch_oldrc2newrc
/usr/bin/perl "-MExtUtils::MY" -e "MY->fixin(shift)" blib/script/swatch_oldrc2newrc
cp swatch blib/script/swatch
/usr/bin/perl "-MExtUtils::MY" -e "MY->fixin(shift)" blib/script/swatch
Manifying blib/man1/swatch.1
Manifying blib/man1/swatch_oldrc2newrc.1
```

7) while in /usr/local/bin/swatch-3.0.8 type “make test” – should indicate All tests successful at the end if everything is okay

Screen shot

```
All tests successful.
```

8) while in /usr/local/bin/swatch-3.0.8 type “make install”.

Screen Shot

```
Writing /usr/lib/perl5/site_perl/5.8.0/i386-linux-thread-multi/auto/swatch/.packlist
Appending installation info to /usr/lib/perl5/5.8.0/i386-linux-thread-multi/perllocal.pod
```

9) while in /usr/local/bin/swatch-3.0.8 type “ make realclean” to complete the installation of swatch

Screen Shot

```
rm -f blib/script/swatch_oldrc2newrc blib/script/swatch

rm -rf ./blib Makefile.aperl blib/arch/auto/swatch/extralibs.all
perlmain.c tmon.out mon.out so_locations pm_to_blib *.o *.a
perl.exe perl perl  swatch.bso swatch.def libswatch.def swatch.exp
swatch.x core core.*perl.*.? *perl.core

mv Makefile Makefile.old > /dev/null 2>&1

rm -rf blib/lib/auto/swatch blib/arch/auto/swatch

rm -rf swatch-3.0.8

rm -rf Makefile Makefile.old
```

4.3.3 Swatch startup script

The Swatch Startup Script is the second component of Swatch and contains the command line options such as the configuration file, the tail, input record separators, etc. Basically this is where you instruct swatch what configuration file or filter to use and what file to monitor. The configuration file is the filter of what to look for and what actions should be taken upon notification. Command line options listed below are what we used for this project.

--config-file=”file name” – tells Swatch where to find it’s configuration file. The default is ${HOME}/.swatchrc.

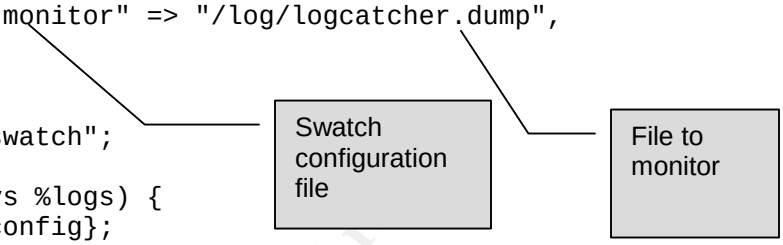
--tail-file=”file name” – tells Swatch which file to tail. Examines each line of text as they are added to the filename.

--input-record-separator=”regular expression” – tells Swatch to use regular expression to delineate the boundary of each input record. The default is a carriage return.

Our script was very simple because we configured Swatch to monitor one file, the Logcatcher dump file on the remote centralized logging server. By monitoring the dump file, we search for patterns in all logs from all hosts journaling logs to the Centralized Logging Server. This was the major selling point of our solution. Instead of having to monitor every log file on multiple hosts, we could just monitor the dump file on the remote centralized logging server. It is also a single point of failure so you must take additional steps to mitigate the risk of the dump file being corrupted or unavailable.

```
#!/usr/local/bin/perl -w

use strict;
```

```
my %logs = (

# This is the swatch start script and will monitor all piped traffic
from the ERP servers(web and app)to the centralized logging server.

"/home/user/swatch/conf/ERP.monitor" => "/log/logcatcher.dump",

);

my $base = "/usr/local/bin/swatch";

foreach my $config (sort keys %logs) {
      my $command = $logs{$config};

      my $cmd = $base . " --config-file=$config" .
            " --tail-file=$command" .
            ">>/tmp/swatchlog 2>>/tmp/swatchlog &";

      system($cmd);
}
wait;
```



4.3.4 Swatch configuration file or filter

The third component of Swatch is the configuration file or filter developed by the Security team. This is the file containing the intelligence of Swatch. The file contains the patterns to look for and the actions to be taken when a pattern is matched.

Before starting on the actual configuration, we needed to develop and document our strategy. This is an important step that should be completed before developing the configuration file. The strategy document became the input or requirements for the configuration file. The hardest part with the strategy document was being able to correlate an actual event with the log entry.

The following are some examples taken from our Swatch strategy document:

Event	Log	Pattern	Priority	Alert
New SETUID root Program	/var/log/messages	NEW SETUID root PROGRAM	2	email to my2way.com
change to file permission - capital IBM file	/var/log/messages	had permissions | IBM	1	email to my2way.com
ERP security loads	secadmin.log	la_apply | secload	3	email to my2way.com
ERP program loads	secadmin.log	pgmload	2	email to my2way.com

The first item of the configuration file is the pattern to look for within the tailed file.

watchfor /"pattern"/

The next items within the configuration file are the actions to be taken when the pattern in matched within the tailed file. We configured Swatch to take the following actions. There are additional actions that can be configured, however, we chose a very simple solution. Our objective was to notify the responsible team of an event so they can take appropriate actions.

- echo [modes] – echo the matched line
- mail [address=address] – send email to address(es) containing the matched lines as they appear
- bell [N] – echo the matched line, and send a bell N times
- throttle hours:minutes:seconds – used to limit the number of times that the matched pattern has actions performed on it

Swatch was configured to monitor (/log/logcatcher.dump), search for the following patterns, and send a notification to the appropriate team when a pattern in matched. These teams will have the responsibility of responding to the alerts sent by Swatch to determine if an actual abuse or attempted abuse of the ERP systems has occurred.

The following Swatch configuration file was used for this solution:

```
# swatch configuration file to monitor aix, http, and ERP logs in the ERP
production environment

# Swatch monitors /log/logcatcher.dump and searches for patterns listed below

 # Configuration for monitoring /var/log/messages.  This is the file that stores
system activity information.

# New SETUID program
# This is a priority 2 type alert.

watchfor   /NEW SETUID root PROGRAM/
        echo normal
        bell
        mail ERP.Security@MyCompany.com,subject=New SETUID root Program
in Production Priority 2 Alert

# File Permission Changes.
# This is a priority 2 type alert
watchfor   /had permission | IBM/
        echo normal
```

```
        bell
        mail ERP.Security@MyCompany.com,subject=File Permission Change in
Production Priority 2

    # swatch configuration file for constant monitoring of the ERP secadmin log.  the
    secadmin log serves two purposes. first it logs every use of the dump and load
    program which is run, by whom and when.  Second, it logs information about
    security on creation for example, when a user creates a new product line, system
    code, or form ID and access to that object being added to a security class.

    # ERP security loads logged
    # this is a priority 3 type alert
    #watchfor       /la_apply|secload/
        echo
        bell
        exec
        throttle 00:10:00
        mail ERP.Security@MyCompany.com,subject=la_apply or secload Priority 3
    Alert

    # program loads
    # this is a priority 2 type alert
    watchfor        /pgmload/
        echo normal
        bell
        mail ERP.Security@hcahealthcare.com,subject=Program Load Priority 2
    Alert

    # swatch configuration for sec.log.  when a user makes changes to the security
    status through laua, lawsec, or secload,  the sec.log file records the changes.
    Also messages are logged to this file in certain exceptional cases, such as when
    there is an error reading the univ.cfg file or when prodsec parameter in univ.cfg is
    set to on so that security can not be disabled, but somebody attempts to turn
    security off
    # security turned off or disabled
    # this is a priority 1 type alert and needs immediate attention
    watchfor        /Security Disabled/
        echo bold, red_h
        bell 3
        mail Team Member@my2way.com,subject=Security Status Priority 1 Alert

    # user attempt to turn off security, but prodsec parameter in univ.cfg restricts
    ability to do so
    # this is a priority 1 type alert and needs immediate attention
```

```
watchfor        /PRODSEC On - Failed to Disable Security/
        echo bold, red_h
        mail ERP.Security@MyCompany.com,subject=Attempt to Disable Security
in Priority 1 Alert

# Unexpected change to SECOFFICER Security Class
# this is a priority 1 type alert and needs immediate attention
watchfor        /SECOFFICER Class Alert/
        echo
        mail ERP.Security@MyCompany.com,subject=Unexpected change to
SECOFFICER Priority 1 Alert
```

4.3.5 Swatch Issues Encountered During Pilot

A word of caution with the configuration file or filter is to start with a limited number of patterns to search for. It is all together too easy to get carried away with the number of patterns to search for and this can make for a very difficult start. We found that it is better to pilot a few patterns to start and work through the issues before expanding. We also had a huge learning curve correlating actual events to log entries. We spent a lot of time analyzing logs to determine which log entry equated to which event. Some of the issues we worked through are illustrated below:

- Swatch failed to read configuration file and kept reverting to default.
    - Solution – Swatch user did not have access to the configuration file so it reverted to the default configuration file. We simply added Swatch user to a group with read access to the file

- Swatch failed to read the tail file or the file we wanted to monitor.
    - Solution – Another permission problem. The Swatch user did not have sufficient access to read the tailed monitoring file. Again, we simply added the Swatch user to a group with read access to the file.

- Swatch returned hundreds of matched patterns for program loads. We discovered that numerous entries were made for each program load.
    - Solution – Add throttle to program load entry in configuration file. Throttle is used to limit the number of times that the matched pattern has actions performed on it

- Multiple Swatch processes running because Swatch did not kill old process before starting a new process.
    - We added “kill process” functionality to our startup script. Old Swatch processes will be killed whenever new process is started.

5. Summary – After solution implementation

The dust has settled and we have been operational with the solution illustrated in this document for a couple months now, but we are already seeing the benefits. Although we found additional items we would like to address, we greatly enhanced the security posture and trust of the ERP System.

The following illustrates the impact of the solution on the overall security state of our ERP System. This has been accomplished with the implementation of this logging and alerting solution using the Logwatcher/Logcatcher application and Swatch monitoring and alerting tool.

- The Logwatcher/Logcatcher application is being used to journal ERP application log files to a remote and secure centralized logging server in real time. File permissions are currently set to a permission level that protects the integrity of the logs and prevents unauthorized disclosure, deletion or access. Furthermore, log file access is restricted to a very few select members of Security and System Administration Teams.

- Log files associated with the ERP System are journaled to and managed on the remote centralized logging server. Logs are journaled in real time so they will continue to exist even if the System Administrator deletes the files on the ERP System. Logs are also available on the Remote Centralized Logging Server for the Security Team to perform review, analysis, and problem resolution.

- The enormous task of sifting though the vast amount of collected data has been automated. Swatch has been configured to monitor all logs journaled to the Remote Centralized Logging Server and send alerts to teams responsible to respond to specified events. The Security Team is now being notified by swatch of system events or changes to configuration that adversely impact the security state of the Company's ERP system.

Continuously Assess and Evaluate Your Monitoring Process

I mentioned in the beginning of this practical that the solution presented is not meant to be a complete solution, but rather a solution that can be implemented quickly and with limited resources. It is also a solution that can be built upon as additional resources, time and money, become available. It is important to continuously assess and re-evaluate your monitoring process. You will find that adjustments or enhancements to the process are necessary as technology, risks, and vulnerabilities change or mature over time. Discovery of additional risk or detection of new vulnerabilities may also be grounds for adjusting the process.

Bibliography

Garfinkel, Simon and Spafford, Gene. Practical UNIX & Internet Security, 2nd Edition. Bonn: O'Reilly & Associates, Inc. 1996

Allen, Julia H. The CERT Guide to System and Network Security Practices. Boston: Addison-Wesley, 2001

Siegert, Andreas. The AIX Survival Guide. Boston: Addison-Wesley, 1996

IBM Redbook. AIX 5L Differences Guide Version 5.2 Edition – Online Version
http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/sg245765.html?Open

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

IBM AIX Manual. System Management Guide: Communications and Networks – Online Version.
http://publibn.boulder.ibm.com/doc_link/en_US/a_doc_lib/aixbman/commadmn/commadmntfrm.htm

.