OSSIM

Open Source Security Information Management

Brian E. Lavender

Sac State

CSC 250, Spring 2008

Final Project

2008

Table of Contents

Introduction

According to O'Reilly's book titled *Network Security Hacks* with its introduction to Snort (Hack #106) states the following observation.

> *Monitoring your logs can take you only so far in detecting intrusions. If logs are being generated by a service that has been compromised, welcome to one of the security admin's worst nightmares: you can no longer trust your logs.*

Even this gives administrators only a certain level of confidence. SNORT analyzes the network for suspicious packets. It will produce the same alarm for a malicious packet targeted for a Windows host as a Unix host, even though the Unix host may not be vulnerable. A Security Information Manager is a tool that correlates information producing a higher confidence level for when an attack occurs. In the open source community, various tools have been created to monitor different aspects of security. OSSIM combines the data from these tools correlating it to a higher confidence when an attack occurs or a host has been compromised and also uses the data to determine the health of our network. It integrates Host Intrusion Detection Systems (HIDS) with Network Intrusion Detection Systems (NIDS) to do this.

How OSSIM Functions

OSSIM consists of three different key components. The server, the frameworkd, and the agent. Management is performed through a web based interface and configuration is done through a series of series of configuration files. Multiple agents can be placed throughout the network. The agent gathers

information from plugins and sends the data to the server. Below are a list of plugins contained with OSSIM. Custom plugins can also be written as detailed in this report.

The book *Network Security Hacks* [1] details how to configure many of these plugins (listed beside the plugin if featured in the book). Having an understanding of each plugin is beneficial to what value the plugin provides to OSSIM.

- Arpwatch, used for mac anomaly detection. (Lockart, 185)
- P0f, used for passive OS detection and os change analisys. (Lockart, 128)
- Pads, used for service anomaly detection.
- Nessus, used for vulnerability assessment and for cross correlation (IDS vs Security Scanner). (Lockart, 197)
- Snort, the IDS, also used for cross correlation with nessus. (Lockart, 349)
- Spade, the statistical packet anomaly detection engine. Used to gain knowledge about attacks without signature. (Lockart, 384)
- Tcptrack, used for session data information which can grant useful information for attack correlation.
- Ntop, which builds an impressive network information database from which we can get aberrant behaviour anomaly detection. (Lockart, 293)
- Nagios. Being fed from the host asset database it monitors host and service availability information. (Lockart, 283)
- Osiris, a Host Intrusion Detection System
- OCS-NG, Cross-Platform inventory solution.
- OSSEC, integrity, rootkit, registry detection and more. (Lockart, 274)

OSSIM gathers data using sensors. There are three primary ways to collect data. OSSIM also uses some integrated tools that won't be discussed her. The following illustrations which were adopted from Joel Winteregg's writeup [5] on OSSIM show the three ways OSSIM collects data. One is processing log data such as syslog (Illustration 1). The second is through passive network monitoring on a network segment using a tool that monitores network traffic such as SNORT (Illustration 2) through a promiscuous interface. The third is a tool that can be queried such as tcpwatch (Illustration 3). Nagios is also another tool that can be queried to show the health of hosts on the network.

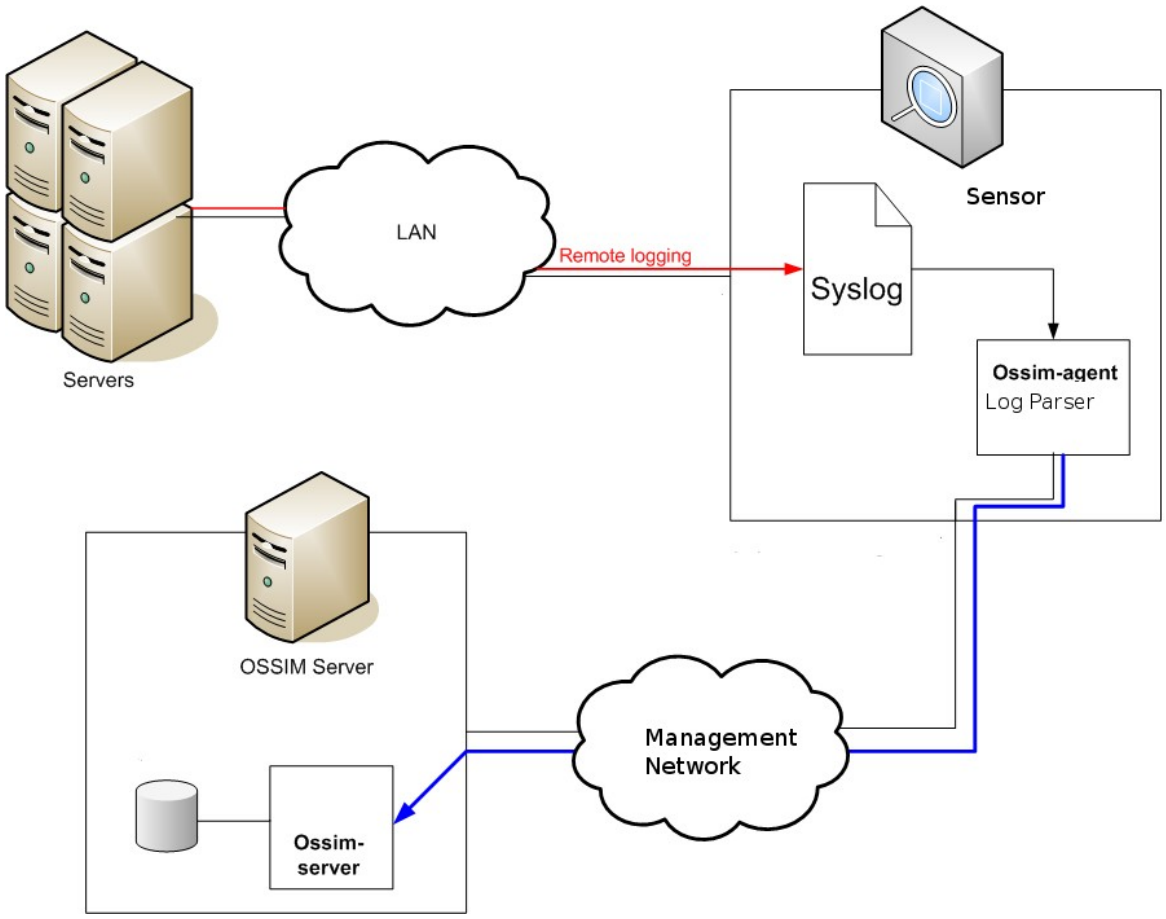


Illustration 1: Syslog

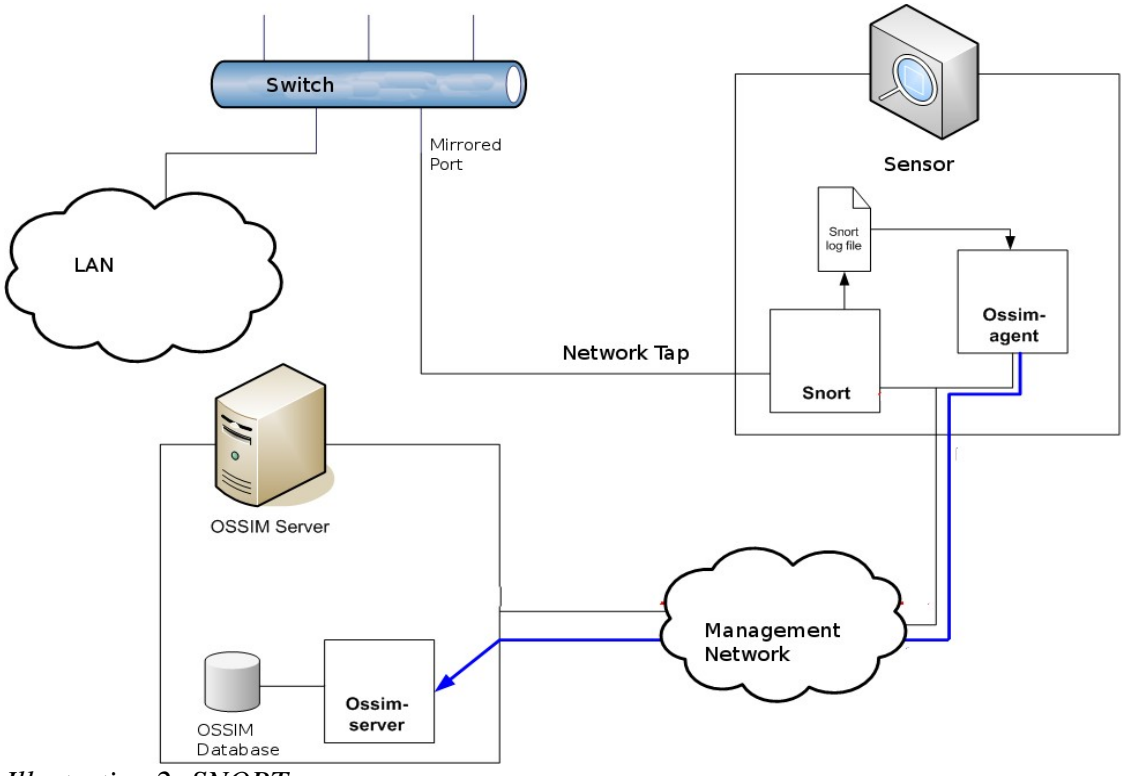


Illustration 2: SNORT

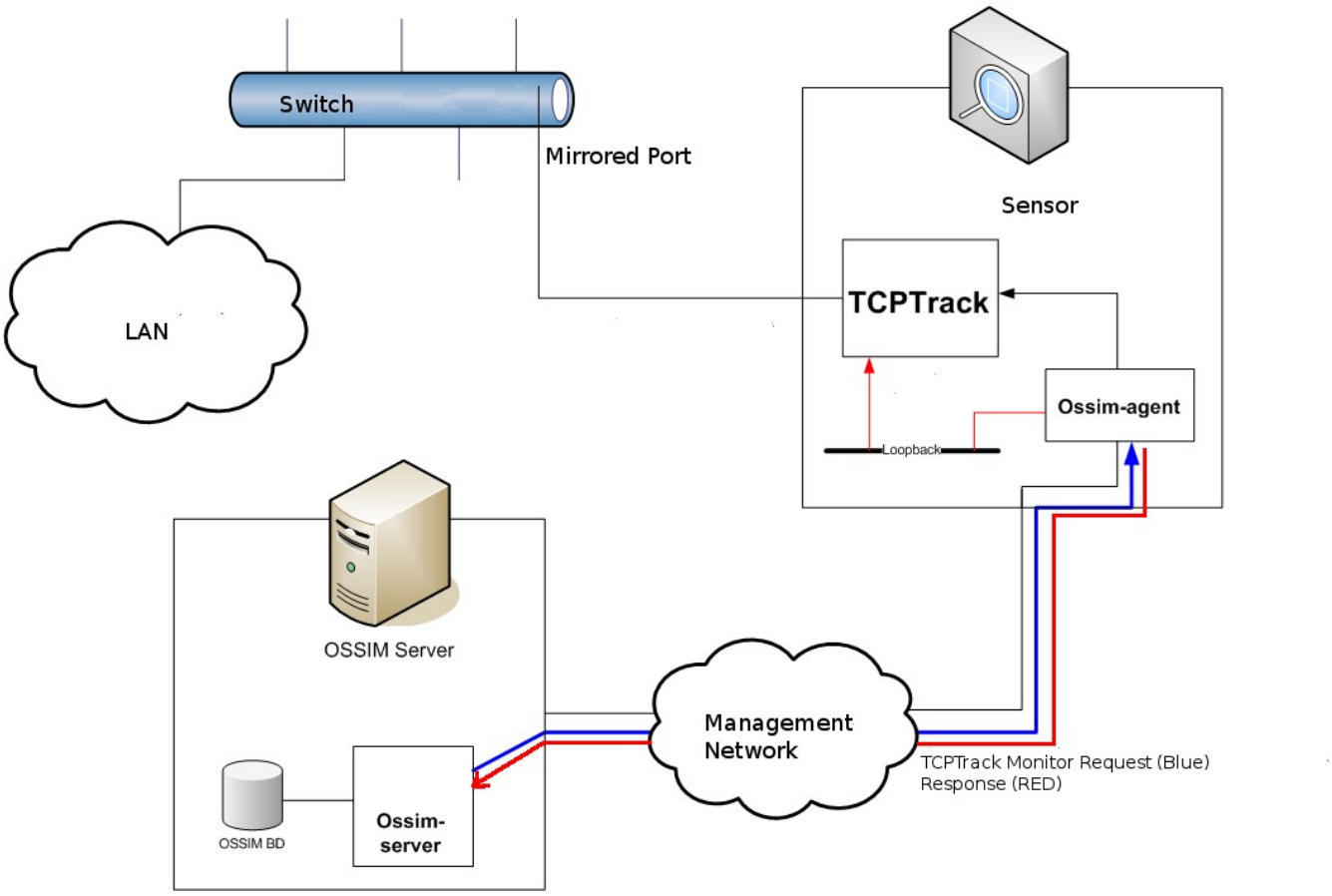


Illustration 3: Monitor

Installation

The easiest way to install OSSIM is to download the AlienVault installer from the OSSIM website. http://www.ossim.com/home.php?id=download

Burn the ISO image to a CD. It is a Debian Installation CD that has been customized to install OSSIM. It will erase the hard drive for the machine on which it is being installed. It can be used to install a virtual machine as well. The installer will ask a few brief questions. It will ask for a static IP address. Once it has asked all its questions, it will proceed to install the entire OSSIM suite (server, frameworkd, and agent) onto the system. Once installation finishes, point your web browser to the machine onto which it was installed. It will give you a login screen. The default login is 'admin' with a password of 'admin'. Log into OSSIM. You will be greeted with 'Executive Panel' which gives a high level summary of the network, incidents, alarms, and vulnerabilities. Currently, it is monitoring the host onto which it is installed.

If you want to see your brand new OSSIM server in action, you can nmap it and watch as it detects the nmap. Here are the steps to take. On the web interface, select the “Events” => “RT Events” menu

options.

Select the “start” button. As events are received by the OSSIM server, they are shown in this window in real time. Now, execute the nmap command against your OSSIM server. Do this from another host.

nmap <IP Address of newly installed OSSIM server>

The SPADE (Statistical Packet and Anomaly Detection Engine) which is a part of SNORT, will pick up the port scan. And then after detecting a series of the packets, it will issue a directive event. This will give you a quick feedback that the OSSIM server is running. There is much more to OSSIM and with some configuration, it can monitor other hosts on the network.

Initial Configuration Steps

To begin to see the value OSSIM provides, policies need to be created. Dominique Karg of the OSSIM development team has written a series of tutorials including one describing initial steps after installation. http://www.ossim.com/home.php?id=download . The following is a summary of the steps described in his tutorial. I will go through the steps briefly, but I highly recommend following his tutorial directly.

First, create a network policy by going to to the screen “Policy => Networks” and specify a network. This network is given an *asset value*, a *compromise threshold* and an *attack threshold* values. In addition, you can specify whether you want hosts in this network to be scanned by Nessus and if Nagios is enabled. See the Creating Assets and Calculating Risk section below for detailed information about how to assign risk. Individual asset values can also be specified for hosts, which will override the value given to the network.

Scan the network you just specified by going to the screen “Tools” => “Netscan” . This will run an nmap scan across the range of IP addresses that you specified in the previous step. It will list the hosts it found along with the services for each host. You can choose which ones are inserted into the database. The *risk value* given for the inserted hosts will be same as the network. It can be modified for each host by going to the screen “Policy” => “Hosts”.

Perform an OCS inventory for each host. OCS automatically collects information about the host operating system, configuration, and installed software. OSSIM integrates the OCS tools into its “Tools => Downloads” screen. The tool has been customized by the OSSIM installer, so all that needs to be done is run the setup script. The configuration parameters are already set to report OCS details back to the OSSIM installer.

Now do a NESSUS scan from the “Events=>Vulnerabilities” screen. The scans can be set to run on a regular basis. Karg's tutorial recommends raising the value for **vulnerability_incident_threshold** on the “Configuration => Main” screen.

At this point, as events arrive, a risk value will be calculated. The higher the asset value given, the

higher the risk for a received event against that host.

Creating ASSETS and Calculating RISK

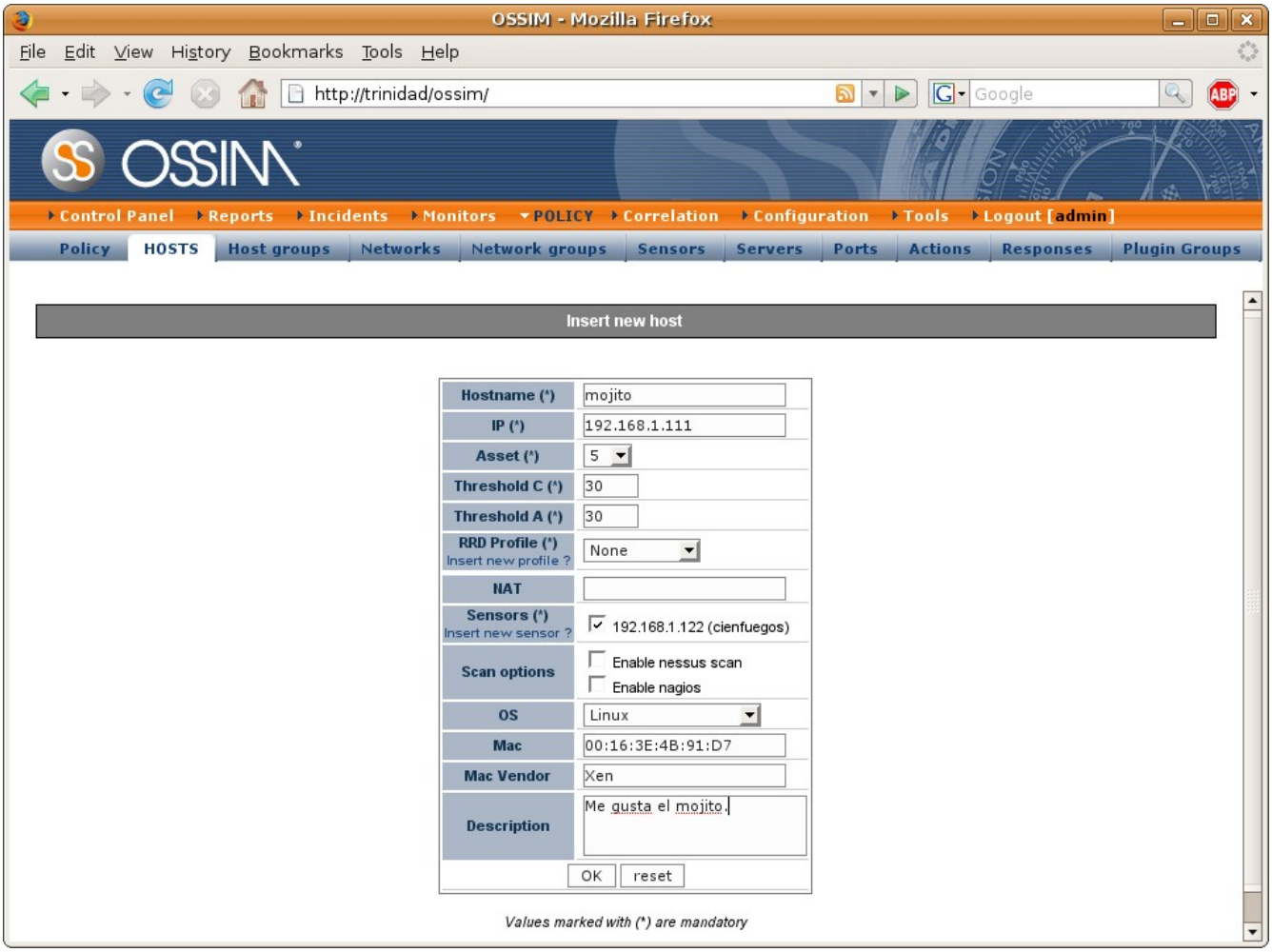
In this section, I will show the meaning of the asset value and how risk is calculated. Ossim uses asset values assigned to the systems as described in the previous section combined with a reliability and priority value from received events to calculate risk. There are three ways that a host receives an asset value: it is given one, through the asset value of the network on which it resides, or it doesn't have an assigned asset value. In turn, this host asset value is used to calculate risk when an event is received. I will describe below where the priority and risk values come from.

Create a host asset value

Asset values were covered in the previous section. For a host you can view its value under policy→hosts. This asset value ranges from 1 to 5. 1 signifies the host has little value. 5 is the highest value of importance one can give a host. Risk is calculated with the following formula.

risk = asset * (reliability * priority / 25)

Below is a screen shot showing the data for the host named mojito. It has an IP address of 192.168.1.111 and highest asset value of 5.



The below image shows the same event and its impact on two different hosts. In the following case, the event *foobar: alien foo on (DST_IP)* occured to two different hosts. This event is a log event that came from syslog and is further explained below. The event has a reliability = 10 and priority = 5 (Shown in the table below).

Event	**foobar: alien foo on (DST_IP)**
reliability	10
priority	5

And if you look at the following diagram, the risk for the first event is 10, and the second event is 2. Illustration 4 shows a screen shot as it is displayed in the events tab of the administrative interface.

risk = asset * (reliability * priority / 25)

Host	IP Address	Risk
mojito	192.168.1.111	**10** = 5 * (10 * 5 / 25)
eldedo	192.168.1.135	**2** = 1 * (10 * 5 / 25)

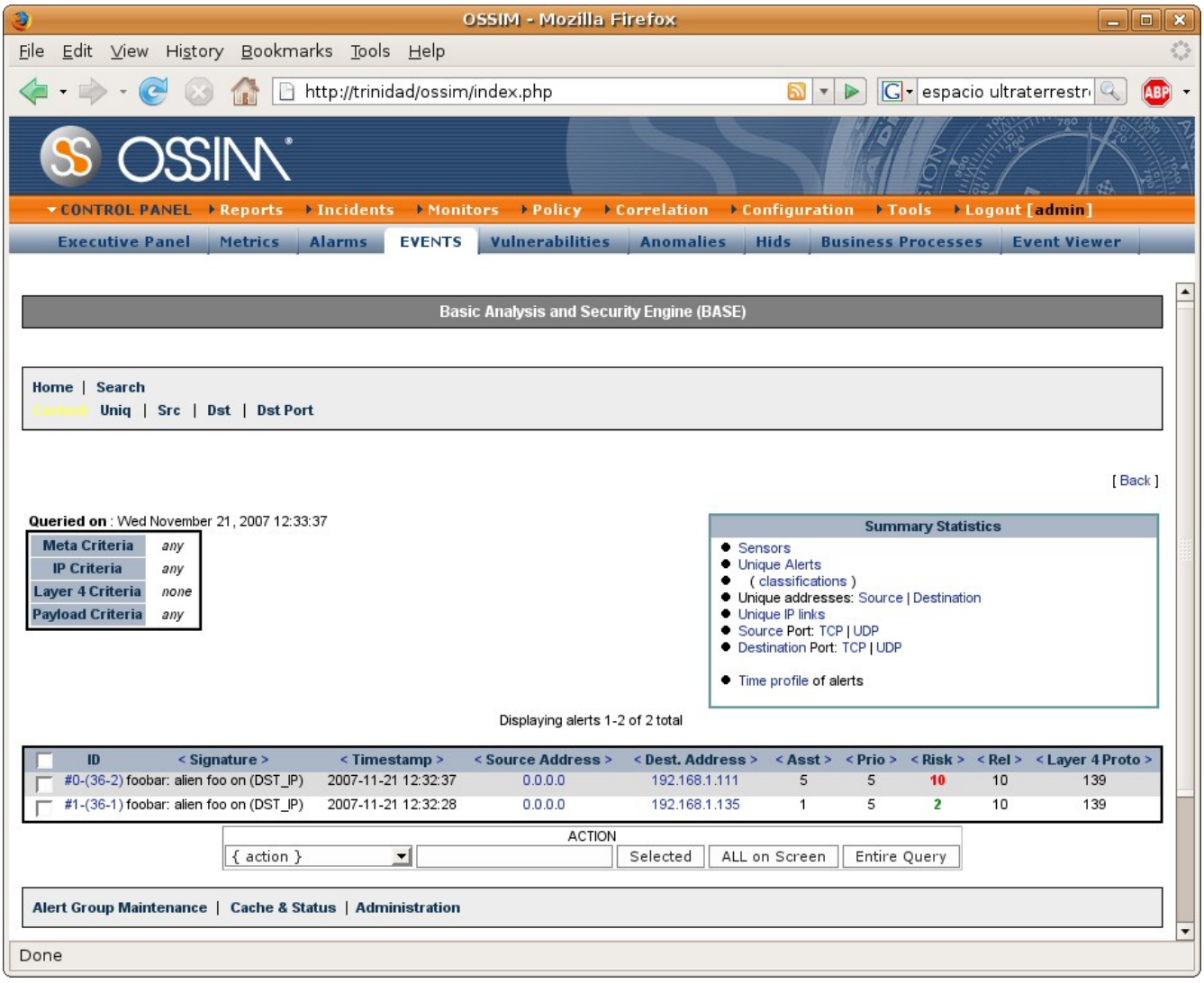


Illustration 4: Events Tab

A Customized Plugin

This is an area where priority and reliability is assigned to an event. OSSIM allows for the creation of custom plugins that will capture events specific to a user's network. This will focus on the steps needed to create an OSSIM plugin. This will be a simple plugin that you can trigger using a small python script

that sends a message to syslog. This process can be used to verify that the agent and server are functioning and that the agent can send information to the server. It will also serve as a tutorial for configuring and utilizing other plugins.

Some Theory

Each plugin has an id and series of sub ids for each type of event it can generate. For each of these sub ids, it has the associated priority and reliability value. When an event is sent to the server, the server gives the event each of these two values. Then the event is processed and a risk is calculated in combination with the asset value of the host associated with the event.

OSSIM Server Configuration

In the previous tables showing risk, an event came from a foobar plugin. The following demonstrates how to create the a plugin for foobar. On the OSSIM server, the ossim database needs to be updated with information regarding the plugin. You can copy and paste the following and it will create the file with the sql. If you create the file manually, be sure to remove the backslashes before any '$' symbol.

```
cat > ./foobar.sql << __END__
-- foobar
-- plugin_id: 20000
--
-- \$Id:\$
--
DELETE FROM plugin WHERE id = "20000";
DELETE FROM plugin_sid where plugin_id = "20000";


INSERT INTO plugin (id, type, name, description) VALUES (20000, 1, 'foobar',
'Foobar demo detector');


INSERT INTO plugin_sid (plugin_id, sid, category_id, class_id, reliability,
priority, name) VALUES (20000, 1, NULL, NULL, 6, 4, 'foobar: new foo found on
(DST_IP)');
INSERT INTO plugin_sid (plugin_id, sid, category_id, class_id, reliability,
priority, name) VALUES (20000, 2, NULL, NULL, 6, 1, 'foobar: foo the same on
(DST_IP)');
INSERT INTO plugin_sid (plugin_id, sid, category_id, class_id, reliability,
priority, name) VALUES (20000, 3, NULL, NULL, 10, 2, 'foobar: foo changed on
(DST_IP)');
INSERT INTO plugin_sid (plugin_id, sid, category_id, class_id, reliability,
priority, name) VALUES (20000, 4, NULL, NULL, 8, 3, 'foobar: foo deleted on
(DST_IP)');
INSERT INTO plugin_sid (plugin_id, sid, category_id, class_id, reliability,
priority, name) VALUES (20000, 5, NULL, NULL, 10, 5, 'foobar: alien foo on
(DST_IP)');
__END__
```

Now the plugin can be inserted into the OSSIM server using the following command.

```
cat foobar.sql | mysql -u root -p ossim
```

The OSSIM server must be restarted so that it is aware of the new plugin information.

```
/etc/init.d/ossim-server restart
```

Once the plugin exists the OSSIM web interface will verify it in the window: Configuration→Plugins (Illustration 5).

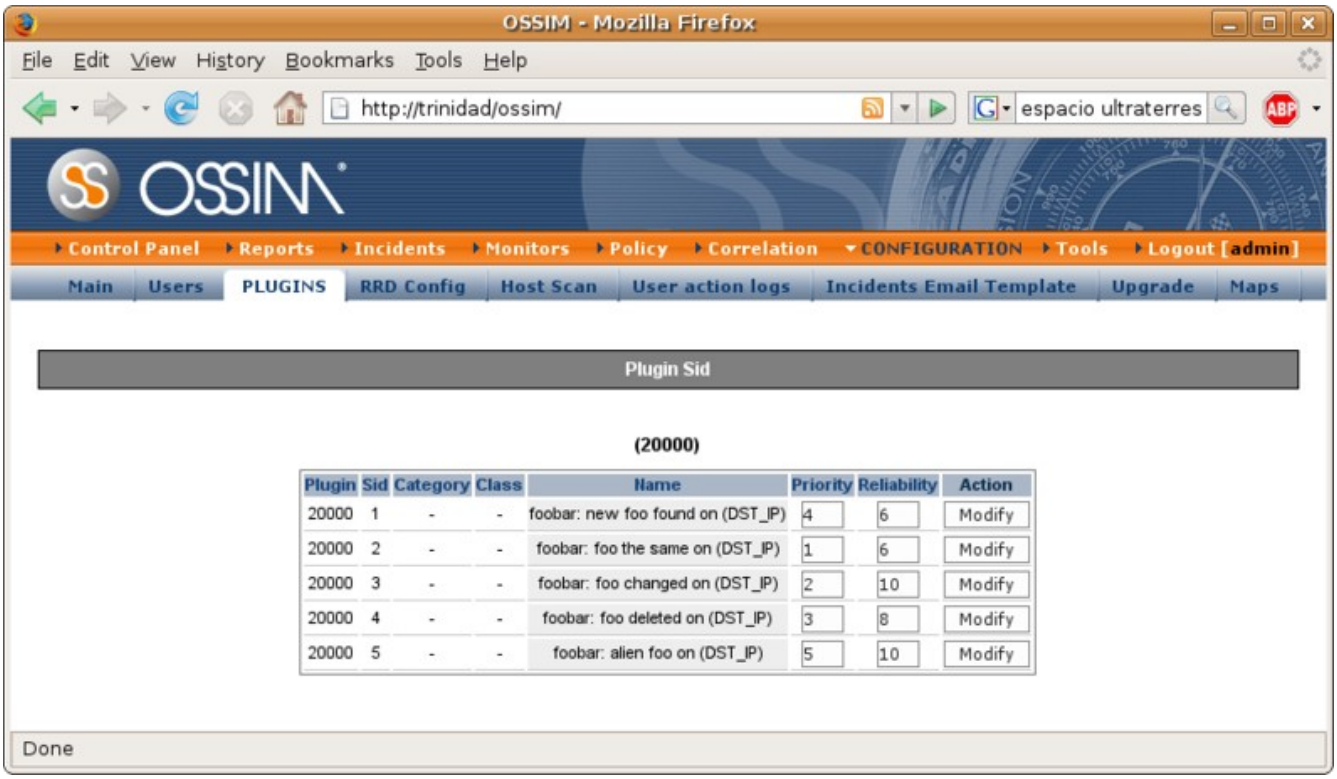


Illustration 5: Plugin

Modification of the values in the above illustration for reliability and priority for each plugin_sid, requires restart of the OSSIM server in order for it to take effect.

OSSIM Agent Configuration

The following steps detail configuration of the agent for the plugin. This plugin is going to monitor syslog for the output, so a config file for the plugin must exist containing the plugin ID and how to match information in syslog. In this case, it matches only one sid, but as you can see from the above sql, there could be five patterns and five sub ids.

Contents of /etc/ossim/agent/plugins/foobar.cfg You can copy and paste into the shell. If you create the file manually, be sure to remove the backslashes before any '$' symbol.

```
cat > /etc/ossim/agent/plugins/foobar.cfg << __END__
;; foobar
;; plugin_id: 20000
;; type: detector
;; description: foobar demo plugin
;;
;; URL:
;;
;; \$Id:\$

[DEFAULT]
plugin_id=20000

[config]
type=detector
enable=yes

source=log
location=/var/log/user.log

# create log file if it does not exists,
# otherwise stop processing this plugin
create_file=false

process=
start=no
stop=no
startup=
shutdown=


## rules

##
## New foo found in bar
##

[foobar - New foo found]
# Sep  7 12:40:55 eldedo FOOBAR[2054]: new foo found
event_type=event
regexp="(\S+\s+\d+\s+\d\d:\d\d:\d\d)\s+(?P<dst_ip>[^\s]*).*?FOOBAR.*?new foo
found"
plugin_sid=1
dst_ip={resolv(\$dst_ip)}
src_ip=0.0.0.0
date={normalize_date(\$1)}

[foobar - foo the same]
# Sep  7 12:40:55 eldedo FOOBAR[2054]: foo the same
event_type=event
regexp="(\S+\s+\d+\s+\d\d:\d\d:\d\d)\s+(?P<dst_ip>[^\s]*).*?FOOBAR.*?foo the same"
plugin_sid=2
dst_ip={resolv(\$dst_ip)}
```

```
src_ip=0.0.0.0
date={normalize_date(\$1)}


[foobar - New changed]
# Sep  7 12:40:55 eldedo FOOBAR[2054]: foo changed
event_type=event
regexp="(\S+\s+\d+\s+\d\d:\d\d:\d\d)\s+(?P<dst_ip>[^\s]*).*?FOOBAR.*?foo changed"
plugin_sid=3
dst_ip={resolv(\$dst_ip)}
src_ip=0.0.0.0
date={normalize_date(\$1)}

[foobar - New deleted]
# Sep  7 12:40:55 eldedo FOOBAR[2054]: foo deleted
event_type=event
regexp="(\S+\s+\d+\s+\d\d:\d\d:\d\d)\s+(?P<dst_ip>[^\s]*).*?FOOBAR.*?foo deleted"
plugin_sid=4
dst_ip={resolv(\$dst_ip)}
src_ip=0.0.0.0
date={normalize_date(\$1)}


[foobar - alien foo]
# Sep  7 12:40:55 eldedo FOOBAR[2054]: alien foo
event_type=event
regexp="(\S+\s+\d+\s+\d\d:\d\d:\d\d)\s+(?P<dst_ip>[^\s]*).*?FOOBAR.*?alien foo"
plugin_sid=5
dst_ip={resolv(\$dst_ip)}
src_ip=0.0.0.0
date={normalize_date(\$1)}
__END__
```

We need to tell the agent that we have a new plugin. Edit the file /etc/ossim/agent/config.cfg and add the following line in the [plugin] section.

```
foobar=/etc/ossim/agent/plugins/foobar.cfg
```

Now to restart the agent so that it is aware of the new plugin information.

```
/etc/init.d/ossim-agent restart
```

Verification

This is a sample python script that will send a message to syslog. I parses the optios sent and sends a log message for each option that matches the case. The following code can be run as a script on any host that has Python installed.

```
#! /usr/bin/python
import syslog
import sys
```

```
syslog.openlog("FOOBAR", syslog.LOG_PID , syslog.LOG_USER )
for arg in sys.argv:
        if arg == "1":
                syslog.syslog(syslog.LOG_WARNING, "new foo found")
        elif arg == "2":
                syslog.syslog(syslog.LOG_WARNING, "foo the same")
        elif arg == "3":
                syslog.syslog(syslog.LOG_WARNING, "foo changed")
        elif arg == "4":
                syslog.syslog(syslog.LOG_WARNING, "foo deleted")
        elif arg == "5":
                syslog.syslog(syslog.LOG_WARNING, "alien foo")

syslog.closelog()
```

Run this program on the server for which you want to generate the event. The following will send the first type syslog message.

```
testfoobar.py 1
```

The second will send the 5th type syslog message, the 4th type syslog message, and then finally the 2nd type syslog message.

```
testfoobar.py 5 4 2
```

Check your events and alarms. An event and/or an alarm should appear on the event tab previously shown.

A sample OSSIM directive

OSSIM stores its rules on the server in a file named /etc/ossim/server/directives.xml. The rules are separated into directives. The following is an example ssh brute force directive. This rules from this directive obtains its information from the ssh auth.log plugin. In this case, the attacker could be switching different hosts to attack in attempt to escape detection on a single host, but this directive will detect those attempts between switched target hosts as well. The reliability begins at 3 after three failed attempts. Three more will raise it to 4. Five more will raise it 6, and then an additional 10 attempts will raise it to 8.

```
<directive id="20" name="Possible SSH brute force login attempt against DST_IP"
priority="5">
  <rule type="detector" name="SSH Authentication failure" reliability="3"
    occurrence="1" from="ANY" to="ANY" port_from="ANY" port_to="ANY"
    time_out="10" plugin_id="4003" plugin_sid="1,2,3,4,5,6">
      <rules>
        <rule type="detector" name="SSH Authentication failure (3 times)"
          reliability="+1" occurrence="3" from="1:SRC_IP" to="ANY"
          port_from="ANY" time_out="15" port_to="ANY"
```

```
          plugin_id="4003" plugin_sid="1,2,3,4,5,6" sticky="true">
          <rules>
            <rule type="detector" name="SSH Authentication failure (5 times)"
              reliability="+2" occurrence="5" from="1:SRC_IP" to="ANY"
              port_from="ANY" time_out="20" port_to="ANY"
              plugin_id="4003" plugin_sid="1,2,3,4,5,6" sticky="true">
              <rules>
                <rule type="detector" name="SSH Authentication failure (10 times)"
                  reliability="+2" occurrence="10" from="1:SRC_IP" to="ANY"
                  port_from="ANY" time_out="30" port_to="ANY"
                  plugin_id="4003" plugin_sid="1,2,3,4,5,6" sticky="true">
                </rule>
              </rules>
            </rule>
          </rules>
        </rule>
      </rules>
    </rule>
</directive>
```

The above directive only explored rules that are sensors. You You in his paper walks through an attack with a sample DCOM exploit (YouYou). Dominique Karg also goes through the meaning of the details for the XML syntax such as sticky .

Conclusion

There are many aspects to OSSIM. More than one server can be created. This document has skipped over those details to cover the base concepts. OSSIM has high complexity, yet at the same time, it has the sophistication to handle the threats networks and hosts are exposed to today. It strength is derived from integration of other tools. It is adept to handle the new weaknesses in the network. Part of the issue of understanding OSSIM is setting up the network and components. The goal of security is to be equally as good as the attacker, and yet at the same time to think ahead of him. And when one is not able to think ahead, have tools that look for anomalies. Through the fact that OSSIM is open source, it has the capability to build through the community. Many eyes are its tools for security.

References

1. Lockhart , Andrew, *Network Security Hacks*, O'Reilly Media, Inc., ISBN 0596006438, 1st Edition, 2004
2. 陈深猷,YouYou <chensy at netway.net.cn>, Lance <lance at antpower.org> *A Practice of OSSIM*, .http://www.ossim.net/docs/A_Practice_for_Ossim.pdf, 2004.
3. Casal, Julio*, OSSIM Fast Guide,* http://www.ossim.net/docs/OSSIM-fastguide.pdf *,* 2004.

4. Draves, Curtis, *OSSIM,* http://www.ossim.net/docs/OSSIM-desc-en.pdf, 2003.
5. Winteregg, Joêl, *Fonctionnement d'OSSIM*, http://www.ossim.net/docs/OssimJWinteregg.pdf, 2005.
6. Lavender, Brian, *Create a Simple OSSIM Plugin*, http://www.ossim.net/dokuwiki/doku.php?id=architecture:plugin_writing, 2007.
7. Lavender, Brian, *Create a Host Asset*, http://www.ossim.net/dokuwiki/doku.php?id=architecture:hostcreate, 2007.
8. Karg, Dominique, *OSSIM Correlation Engine Explained,* http://www.ossim.net/docs/correlation_engine_explained_rpc_dcom_example.pdf, 2004.