

# IDS and log processing demo with Snort, Suricata, Wazuh

## Executive Summary

The objective of the project was to establish an intrusion detection system laboratory with the assistance of the Snort, Suricata and the integration of Wazuh with Suricata. The project aimed to gain an understanding of how rules were written for these tools, how they captured incoming traffic, how the intrusion attempts were logged, how to create custom rules according to our preferences and also how suricata could be integrated into Wazuh for Log processing.

## Requirements

- VirtualBox Hypervisor
- Ubuntu VM with Snort
- Kali Linux VM
- Vulnerable Linux machine

## Snort configuration

In this lab, I have created an internal network with three virtual machines, Kali linux which is attacker VM, Ubuntu vuln which is a vulnerable machine and a Ubuntu VM which is installed with Snort. As shown in screenshot 1, First I made sure that IP subnet was assigned correctly.

```
jp@jp-VirtualBox:~$ ls -al /etc/snort/
total 388
drwxr-xr-x  3 root root   4096 Sep 29 18:53 .
drwxr-xr-x 131 root root 12288 Sep 29 18:49 ..
-rw-r--r--  1 root root   3757 Apr  3  2018 classification.config
-rw-r--r--  1 root root  82469 Apr  3  2018 community-sid-msg.map
-rw-r--r--  1 root root  31643 Apr  3  2018 gen-msg.map
-rw-r--r--  1 root root    687 Apr  3  2018 reference.config
drwxr-xr-x  2 root root   4096 Sep 29 19:02 rules
-rw-r-----  1 root snort 29010 Sep 29 18:53 snort.conf
-rw-r-----  1 root snort 20480 Sep 29 18:29 .snort.conf.swo
-rw-r-----  1 root snort 24576 Sep 29 18:50 .snort.conf.swp
-rw-----  1 root root    806 Sep 29 17:50 snort.debian.conf
-rw-r--r--  1 root root   2335 Apr  3  2018 threshold.conf
-rw-r--r--  1 root root 160606 Apr  3  2018 unicode.map
```

*Screenshot 1: Contents of /etc/snort.*

In the next step I have modified the `snort.conf` file that had `HOME_NET` to `192.168.2.0/24` which was the subnet of my VM as shown in screenshot 2.

```
# Setup the network addresses you are protecting
#
# Note to Debian users: this value is overridden when starting
# up the Snort daemon through the init.d script by the
# value of DEBIAN_SNORT_HOME_NET s defined in the
# /etc/snort/snort.debian.conf configuration file
#
ipvar HOME_NET 192.168.2.0/24

# Set up the external network addresses. Leave as "any" in most situations
ipvar EXTERNAL_NET any
# If HOME_NET is defined as something other than "any", alternative, you can
# use this definition if you do not want to detect attacks from your internal
# IP addresses:
#ipvar EXTERNAL_NET !$HOME_NET
```

*Screenshot 2: changing IP of home network to the subnet 192.168.2.0/24.*

Then, I used the command `:578,696s/^/#` to filter all the rules that were present previously since I wanted to create custom rules. After that I made sure that all the default rules were commented out with the command `usermod -l newUsername oldUsername` as shown in screenshot 3.

```
+++++
Initializing rule chains...
0 Snort rules read
  0 detection rules
  0 decoder rules
  0 preprocessor rules
0 Option Chains linked into 0 Chain Headers
0 Dynamic rules
+++++

+-----[Rule Port Counts]-----+
|      tcp      udp      icmp      ip
|      src      0       0       0       0
|      dst      0       0       0       0
|      any      0       0       0       0
|      nc       0       0       0       0
|      s+d      0       0       0       0
+-----+
```

*Screenshot 3: Commented out the default rules.*

This completed the setup to create my own rules to detect any network traffic with the help of Snort.

IP address of Kali machine: 192.168.2.5

IP address of Ubuntu snort machine: 192.168.2.4

IP address of Ubuntu vulnerable machine: 192.168.2.8

## Procedure

After the configuration of Snort was completed. I used `sudo vim /etc/snort/rules/local.rules` to create my own rules as shown in the screenshot 4.

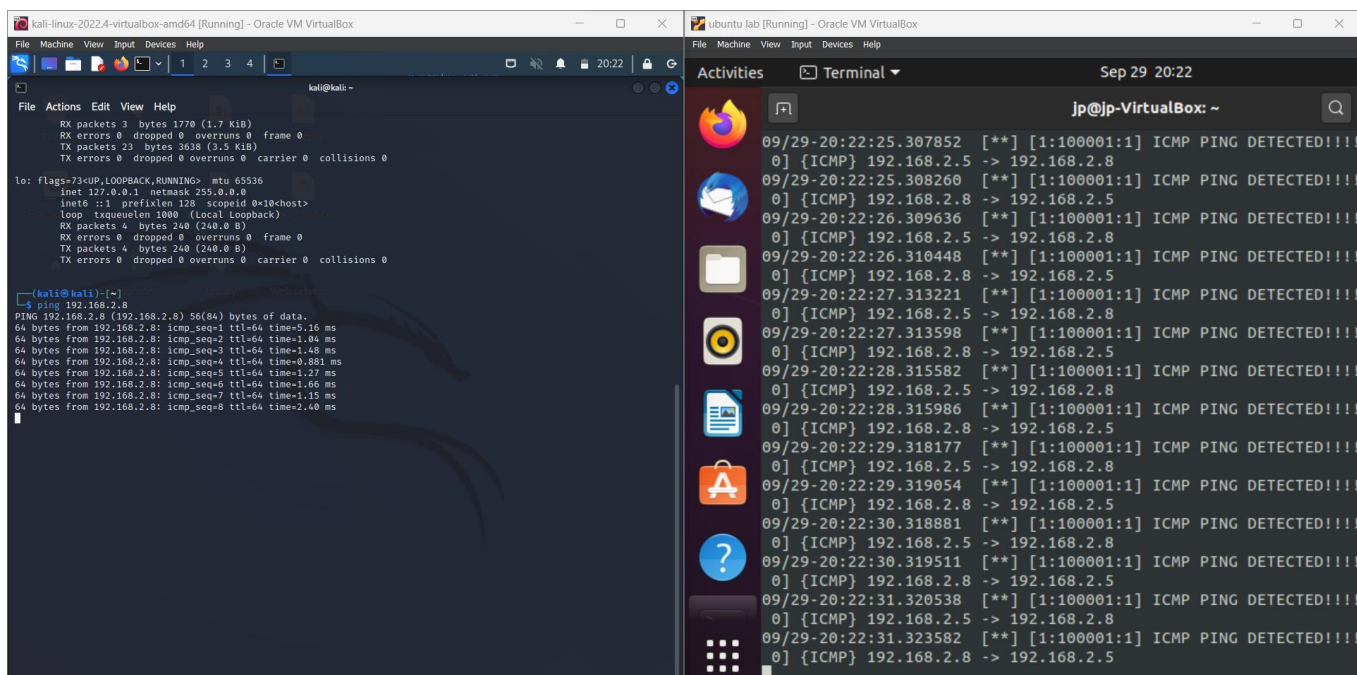
```

1 # $Id: local.rules,v 1.11 2004/07/23 20:15:44 bmc Exp $
2 # -----
3 # LOCAL RULES
4 # -----
5 # This file intentionally does not come with signatures. Put your local
6 # additions here.
7
8 alert icmp any any -> $HOME_NET any (msg: "ICMP PING DETECTED!!!!"; sid:100
  001; rev:1;)

```

Screenshot 4: Rule to detect ping requests.

In the local.rules file, I created an alert to detect any ping request done by an attacker machine as shown in the screenshot. Then I started the Snort IDS with the command `sudo snort -q -l /var/log/snort/ -i enp0s3 -A console -c /etc/snort/snort.conf`. Here I used `-q` for quiet mode, `-l /var/log/snort/` to log files in the specific directory, `-i enp0s3` for the interface `-A console` to display logs, `-c /etc/snort/snort.conf` to detect the configuration files. As shown in screenshot 5, the Ping requests sent from Kali to Ubuntu vuln were being detected by the Snort IDS.

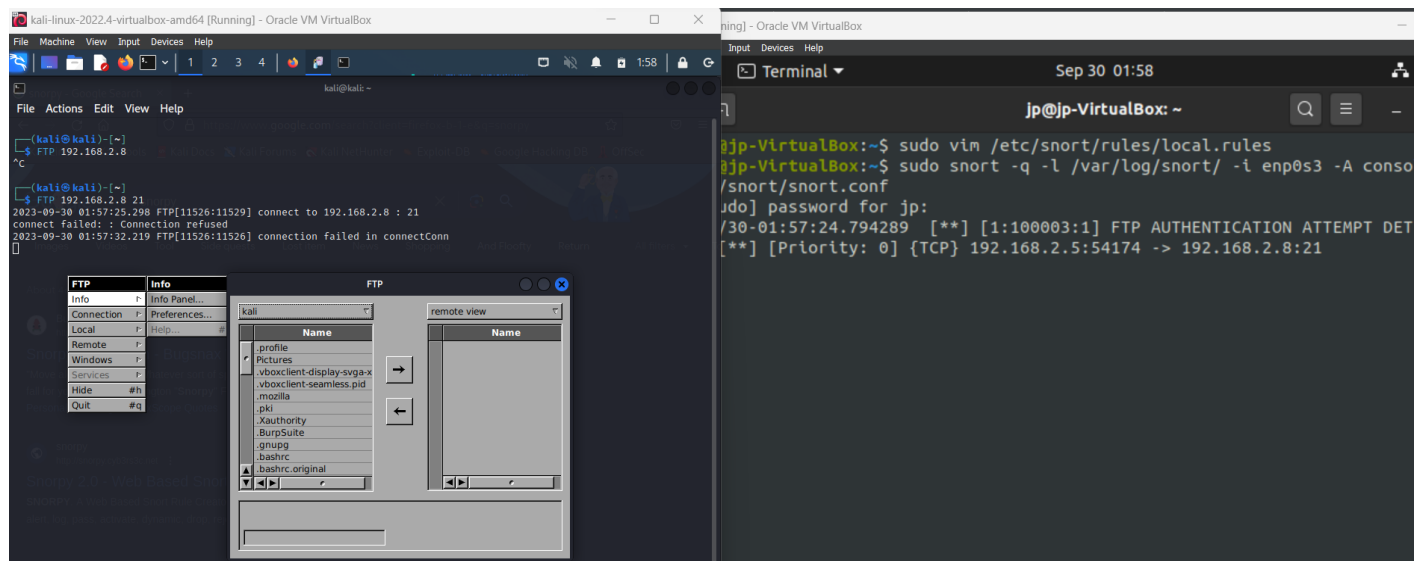


Screenshot 5: IDS alerting the ping requests on the right side.

The image displays two side-by-side screenshots of a Kali Linux virtual machine (VM) running on Oracle VM VirtualBox. The left screenshot shows a terminal window with the command `ssh ubuntu@192.168.2.8` and its output, including a warning about a new host key and a password prompt. The right screenshot shows the same terminal window with a successful SSH connection to the Ubuntu VM, displaying a standard Ubuntu login prompt.

Then, I configured an FTP authentication attempt detection rule with the addition of the following command to local.rules file, `alert TCP any any -> $HOME_NET 21 ( msg: "FTP AUTHENTICATION ATTEMPT DETECTED !!!"; std: 100003; rev:1; )`. After that I started the Snort IDS again and made an FTP attempt from kali machine to ubuntu vuln as shown in the screenshot 7. The Snort was able to capture the authentication attempt.





Screenshot 7: IDS detecting FTP authentication attempts on the right side.

After that, I configured Snort to detect an SQL injection attempt with the following rules:  
 command alert tcp any any -> any 80 (msg: "SQL Injection ATTEMPTED !!!!";  
 content: "%27" ; sid:100004; rev:1; )

alert tcp any any -> any 80 (msg: "SQL Injection ATTEMPTED !!!!"; content:  
 "%22" ; sid:100005; rev:1; ).

If I wanted to display my logs I would go to /var/log/snort and enter the command sudo  
 tcpdump -r snort.log.xxxxxxx as shown in screenshot 8.

```
jp@jp-VirtualBox:/var/log/snort$ sudo tcpdump -r snort.log.1696051835
reading from file snort.log.1696051835, link-type EN10MB (Ethernet)
01:31:17.577001 IP 192.168.2.5.34052 > 192.168.2.8.ssh: Flags [S], seq 549246447, win 64240, options [mss 1460,sackOK,TS val 3020345614 ecr 0,nop,wscale 7], length 0
01:31:17.577973 IP 192.168.2.5.34052 > 192.168.2.8.ssh: Flags [.], ack 2870310766, win 502, options [nop,nop,TS val 3020345616 ecr 4128442809], length 0
01:31:17.577974 IP 192.168.2.5.34052 > 192.168.2.8.ssh: Flags [P.], seq 0:35, ack 1, win 502, options [nop,nop,TS val 3020345616 ecr 4128442809], length 35
01:31:17.588018 IP 192.168.2.5.34052 > 192.168.2.8.ssh: Flags [.], ack 42, win 502, options [nop,nop,TS val 3020345626 ecr 4128442819], length 0
01:31:17.594455 IP 192.168.2.5.34052 > 192.168.2.8.ssh: Flags [P.], seq 1539:1587, ack 1098, win 501, options [nop,nop,TS val 3020345633 ecr 4128442824], length 48
01:31:17.642304 IP 192.168.2.5.34052 > 192.168.2.8.ssh: Flags [.], ack 1534, win 501, options [nop,nop,TS val 3020345679 ecr 4128442831], length 0
01:31:21.359761 IP 192.168.2.5.34052 > 192.168.2.8.ssh: Flags [P.], seq 1587:1603, ack 1534, win 501, options [nop,nop,TS val 3020349399 ecr 4128442831], length 16
01:31:21.404229 IP 192.168.2.5.34052 > 192.168.2.8.ssh: Flags [P.], seq 1603:1647, ack 1534, win 501, options [nop,nop,TS val 3020349443 ecr 4128446634], length 44
01:31:21.407410 IP 192.168.2.5.34052 > 192.168.2.8.ssh: Flags [.], ack 1578, win 501, options [nop,nop,TS val 3020349447 ecr 4128446637], length 0
01:31:21.409034 IP 192.168.2.5.34052 > 192.168.2.8.ssh: Flags [P.], seq 1647:1715, ack 1578, win 501, options [nop,nop,TS val 3020349448 ecr 4128446637], length 68
01:31:21.460158 IP 192.168.2.5.34052 > 192.168.2.8.ssh: Flags [.], ack 1630, win 501, options [nop,nop,TS val 3020349499 ecr 4128446649], length 0
```

*Screenshot 8: Displaying Snort IDS logs.*

## Suricata configuration and IDS

I have also configured suricata.yaml file as shown in screenshot 9

```

11 ##
12 ## Step 1: Inform Suricata about your network
13 ##
14
15 vars:
16   # more specific is better for alert accuracy and p
17   address-groups:
18     HOME_NET: "[192.168.2.0/24]"

```

*Screenshot 9: Updating subnet in suricata.yaml file.*

This completed the configuration of `suricata.yaml` file in `/etc/suricata/` directory. After that I started the suricata service with the command `sudo systemctl start suricata.service`. this can be verified as shown the screenshot 10.

```

jp@jp-VirtualBox:~$ curl http://testmyids.org/uid/index.html
uid=0(root) gid=0(root) groups=0(root)
jp@jp-VirtualBox:~$ sudo cat /var/log/suricata/fast.log
10/01/2023-16:17:23.257520  /**] [1:2013504:6] ET POLICY GNU/Linux APT User-Age
nt Outbound likely related to package management /**] [Classification: Not Susp
icious Traffic] [Priority: 3] {TCP} 192.168.2.4:55688 -> 91.189.91.39:80
10/01/2023-16:30:22.920777  /**] [1:2013028:7] ET POLICY curl User-Agent Outbou
nd /**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.16
8.2.4:44688 -> 13.249.190.56:80

```

*Screenshot 10: Testing suricata logging.*

Then, I switched off suricata to write custom rules with the command `sudo vim /etc/suricata/rules/local.rules`. After that I wrote the same ICMP and SSH detection rules I used for Snort. After that, I added the `local.rules` patch to `suricata.yaml` file as shown in screenshot 11



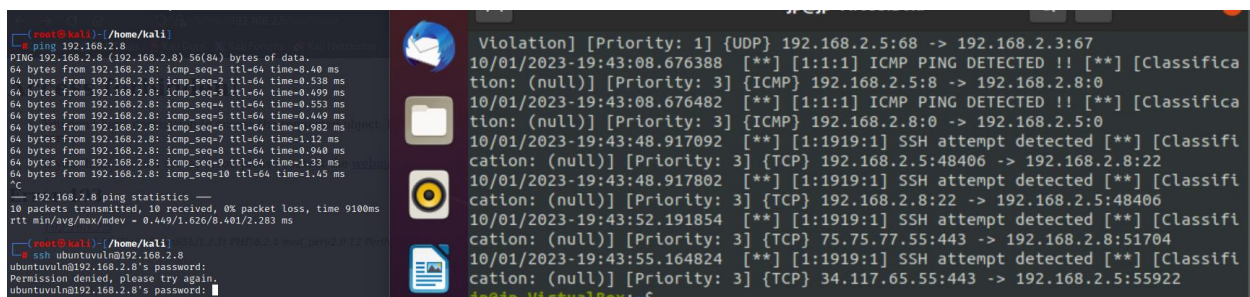
```
##
## Configure Suricata to load Suricata-Upda
##

default-rule-path: /var/lib/suricata/rules

rule-files:
- suricata.rules
- /etc/suricata/rules/local.rules
##
## Auxiliary configuration files.
##
```

Screenshot 11: Adding local.rules path to suricata.yaml File.

As shown in screenshot 12, the ICMP and SSH login attempts were logged by Suricata.



```
(root@kali) ~/home/kali
ping 192.168.2.8
PING 192.168.2.8 (192.168.2.8) 56(84) bytes of data:
64 bytes from 192.168.2.8: icmp_seq=1 ttl=64 time=0.40 ms
64 bytes from 192.168.2.8: icmp_seq=2 ttl=64 time=0.538 ms
64 bytes from 192.168.2.8: icmp_seq=3 ttl=64 time=0.499 ms
64 bytes from 192.168.2.8: icmp_seq=4 ttl=64 time=0.552 ms
64 bytes from 192.168.2.8: icmp_seq=5 ttl=64 time=0.449 ms
64 bytes from 192.168.2.8: icmp_seq=6 ttl=64 time=0.982 ms
64 bytes from 192.168.2.8: icmp_seq=7 ttl=64 time=1.12 ms
64 bytes from 192.168.2.8: icmp_seq=8 ttl=64 time=0.948 ms
64 bytes from 192.168.2.8: icmp_seq=9 ttl=64 time=1.33 ms
64 bytes from 192.168.2.8: icmp_seq=10 ttl=64 time=1.45 ms
^C
--- 192.168.2.8 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9100ms
rtt min/avg/max/mdev = 0.449/1.626/6.401/2.283 ms

(root@kali) ~/home/kali
ssh ubuntu@192.168.2.8
ubuntu@192.168.2.8's password:
Permission denied, please try again.
ubuntu@192.168.2.8's password:

Violation] [Priority: 1] {UDP} 192.168.2.5:68 -> 192.168.2.3:67
10/01/2023-19:43:08.676388 [**] [1:1:1] ICMP PING DETECTED !! [**] [Classifica
tion: (null)] [Priority: 3] {ICMP} 192.168.2.5:8 -> 192.168.2.8:0
10/01/2023-19:43:08.676482 [**] [1:1:1] ICMP PING DETECTED !! [**] [Classifica
tion: (null)] [Priority: 3] {ICMP} 192.168.2.8:0 -> 192.168.2.5:0
10/01/2023-19:43:48.917092 [**] [1:1919:1] SSH attempt detected [**] [Classifi
cation: (null)] [Priority: 3] {TCP} 192.168.2.5:48406 -> 192.168.2.8:22
10/01/2023-19:43:48.917802 [**] [1:1919:1] SSH attempt detected [**] [Classifi
cation: (null)] [Priority: 3] {TCP} 192.168.2.8:22 -> 192.168.2.5:48406
10/01/2023-19:43:52.191854 [**] [1:1919:1] SSH attempt detected [**] [Classifi
cation: (null)] [Priority: 3] {TCP} 75.75.77.55:443 -> 192.168.2.8:51704
10/01/2023-19:43:55.164824 [**] [1:1919:1] SSH attempt detected [**] [Classifi
cation: (null)] [Priority: 3] {TCP} 34.117.65.55:443 -> 192.168.2.5:55922
4n@in-VirtualBox:~$
```

Screenshot 12: Suricata detecting ICMP and SSH pings.

## Wazuh configuration

I setup wazuh in Linode which was a cloud server setup tool that contained lot of technologies and tools like Splunk, Ubuntu, windows etc. After creation of Wazuh server in linode I copied the installation and pasted in the suricata Ubuntu I set up earlier as shown in screenshot 13.

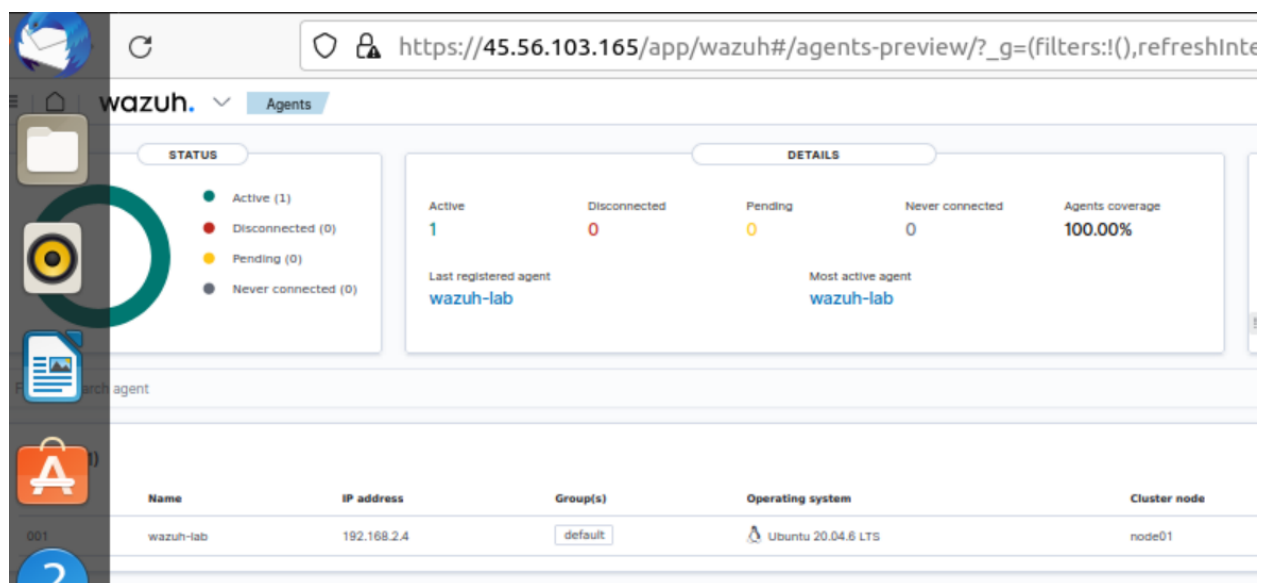
```

jp@jp-VirtualBox:~$ curl -so wazuh-agent.deb https://packages.wazuh.com/4.x/apt
/pool/main/w/wazuh-agent/wazuh-agent_4.5.2-1_amd64.deb && sudo WAZUH_MANAGER='4
5.56.103.165' WAZUH_AGENT_GROUP='default' WAZUH_AGENT_NAME='wazuh-lab' dpkg -i
./wazuh-agent.deb
[sudo] password for jp:
Selecting previously unselected package wazuh-agent.
(Reading database ... 190056 files and directories currently installed.)
Preparing to unpack ./wazuh-agent.deb ...
Unpacking wazuh-agent (4.5.2-1) ...
Setting up wazuh-agent (4.5.2-1) ...
Processing triggers for systemd (245.4-4ubuntu3.22) ...

```

*Screenshot 13: Installing Wazuh in ubuntu machine though Linode.*

After that I started wazuh-agent as shown in screenshot 14 and it was working perfectly.



*Screenshot 14: ubuntu IDS machine added as agent in Wazuh.*

Then I added eve.json file to configuration file to import log information from suricata as shown in screenshot 15.

## < Manager configuration

Refresh

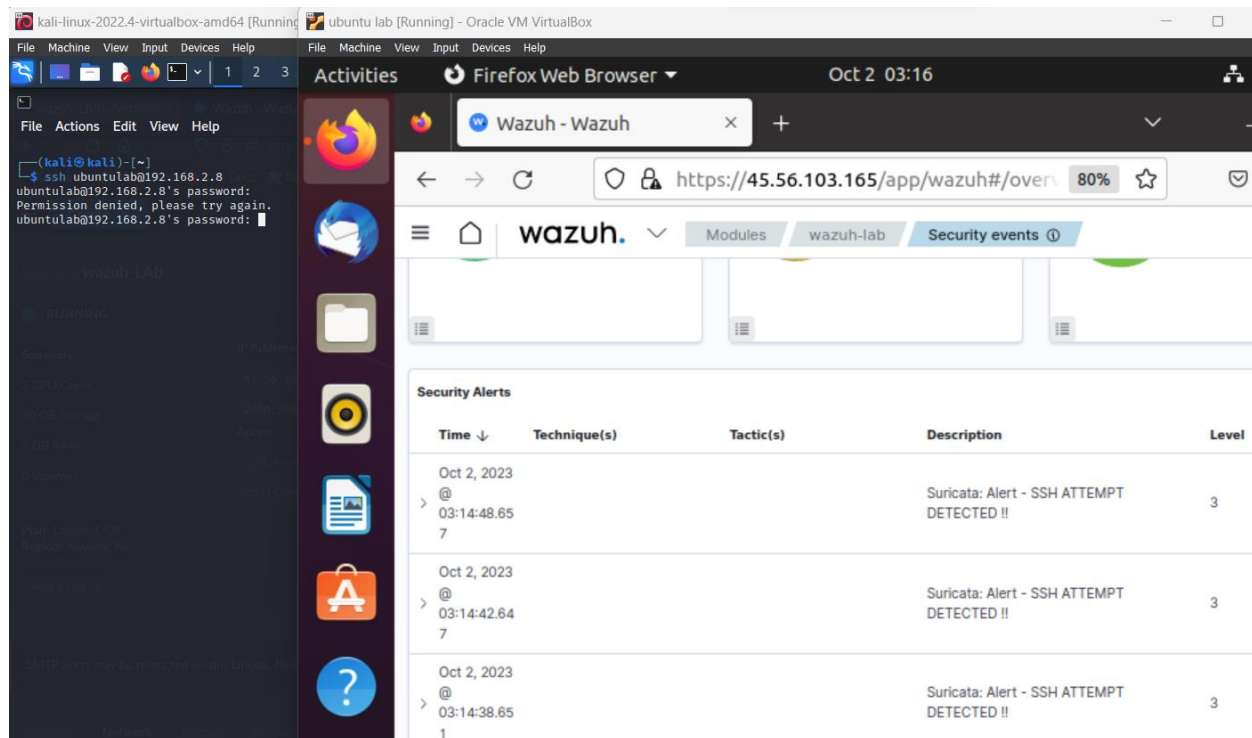
Save

### Edit **ossec.conf** of **Manager**

```
386
387 <localfile>
388 |   <log_format>syslog</log_format>
389 |   <location>/var/log/syslog</location>
390 </localfile>
391
392 <localfile>
393 |   <log_format>syslog</log_format>
394 |   <location>/var/log/dpkg.log</location>
395 </localfile>
396
397 <localfile>
398 |   <log_format>syslog</log_format>
399 |   <location>/var/log/kern.log</location>
400 </localfile>
401
402 <localfile>
403 |   <log_format>syslog</log_format>
404 |   <location>/var/log/suricata/eve.json</location>
405 </localfile>
406
407 </ossec_config>
408
```

Screenshot 15: Displaying Snort IDS logs.

That completed the configuration of Wazuh. In the next step, I made SSH connection request from kali machine to ubuntu vuln machine and Wazuh was able to log this information as shown in screenshot 16.



*Screenshot 16: Wazuh logging the SSH attempts through Suricata IDS.*

## Conclusion

This concludes the demonstration of Intrusion Detection System implementation with Snort, Suricata and integration of Suricata with Wazuh for log processing. Wazuh tool is a tool similar to Splunk which can be used for log analysis, File integrity monitoring, vulnerability detection, IDS/IPS.