IDS and log processing demo with Snort, Suricata, Wazuh

Executive Summary

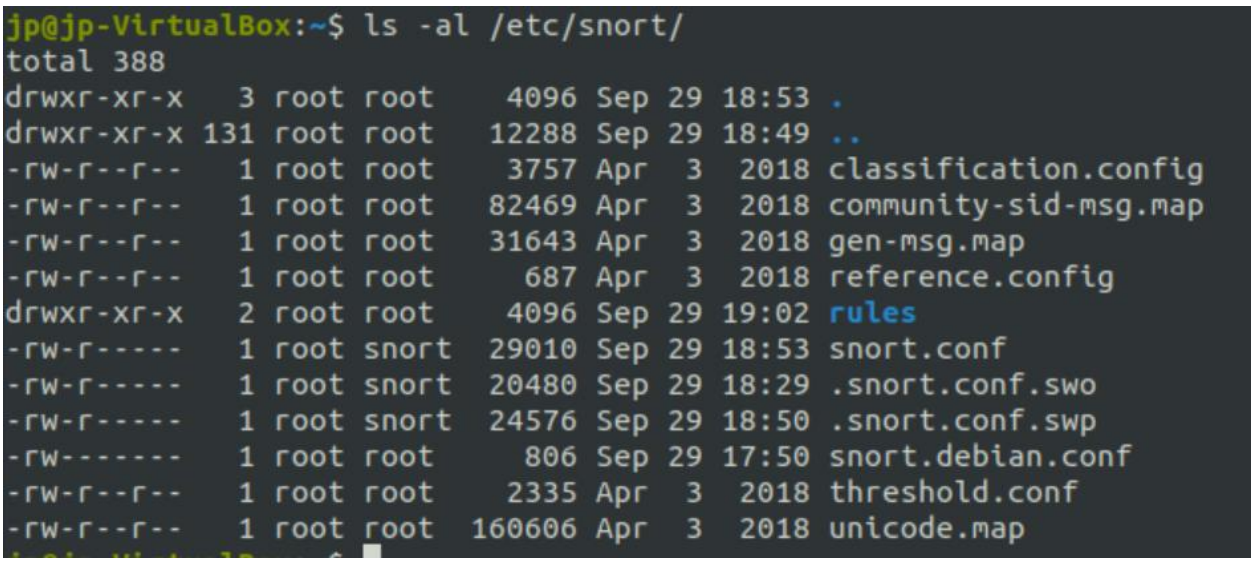
The objective of the project was to establish an intrusion detection system laboratory with the assistance of the Snort, Suricata and the integration of Wazuh with Suricata. The project aimed to gain an understanding of how rules were written for these tools, how they captured incoming traffic, how the 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.



Screenshot 1: Contents of /etc/snort.

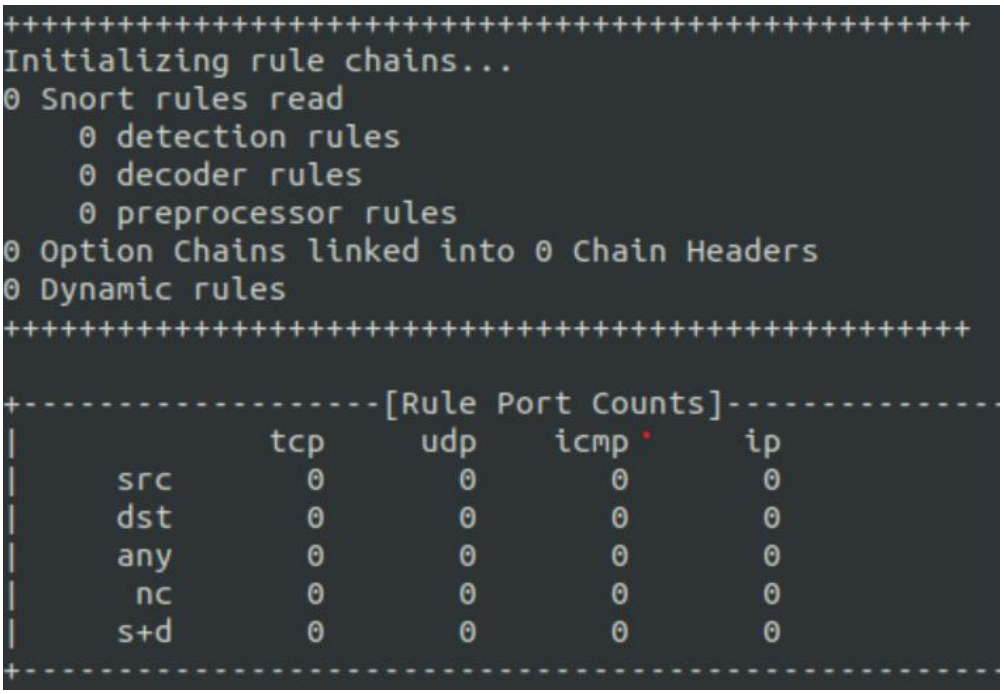
In the next step I have modified the `snort.conf` file that had `HOME_NET` to `192.168.2.0/24` which was the subnet of my VM as shown in screenshot 2.

```
# Setup the network addresses you are protecting
#
# Note to Debian users: this value is overriden 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.



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.

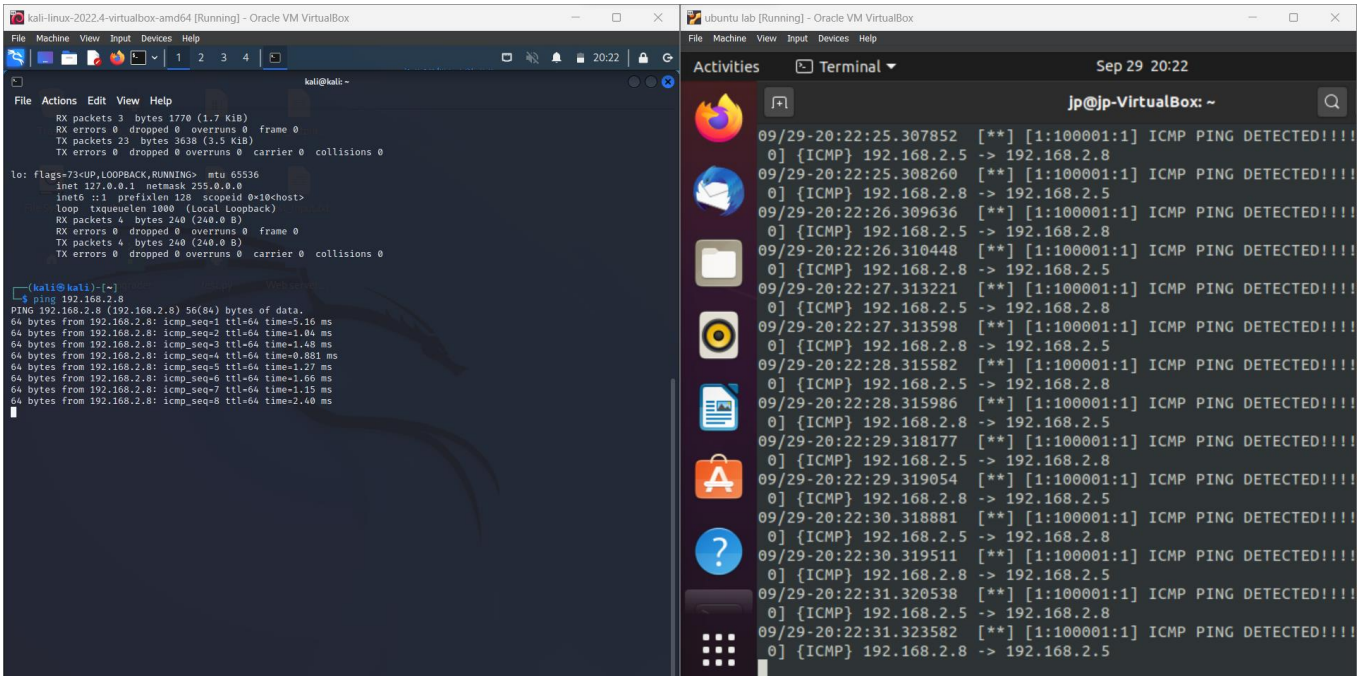
```
# $Id: local.rules,v 1.11 2004/07/23 20:15:44 bmc Exp $
# ---------------
# LOCAL RULES
# ---------------
# This file intentionally does not come with signatures.  Put your local
# additions here.

alert icmp any any -> $HOME_NET any (msg: "ICMP PING DETECTED!!!!"; sid:100
001; rev:1;)
```

Screenshot 4: Rule to detect ping requests.

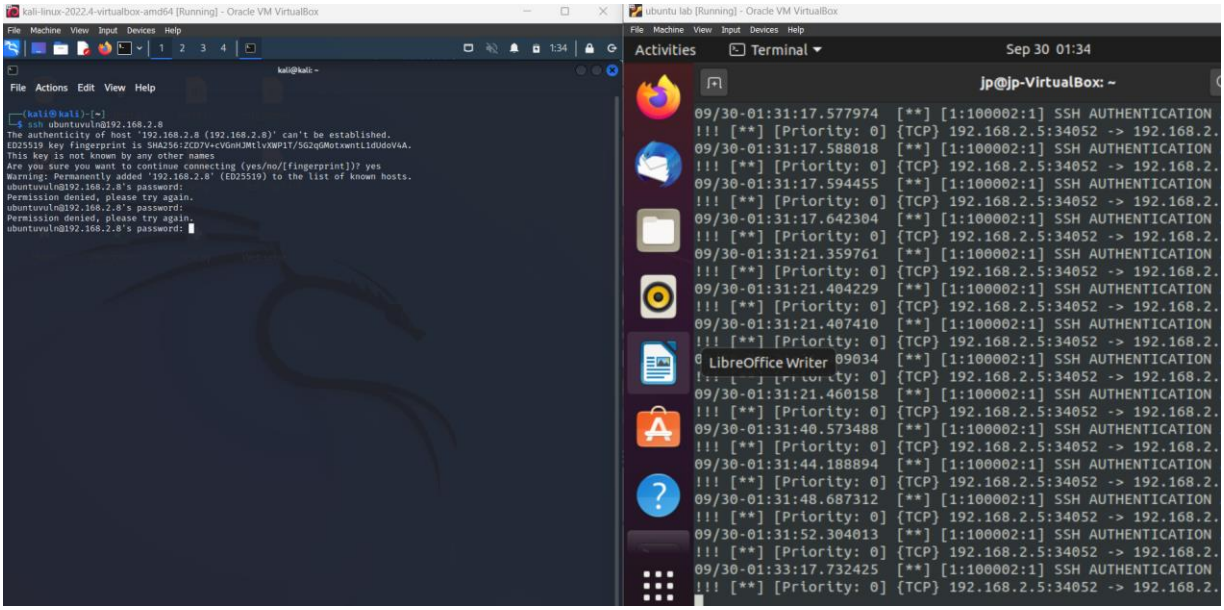
In the local.`rules` file, I created an alert to detect any ping request done by an attacker machine as shown in the screenshot. Then I started the Snort IDS with the command `sudo snort -q -l /var/log/snort/ -i enp0s3 -A console -c /etc/snort/snort.conf` . Here I used

-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.



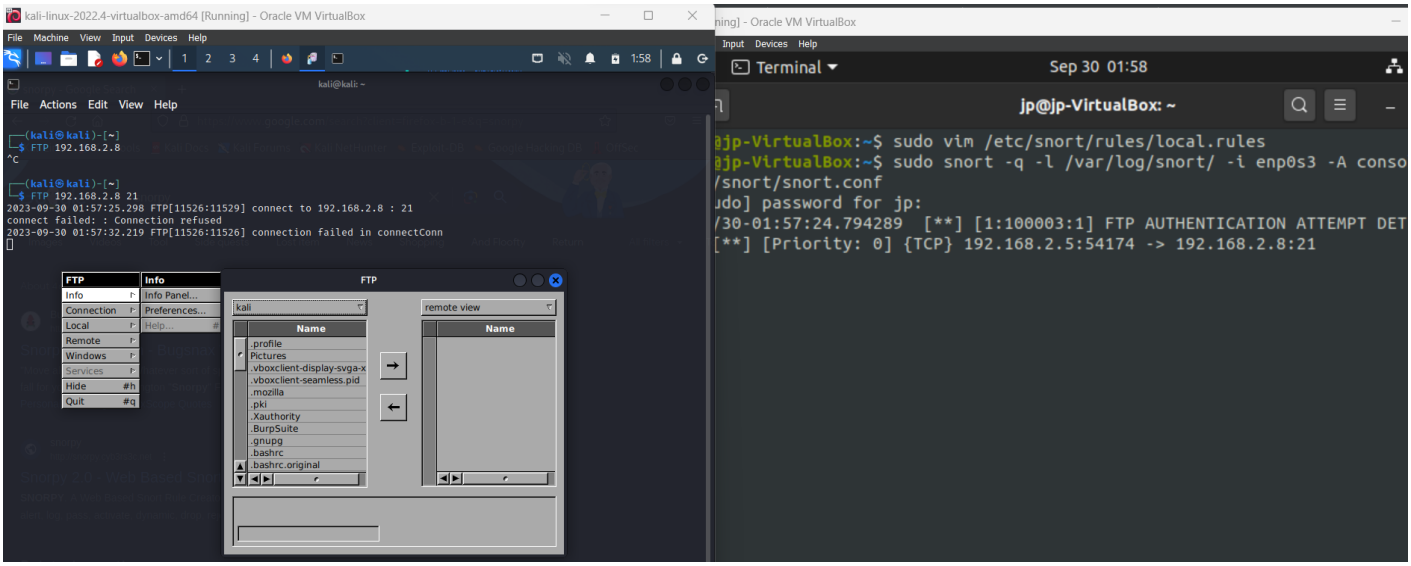
Screenshot 5: IDS alerting the ping requests on the right side.

After that I configured an SSH authentication attempt detection rule with the addition of the following command to local.rules file, `alert TCP any any -> $HOME_NET 22 ( msg: ”SSH AUTHENTICATION ATTEMPT DETECTED !!!!!”; std: 100002; rev:1;)` . After that I started the Snort IDS again and made an SSH attempt from kali machine to ubuntu vuln as shown in the screenshot 6. The Snort was able to capture the authentication attempt.



Screenshot 6: IDS alerting the SSH login attempts on the right side.

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.

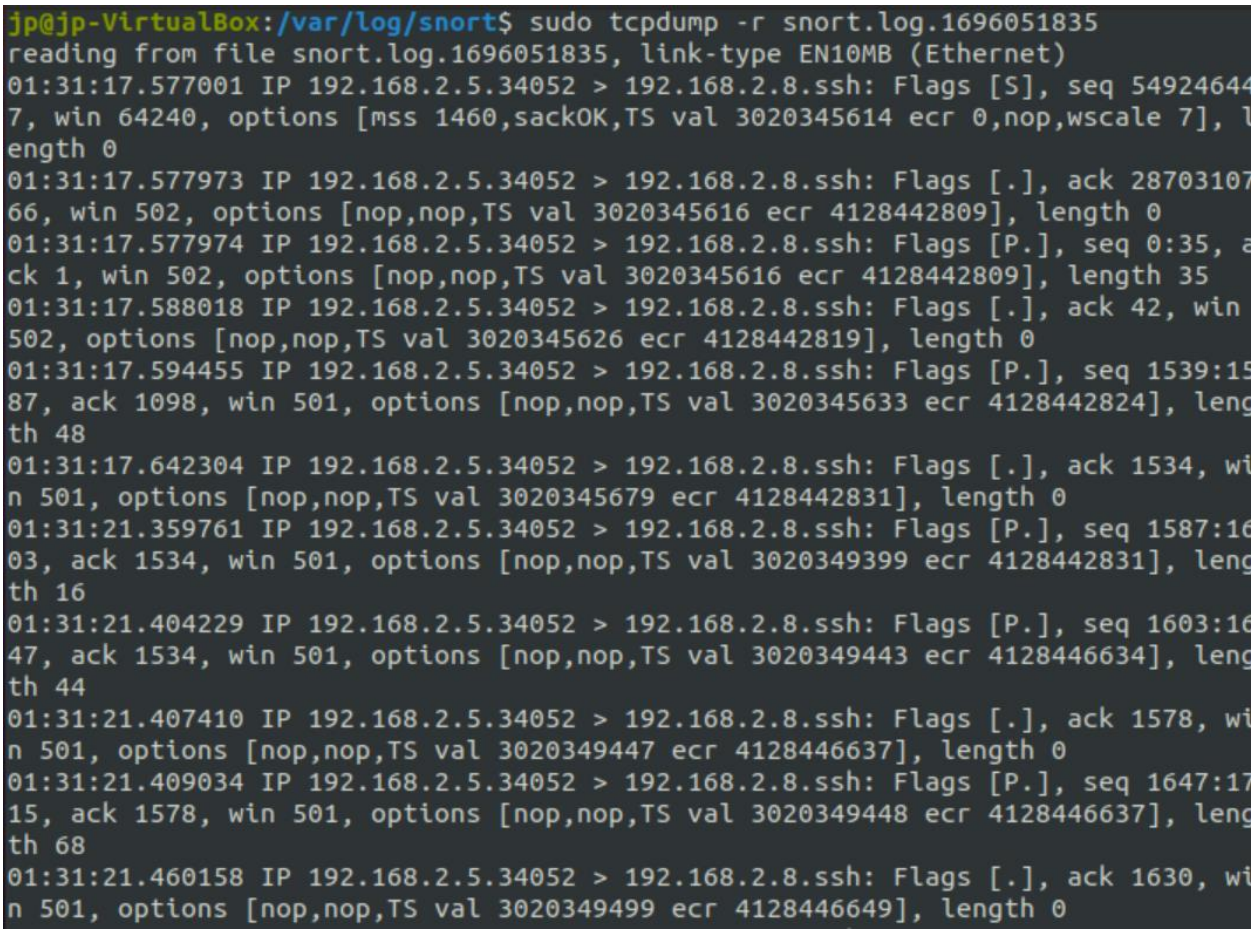


Screenshot 7: IDS detecting FTP authentication attempts on the right side.

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; ).`

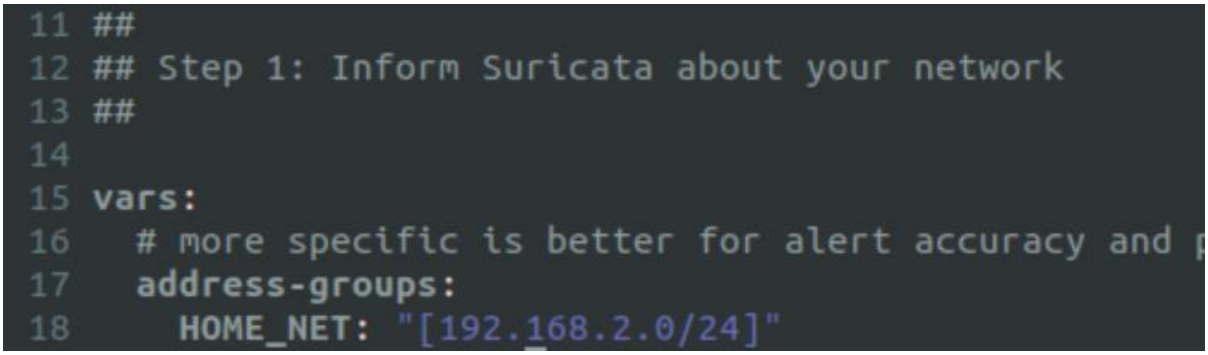
If I wanted to display my logs I would go to `/var/log/snort` and enter the command `sudo tcpdump -r snort.log.xxxxxxx` as shown in screenshot 8.



Screenshot 8: Displaying Snort IDS logs.

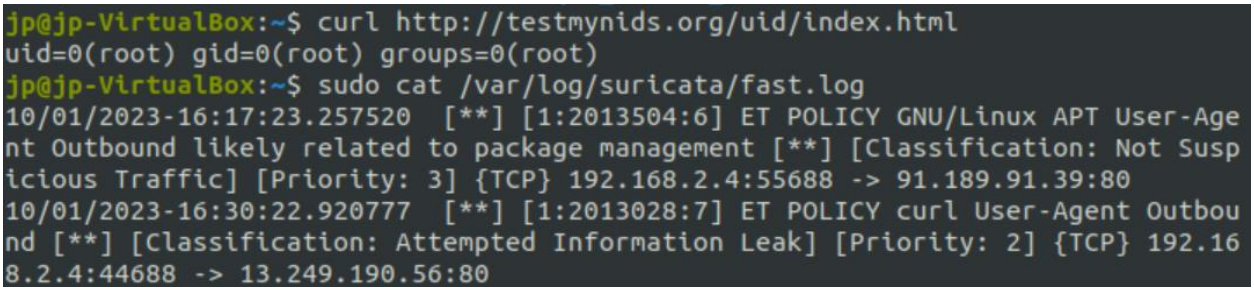
Suricata configuration and IDS

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



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.



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

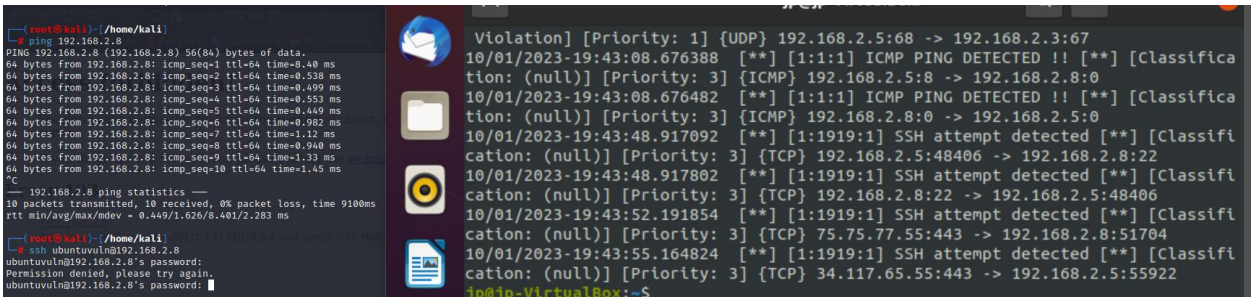
```
##
## Configure Suricata to load Suricata-Upda
##

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.



Screenshot 12: Suricata detecting ICMP and SSH pings.

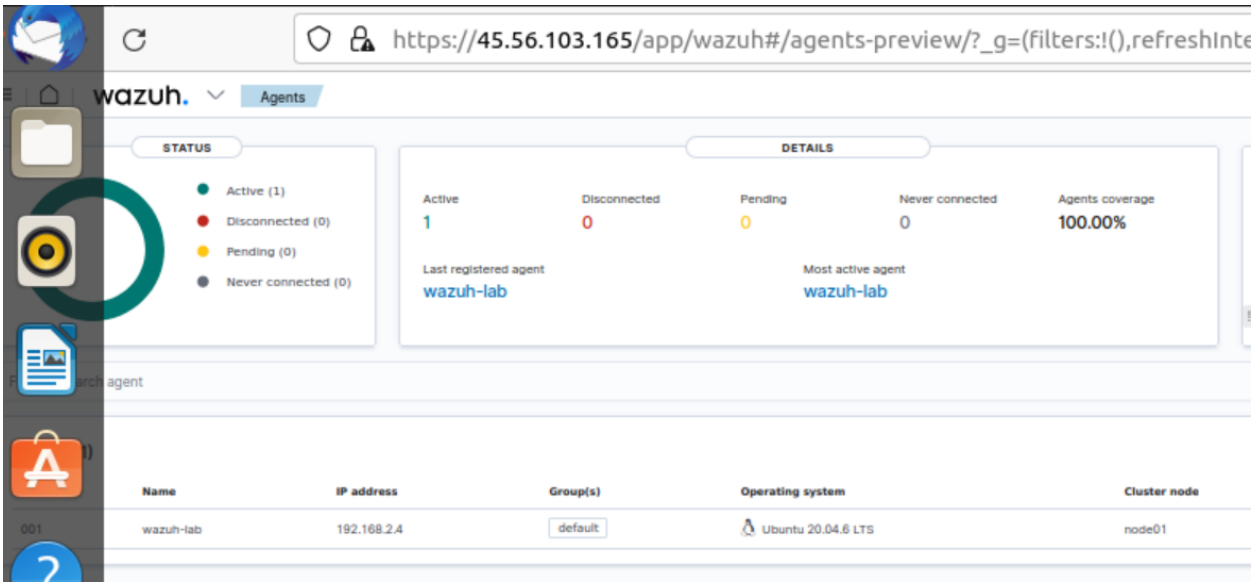
Wazuh configuration

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.

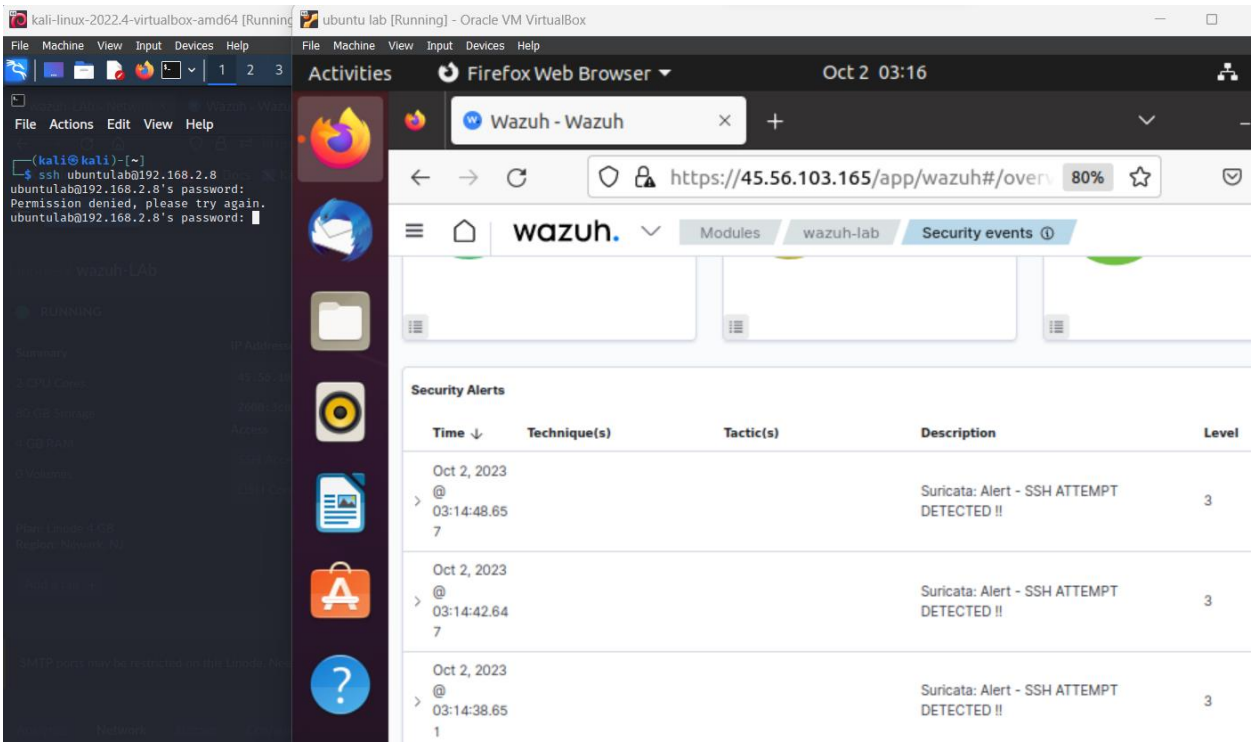


Screenshot 14: ubuntu IDS machine added as agent in Wazuh.

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

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 ubuntuvuln 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.