Purpose

This document describes how to use the Simple Log Watcher (swatchdog) in conjunction with Nagios XI in order to be notified when certain events are noted in the system log.

Target Audience

This document is intended for use by Nagios administrators who wish to be notified asynchronously of events captured in log files.

Introduction to Swatchdog

Just about everything interesting on a POSIX system can be recorded to a log file somewhere. This can range from the mundane (a hit to the web server) to the critical (a failed disk). For the system administrator then, the challenge becomes noticing the log entries they really care about amidst a sea of information that they may want around, but is not what demands their immediate attention. The purpose of log watching applications such as Swatchdog is to keep an eye on the logs for the administrator, and somehow notify them when one of those high-priority events takes place.

Swatchdog is written in Perl, and as you might expect accomplishes this task using Perl's regular expression capabilities. In its configuration file (which defaults to `~/.swatchdogrc`), you will create one or more stanzas telling Swatchdog to first “watch for this pattern”, and if/when that pattern is matched "take this action". For instance, a very simple directive might look like this:

```
watchfor /sudo/
        echo red
```

Here, our regular expression is actually just a literal string - `"sudo"`. The behaviour resulting from this configuration is that any time the word sudo appeared in the log file, the line of the log containing it would be printed to the terminal where Swatchdog is running, in red text. This of course is not really all that useful, as you probably are not staring at the terminal waiting for Swatchdog to output something, and actually are most likely running it within screen or daemonized, but it gives an easy way to test your configuration.

In real usage, you will probably use one of Swatchdog's other modules, especially *exec*, *pipe*, and *mail*. The *exec* action allow the execution of an arbitrary command on the system, which allows you to write your own scripts to run in response to a log match. You can also use variables to pass particular fields from the matched log line or the entire line as arguments to the command. The *pipe* action is very similar to exec, but rather than running a command with optional passed arguments it uses the matched line as standard input piped into the command. The *mail* action allows you to send an e-mail with the matched lines as the body of the message, which could also be passed through an e-mail to SMS gateway to allow Swatchdog to send text messages.

Communicating with Nagios through NSCA

For this tutorial our focus is on getting Swatchdog to communicate with Nagios XI and letting Nagios XI handle notifications. To do that the NSCA add-on will be used, available from:
https://github.com/NagiosEnterprises/nsca

NSCA has two parts, a daemon that resides on your Nagios XI server and a send command `send_nsca` that will need to be executed by Swatchdog. You will need to configure NSCA in Nagios XI before proceeding. By default the `/etc/xinetd.d/nsca` has the `only_from` line configured with `127.0.0.1`, this will need to be updated to include the IP address of the machine that will be running Swatchdog. You will need to set the Decryption Method (this guide is using **XOR**) and define a password (this guide is using **`Str0ngP@sswr0d`**). The following documentation explains how to configure NSCA:
Using And Configuring NSCA With Nagios XI

Editing Files

In many steps of this documentation you will be required to edit files. This documentation will use the vi text editor. When using the vi editor:

- To make changes press **`i`** on the keyboard first to enter insert mode
- Press **`Esc`** to exit insert mode
- When you have finished, save the changes in vi by typing **`:wq`** and press Enter

Install Swatchdog

Establish a terminal session to the server you want to run Swatchdog on and execute the following commands:

```
yum install -y epel-release
yum install -y perl-devel perl-Date-Calc perl-TimeDate perl-Date-Manip perl-File-Tail
cd /tmp
wget https://downloads.sourceforge.net/project/swatch/swatchdog/swatchdog-3.2.4.tar.gz
tar xzf swatchdog-3.2.4.tar.gz
cd swatchdog-3.2.4
perl Makefile.PL
make
make test
make install
make realclean
```

Install NSCA

In your terminal session you just used to install Swatchdog, execute the following commands:

```
yum install -y gcc
cd /tmp
wget https://github.com/NagiosEnterprises/nsca/releases/download/release-2.9.2/nsca-2.9.2.tar.gz
tar xzf nsca-2.9.2.tar.gz
cd nsca-2.9.2
./configure
make all
mkdir -p /usr/local/nagios/bin
mkdir -p /usr/local/nagios/etc
cp src/send_nsca /usr/local/nagios/bin/
cp sample-config/send_nsca.cfg /usr/local/nagios/etc/
```

You will need to set the Decryption Method (this guide is using **XOR**) and define a password (this guide is using **Str0ngP@sswr0**d) in the file `/usr/local/nagios/etc/send_nsca.cfg`. Open the file in the vi editor using the following command:

```
vi /usr/local/nagios/etc/send_nsca.cfg
```

Define the password:

```
password=Str0ngP@ssw0rd
```

The encryption method should already be defined:

```
encryption_method=1
```

Save the file and close.

You can test sending a NSCA check result to the Nagios XI server by executing the following command (it is one long command wrapped over two lines):

```
echo -e "$HOSTNAME\tTest Service\t0\tThis is a test" | /usr/local/nagios/bin/send_nsca
-H 10.25.5.11 -p 5667 -c /usr/local/nagios/etc/send_nsca.cfg
```

In that command, `10.25.5.11` is the IP address of the Nagios XI server.

In Nagios XI navigate to **Admin** > **Monitoring Config** > **Unconfigured Objects** and you should see the Test Service appear in the list.

The following will also indicate that Nagios XI successfully received the NSCA check result:

```
[1487132086] Warning:  Passive check result was received for service 'Test Service'
on host 'tomcat-c7x-x64.box293.local', but the host could not be found!
[1487132086] Error: External command failed -> PROCESS_SERVICE_CHECK_RESULT;tomcat-
c7x-x64.box293.local;Test Service;0;This is a test
[1487132086] External command error: Command failed
```

The "errors" are expected, this is because Nagios XI does not have a service object for the check result received. This just tells us that NSCA is correctly working.

The last step to installing NSCA is to create a bash script that Swatchdog will use to send the logs to Nagios XI with. Create a new file be executing the following command:

```
vi /usr/local/bin/to-nsca.sh
```

Paste the following into the `to-nsca.sh` file:

```
#!/bin/sh
Host="$HOSTNAME"
Service="Log Events"
State="1"
StatusInfo="${*}"
NagiosServer="10.25.5.11"
NSCACommand="/usr/local/nagios/bin/send_nsca -H $NagiosServer -p 5667 -c /usr/local/nagios/etc/send_nsca.cfg"
Message="$Host\t$Service\t$State\t$StatusInfo\n"
/bin/echo -e "$Message" | $NSCACommand
```

Save the file and exit vi.

Make the file executable:

```
chmod +x /usr/local/bin/to-nsca.sh
```

This command will test the script:

```
/usr/local/bin/to-nsca.sh Test ABC 123
```

What the bash script does is send the status output (`Test ABC 123`) to your Nagios server. You should expect to see similar information on the **Unconfigured Objects** page and also in the log file. Using the script means it will be easier to configure Swatchdog configs.

The following variables in the script will need to be changed to your requirements:

- **`NagiosServer="10.25.5.11"`** to match the IP address of your Nagios XI server.
- **`Host="$HOSTNAME"`** is using the hostname of the server. You can change this to anything, for example **`Host="webserver01"`**
- **`Service="Log Events"`** is targetting the service on the Nagios XI server called Log Events. You can change this to anything, for example **`Service="Log Alarms"`**
- **`State="1"`** is sending the check result to Nagios XI as a WARNING state. The available states are:
  - `0` = OK
  - `1` = WARNING
  - `2` = CRITICAL
  - `3` = UNKNOWN

Configure Swatchdog

Now you are ready to configure Swatchdog to send alerts to your Nagios XI server using NSCA. Here is the scenario that will be used to configure Swatchdog.

- The user **anthony** has logged onto the server Swatchdog is installed on
- He uses the command `sudo touch /tmp/important_file.txt`
- Because he used **sudo**, this information is logged in `/var/log/secure` as follows:
  - `Feb 16 10:37:51 tomcat-c7x-x64 sudo: anthony : TTY=pts/0 ; PWD=/root ; USER=root ; COMMAND=/bin/touch /tmp/test.txt`

Based off that scenario, you will create a Swatchdog configuration that will detect when the `sudo` command is used.

When Swatchdog matches a line, it splits that line up into incrementing variables, using white-space as a separator. For example, the variables generated from the line above are:

- `$_[0] = Feb`
- `$_[1] = 16`
- `$_[2] = 10:37:51`
- `$_[3] = tomcat-c7x-x64`
- `$_[4] = sudo:`
- `$_[5] = anthony`
- `$_[6] = :`
- `$_[7] = TTY=pts/0`
- `$_[8] = ;`
- `$_[9] = PWD=/root`
- `$_[10] = ;`

- `$_[11] = USER=root`
- `$_[12] = ;`
- `$_[13] = COMMAND=/bin/touch`
- `$_[14] = /tmp/test.txt`

When Swatchdog is executed, it looks for the file `.swatchdogrc` located in the home folder of the user who executed Swatchdog. In this example the root user is being used, so the location of the file is `"/root/.swatchdogrc"`. Create the file by executing the following command:

```
vi /root/.swatchdogrc
```

Paste the following into the .swatchdogrc file:

```
watchfor /sudo/
      echo red
      exec /usr/local/bin/to-nsca.sh '$_[5] : $_[13] $_[14] $_[15] $_[16] $_[17] $_[18]'
```

Save the file and exit vi.

You will now need to start Swatchdog using the following command:

```
swatchdog -t '/var/log/secure'
```

This will start Swatchdog, it will be running in the foreground and the output will be something like:

```
*** swatchdog version 3.2.4 (pid:9560) started at Thu Feb 16 13:30:04 AEDT 2017
```

To test that Swatchdog works, you can execute a command using sudo (as a normal user with sudo privileges), for example:

```
sudo touch /tmp/test.txt
[sudo] password for anthony:
```

After executing this command you'll see the following output on the Swatchdog terminal:

```
Feb 16 13:27:45 tomcat-c7x-x64 sudo: anthony : TTY=pts/0 ; PWD=/root ;
USER=root ; COMMAND=/bin/touch /tmp/test.txt
1 data packet(s) sent to host successfully.
```

If everything is configured correctly you'll also see this in the `nagios.log` file on the Nagios XI server:

```
[1487212066] Warning:  Passive check result was received for service 'Log Events' on
host 'tomcat-c7x-x64.box293.local', but the host could not be found!
[1487212066] Error: External command failed -> PROCESS_SERVICE_CHECK_RESULT;tomcat-
c7x-x64.box293.local;Log Events;1;anthony : COMMAND=/bin/touch /tmp/test.txt
[1487212066] External command error: Command failed
```

Great, it's working. Here is an explanation of what just happened, starting with the configuration that was defined:

```
watchfor /sudo/
      echo red
      exec /usr/local/bin/to-nsca.sh '$_[5] : $_[13] $_[14] $_[15] $_[16] $_[17] $_[18]'
```

- The word `sudo` was matched in the new entry that was received in the log file
- The `echo` line outputted the log line on the terminal (helpful for seeing what is going on but is purely informational)

- The `exec` line executed the `to-nsca.sh` script using the specific variables as input:
  - `$_[5] = anthony`
  - `$_[13] = COMMAND=/bin/touch`
  - `$_[14] = /tmp/test.txt`
  - `$_[15] $_[16] $_[17] $_[18]` had no values (the reason for these will be explained shortly)
  - This will result in a status information line in Nagios XI that will look like this:
    - `anthony : COMMAND=/bin/touch /tmp/test.txt`
  - You can see how the information that is useful was sent to Nagios XI by constructing a string using variables

You can see the obvious security implication here. You are now able to send to Nagios XI the information of when someone acts as root through the sudo utility, telling you who it was and what they did. Nagios XI can then send an alert and it can be investigated immediately.

Why are `$_[15] $_[16] $_[17] $_[18]` being included in the Swatchdof exec line when there are no values for them? If the sudo command executed by anthony had more arguments, then they would have populated these variables and would be included in the information sent to Nagios XI. Swatchdog does not appear to have a way of defining a range or variables to use, hence why there are so many variables on the `exec line`.

Swatchdog is currently running in the foreground, to kill it press **CTRL + C**. Configuring your server to run Swatchdog as a daemon will be covered in more detail later in this documentation.

The `.swatchdogrc` file can have as many "watchfor" statements as you require.

Swatchdog Running Parameters

In the last chapter you will have started swatchdog using the following command:

```
swatchdog -t '/var/log/secure'
```

The `-t` argument tells Swatchdog to `"tail"` the file `'/var/log/secure'`. Tail means to watch new lines being added to the file. When Swatchdog is started with the `-t` argument it will continually tail the file `/var/log/syslog` or `/var/log/messages`.

If you want Swatchdog to tail more than one file you can provided multiple files separated by a space, making sure then are enclosed by 'single quotes', for example:

```
swatchdog -t '/var/log/messages /var/log/secure'
```

Run Swatchdog As A Daemon

Currently you have been able to start Swatchdog using the following command:

```
swatchdog -t '/var/log/secure'
```

However, as identified earlier this runs in the foreground and requires you to be logged on to the server. You can start Swatchdog as a daemon using the `--daemon` argument:

```
swatchdog -t '/var/log/secure' --daemon
```

You can check if it is running be executing the following command:

```
ps -ef | grep swatchdog
```

Which will output something like:

```
root      9636     1  0 14:04 ?        00:00:00 /usr/local/bin/swatchdog -t
/var/log/messages /var/log/secure --daemon
```

However this isn't entirely reliable, if you restart the server then Swatchdog isn't going to be started until you login again and execute the command.

You can create a service that starts the command when the system boots. Here is an example of how to create a service in RHEL/CentOS 7.x.

NOTE: You need to remove any of the "echo red" lines from your `.swatchdogrc` file as they will cause unexpected results in Nagios XI.

Create the service file by executing the following command:

```
vi /etc/systemd/system/swatchdog.service
```

Paste the following into the `swatchdog.service` file:

```
[Unit]
Description=Swatchdog Service
After=network.target

[Service]
Type=forking
User=root
ExecStart=/usr/local/bin/swatchdog -c /root/.swatchdogrc -t '/var/log/messages /var/log/secure' --daemon

[Install]
WantedBy=multi-user.target
```

Save the file and exit vi.

You will now need to execute these command to recognize the new service and configure it to start on boot:

```
systemctl daemon-reload
systemctl enable swatchdog.service
```

Now start Swatchdog using the following command:

```
systemctl start swatchdog.service
```

You should perform tests to ensure that Swatchdog is working as expected.

If you have a different operating system then you will need to research on how to create a service as this is getting out of the scope of what this documentation can cover.

Create Nagios XI Service

Now that Swatchdog is configured the last remaining step is to create the service objects in Nagios XI. Please refer to the following documentation:
Monitoring Unconfigured Objects With Nagios XI

These services are created as Passive services. The following documentation explains passive services in more detail:
Configuring Passive Services With Nagios XI

Finishing Up

This completes the documentation on log monitoring with Swatchdog and Nagios XI.

If you have additional questions or other support related questions, please visit us at our Nagios Support Forums:

https://support.nagios.com/forum

The Nagios Support Knowledgebase is also a great support resource:

https://support.nagios.com/kb