Suricata with Transparent Proxy

Ajay Brahmakshatriya,[*]
Pratik Bhatu[†]
20-02-2015

Abstract

This document describes the process of configuring Suricata in Intrusion Detection Mode and Intrusion Prevention Mode. It also compares data traffic logs with Squid and describes the configuration of Squid with instructions. It contains benchmarks for performance in high traffic situations.

This manual is meant to be instructional to help network administrators configure Suricata for blocking and detecting unwanted traffic. It contains benchmarks which would help in deciding system configuration.

Contents

1 Overview **2**

2 Installing Suricata 2.0.6 in Ubuntu **2**

3 Rules **3**

4 Configuration **4**

5 Execution **5**

5.1 Running 5

5.2 Logging 6

6 Installing Squid Server for Testing **6**

7 Testing and Benchmarking **7**

[*] cs12b1004@iith.ac.in
[†] cs12b1010@iith.ac.in

1 Overview

What is suricata?: It is an Intrusion Detection and Prevention engine. It operates in two modes IPS(Prevention) and IDS(Detection). It provides a framework of writing **r**ules to filter traffic. This makes it quite flexible and so system administrator can write desired rules to get alerts regarding suspicious traffic and also block it. It is a multi-threaded engine and so it is fast and efficient in analyzing the network traffic.

IDS Mode: In this mode, the engine only monitors the data and does not block any malicious data. It processes each packet and based on the rules it alerts the administrator. Here the traffic is simply mirrored to suricata.

IPS Mode: In this mode suricata acts as a firewall between the host and the outside world. It is capable of actively dropping traffic that are deemed malicious by the rules.

Configuration: The configuration for the engine is in a file **suricata.yaml**. We will discuss more about possible configurations later.

Rules: We can write rules on how to process each packet. Suricata gives us the ability to block traffic based on target host/port, source port/host but also based on content of the packet (eg: words contained). However it does not work on httpspacket since the content is encrypted. There are a default set of rules that are used by suricata that can be downloaded from `emergingthreats.net` or the setup can be done using oinkmaster.

Logs: Suricata maintains a set of logs like http, dns, packet, stats and these can be configured in suricata.yaml. fast.log contains the intrusions detected or prevented by suricata.

2 Installing Suricata 2.0.6 in Ubuntu

Pre-installation requirements: This is the list of packages that need to be already installed in the host machine. Run the following commands for installing/updating them.

```
sudo apt-get -y install libpcre3 libpcre3-dbg libpcre3-dev \
build-essential autoconf automake libtool libpcap-dev \
libnet1-dev libyaml-0-2 libyaml-dev zlib1g zlib1g-dev \
libcap-ng-dev libcap-ng0 make libmagic-dev
```

Download: To download the source code of suricata run the following command.

```
wget -E http://www.openinfosecfoundation.org/download/suri\
cata-2.0.6.tar.gz
tar -xvzf suricata-2.0.6.tar.gz
cd suricata-2.0.6
```

Configure IDS Mode: To build suricata so that it only runs in IDS mode simply enter the following commands. This configures the make file so that the installation directory is /usr, the configuration files are in /etc/suricata and the log files are in /var/log/suricata.

```
./configure --prefix=/usr --sysconfdir=/etc\
 --localstatedir=/var
```

Configure IPS Mode: In this mode suricata can also work in prevention mode. For this we need to install some more prerequisite packages:

```
sudo apt-get -y install libnetfilter-queue-dev \
libnetfilter-queue1 libnfnetlink-dev libnfnetlink0
```

Like in IDS mode here too we need to configure the make file for compilation. Enter:

```
./configure --enable-nfqueue --prefix=/usr --sysconfdir=/etc \
 --localstatedir=/var
```

For both these modes if you want to enable packet profiling for reading pcap files do:

```
./configure --enable-profiling
```

This would be later used in benchmarking process.

Compiling and Installing: Now we just need to build the source code and install the engine. Enter:

```
sudo make
sudo make install-conf
sudo ldconfig
```

This install the engine in corresponding folders and generates the configuration file, suricata.yaml in `/etc/suricata`. One could alternately run:

```
make install-rules
```

This would do the regular installation and would automatically download and set up the latest rule set from Emerging Threats available for Suricata.

3 Rules

If one does not run `make install-rules` he needs to manually create the directory `/etc/suricata/rules` and download the latest rule set from `emergingthreats.net` if he wants to use the current updated rule set for detecting malicious traffic. One can alternatively write his own new rules and change the suricata.yaml to select desired rules. We will describe this in the next section. Next we see on how to write custom rules.

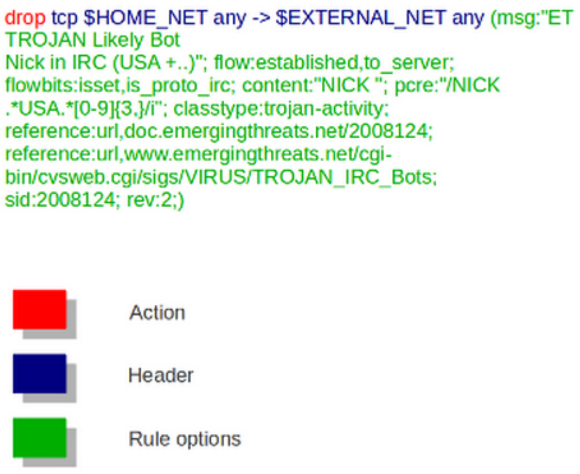
Writing Rules: A rule consists of the action, header and rule-options.

Actions: There are four types of actions.

- pass : Skip packet and don't match with further rules.
- drop: Drop the packet in IPS mode. Generates tcp timeout at host.
- reject: Actively reject the packet by sending a reset packet to sender and host for TCP.
- alert: Just generate an alert to inform system administrator.

Protocol: The options available are tcp, udp, icmp and ip where ip is equivalent to all packets. Suricata also includes http, ftp, tls, smb and dns as options for matching protocols at higher levels.

Source and Destination: After specifying protocol we can specify source address, source port, destination address and destination port in the following format.



```
drop tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"ET
TROJAN Likely Bot
Nick in IRC (USA +..)"; flow:established,to_server;
flowbits:isset,is_proto_irc; content:"NICK "; pcre:"/NICK
.*USA.*[0-9]{3,}/i"; classtype:trojan-activity;
reference:url,doc.emergingthreats.net/2008124;
reference:url,www.emergingthreats.net/cgi-
bin/cvsweb.cgi/sigs/VIRUS/TROJAN_IRC_Bots;
sid:2008124; rev:2;)
```

`src_addr src_port -> dest_addr dest_port`

Here we can specify values or put `any` for it to match any address/-port. We can also use variables defined in suricata.yaml like `HOME_NET`. Now for port numbers we can also define ranges using `:`, for example [2:45]. We can also use negation, for example !80 meaning every port but 80 and also multiple ports [2, 4, 6].

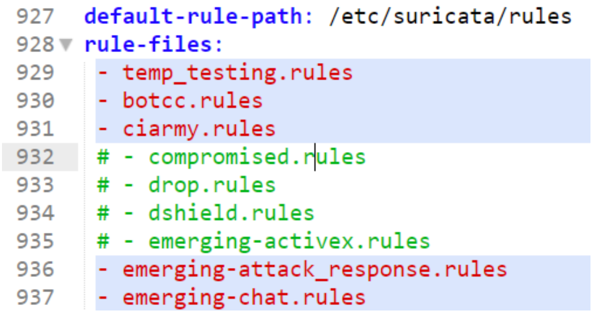
Direction: The direction tells in which way the rule has to match. For example, source − > destination (only from source to destination) and source <> destination (both directions).

Rule Options: They are of the format [`name : settings;`]. We can filter based on packet containing some text (eg. content : ".torrent"). For more information regarding this visit `goo.gl/HZ7WHs`

4 Configuration

The configuration for suricata is stored in `suricata.yaml` in `/etc/suricata`. It holds special syntaxes and all the configuration variables like networks, interfaces, log/rules directories, etc..It has thorough comments regarding the various options and we'll cover a few of them.

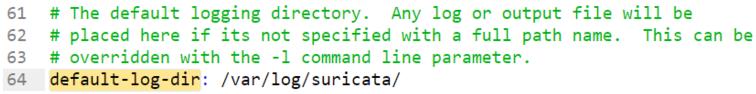
Rule Enabling/Disabling: You can change the default path to the rules directory by changing the `default-rule-path` variable. The rules are under the rule-files section. To disable a rule simply add a `#` as a prefix to that line. Here the rules



```
default-rule-path: /etc/suricata/rules
rule-files:
 - temp_testing.rules
 - botcc.rules
 - ciarmy.rules
 # - compromised.rules
 # - drop.rules
 # - dshield.rules
 # - emerging-activex.rules
 - emerging-attack_response.rules
 - emerging-chat.rules
```

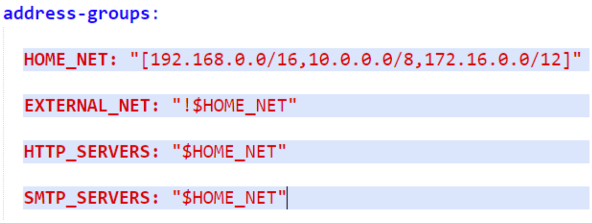
in green are disabled.

Log Directory: The directory where the logs are stored can be changed.



```
# The default logging directory.  Any log or output file will be
# placed here if its not specified with a full path name.  This can be
# overridden with the -l command line parameter.
default-log-dir: /var/log/suricata/
```

Rule-vars: There are variables which can be used in rules. Within rules, there is a possibility to set for which IP address the rule should be checked and for which IP address it should not. This way, only relevant rules will be used. So we have address groups. For example, `HOME_NET` would be any network you want Suricata to inspect.



```
address-groups:

  HOME_NET: "[192.168.0.0/16,10.0.0.0/8,172.16.0.0/12]"

  EXTERNAL_NET: "!$HOME_NET"

  HTTP_SERVERS: "$HOME_NET"

  SMTP_SERVERS: "$HOME_NET"
```

5 Execution

Now we describe the steps involved in running suricata and viewing the logs.

5.1 Running

First we decide the interface to monitor say ethX. We need to switch-off `GRO`(Generic Receive Offload) for that interface for suricata to function normally. Enter:

```
sudo ethtool -K ethX gro off
```

IDS Mode: Simple enter the following command.

```
sudo suricata -c /etc/suricata/suricata.yaml -i ethX
```

This tells suricata to use the configuration in suricata.yaml and monitor interface ethX. One can maintain various configurations in different yaml files.

IPS Mode: This is not as trivial as IDS mode since here we need suricata to actively drop traffic and so we need to channel the entire traffic through suricata. We do this by setting `iptable` rules for incoming and outgoing traffic, redirecting them through suricata. Enter the following commands:

```
sudo iptables -o ethX -I OUTPUT -j NFQUEUE
sudo iptables -i ethX -I INPUT -j NFQUEUE
```

To run it enter:

```
sudo suricata -c /etc/suricata/suricata.yaml -q 0
```

5.2 Logging

There are various logs we can view based on what we have enabled in the configuration file. The logs are stored by default in the `/var/log/suricata` directory. To view the live updates of the logs use the command:

```
tail -f file.log
```

Let's cover the common ones.

http.log: This conatins the logs for each http request and response. We can enable custom http logging i.e. printing in different format by changing the yaml file in `http-log` section.

fast.log: This contains the alerts and drop messages when a packet matches a rule.

stats.log: This contains various statistics that are useful like the no. of packets lost, amount of data processed, no. of tcp/udp.. packets.

6 Installing Squid Server for Testing

Overview: Squid is used as a proxy server for directing all the Internet traffic though the machine on which suricata is installed. Squid just acts as proxy and does not do any filtering or monitoring of content. The machine on which the squid instance is setup has two interfaces, One interface has a direct access to the Internet (without any proxy), and the second interface to connected to the college network.

We have configured the squid server to forward the Internet requests from the inside network to the open internet interface. The server is also configured to not cache any content. It is acting as a plain proxy.

Installation: Simply issue the following commands:

```
sudo apt-get update
sudo apt-get install squid3
```

Configuration: The configuration file can found at `/etc/squid/squid.conf`. To accept the traffics from all addresses from the network, we need to create an ACL (Access control list) which has all the addresses. We do so by adding:

```
acl myNet src 172.16.0.0/255.255.0.0
```

We then need to allow all http access to this acl, so we add

```
http_access all myNet
```

These lines should be added before the line

```
http_access deny all
```

so that these rules are applied before. We make sure that the http port is set to 3128. This is specified by the line

```
http_port 3128
```

This should be there by default. If we want to run the server on any other port, we can make the changes accordingly.

To configure the server to cache nothing add:

```
cache deny all
```

After we are done with this, we need to ask the running instance of squid to pick up the new configuration. This can be done using

```
sudo squid3 -k reconfigure
```

We can also use the commands

```
sudo /etc/init.d/squid3 start|stop|restart
```

to start, stop or restart the server instance.

Additional configuration: We have configure the squid server to work here when it has access directly to the open internet. We might also want to configure this instance of squid to run behind/with another proxy. There are two relations a proxy can have with another proxy a peer or parent. We can set a proxy to send all the traffic to another proxy by adding the following lines to the configuration file

```
cache_peer 192.168.35.5 parent 3128 3130 default
```

Here 3128 is the port of the parent proxy 192.168.35.5 and 3130 is the port for the ICP data. The parent proxy may also require authentication, which can be either hardcoded in the config file like

```
cache_peer 192.168.35.5 parent 3128 3130
default login=cs12b1004:*******
```

or we can ask the user to enter the authentication for the parent proxy as

```
cache_peer 192.168.35.5 parent 3128 3130
default login=PASSTHRU
```

we also need to add the config line:

```
never_direct allow all
```

to make sure that all the traffic goes through the parent server.

We can also have multiple parent proxies, which can be added by adding similar cache_peer lines. We need to mention some kind of load balancing algorithm like round robin in the last options field. With this we should be able to have a proxy running which we will be able to direct all traffic.

Logs: The squid server also generates logs which we will use to compare with the suricata logs. These logs can be found at `/var/log/squid3/access.log`.

Although we have used a proxy here, the final aim would be to setup suricata on the gateway router to monitor/filter traffic. So we can remove the proxy server. Essentially we would have a transparent proxy implemented via routing rules. That would certainly reduce the processing load of a squid instance

7 Testing and Benchmarking

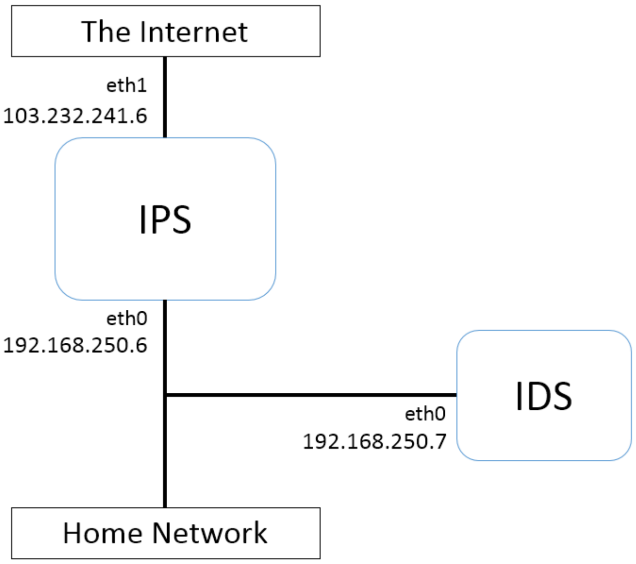
Overview: For testing purposes we have configured two virtual machines, one in IPS Mode and the other in IDS mode. When we deploy we expect IPS suricata to be running on the gateway router. We have also configured `Squid` proxy server on the IPS Machine. Therefore all data from home network would be sent to the proxy server i.e. the IPS Machine. The data coming to this machine before being processed by suricata is mirrored and sent to the IDS machine. Enter this command in IPS machine:

```
iptables -I PREROUTING -t mangle -i eth0 -j TEE \
 --gateway 192.168.250.7

iptables -I POSTROUTING -t mangle -j TEE \
--gateway 192.168.250.7
```

This ensures all data incoming to eth0 and also generated from the machine is mirrored to IDS machine(192.168.250.7).

Network Diagram:



Log Comparison: Since the purpose of suricata is to replace squid server in future, it is essential to know how suricata logging compares to the squid logging. To know this we emptied all the logs from both the servers. We started squid and suricata together and used the proxy server from one user for a few minutes. We observe that the number of requests logged in both files are same. For more rigorous testing we need to deploy the server for more number of users. This would give us a variety of domains and different types of content and so we can check whether suricata is missing any logs compared to squid, or even catching any more requests.

Following are the excerpts from both the logs.

Squid Logs:

```
424467883.995 263 172.16.3.43 TCP_MISS/200 2675 GET
http://upload.wikimedia.org/wikipedia/meta/2/27/Wikisource-
logo_sister_1x.png - HIER_DIRECT/208.80.154.240 image/png

1424467884.287 263 172.16.3.43 TCP_MISS/200 1869 GET
http://upload.wikimedia.org/wikipedia/meta/a/af/Wikiversity-
logo_sister_1x.png - HIER_DIRECT/208.80.154.240 image/png
```

Suricata Logs:

```
02/21/2015-03:01:23.994598 upload.wikimedia.org [**] /wikipedia/meta/2/27/Wikisource-logo_sister_1x.png [**] Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/40.0.2214.111 Safari/537.36 [**] 103.232.241.6:40958 − > 208.80.154.240:80

02/21/2015-03:01:24.286601 upload.wikimedia.org [**] /wikipedia/meta/a/af/Wikiversity-logo_sister_1x.png [**] Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/40.0.2214.111 Safari/537.36 [**] 103.232.241.6:40958 − > 208.80.154.240:80
```

We see that there is a one to one corresponding between these two logs. Both the logs have been attached with the manual for further reference.

Benchmarking: For benchmarking we study the CPU and RAM usage of suricata under various situations. On one side we vary the no. of active rules that need to be checked on each packet and on the other we vary the amount of request it needs to process.

Memory Usage: For getting memory usage we use the following command:

```
dstat −m <refresh_interval >[1] <no. of intervals >[100] \
> dstat.data
```

The values in the square brackets are the values we used for getting the data. For the amount of data we were able to direct to it with 3 users simultaneously, asynchronously we see that there is **no great variation** in the memory usage.

CPU Usage: We can see the current CPU usage by using:

```
top
```

This shows us the current CPU usage by suricata. Again like in the memory case we do not have enough traffic to cause significant changes in CPU usage.

Packet Profiling: Instead of sending real traffic to suricata which is difficult to do on a large scale since we haven't deployed it, we run Suricata in packet profiling mode.

In this mode suricata reads from a pcap capture file instead of monitoring an active interface. We do this by:

```
sudo suricata −c /etc/suricata/suricata.yaml −r \
/path/to/file.pcap
```