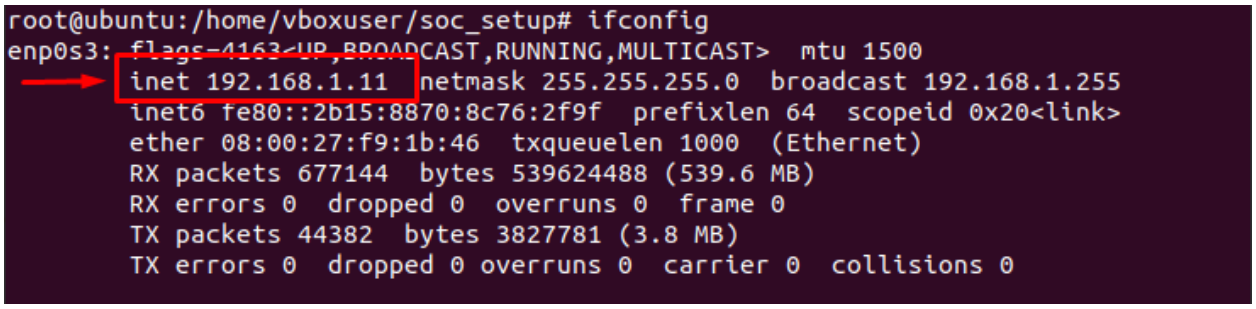
Installation and use of SIEM, HIDS & NIDS

Configuration & Environment Information:

For this demonstration I am using 2 Physical devices. On my windows 10 there are 2 VM on the virtual box (kali & ubuntu). The 2 VM's are on the Bridge adapter and the total network sharing IP's from the same block 192.169.1.0/24. Config of the devices & use cases are given below.

Ubuntu(Separate PC):

Name: vboxuser@ubuntu
IP: 192.168.1.11
Purpose: install a SIEM (Elasticsearch, Kibana and Filebeat) , HIDS (Wazuh Manager) and NIDS (Suricata)

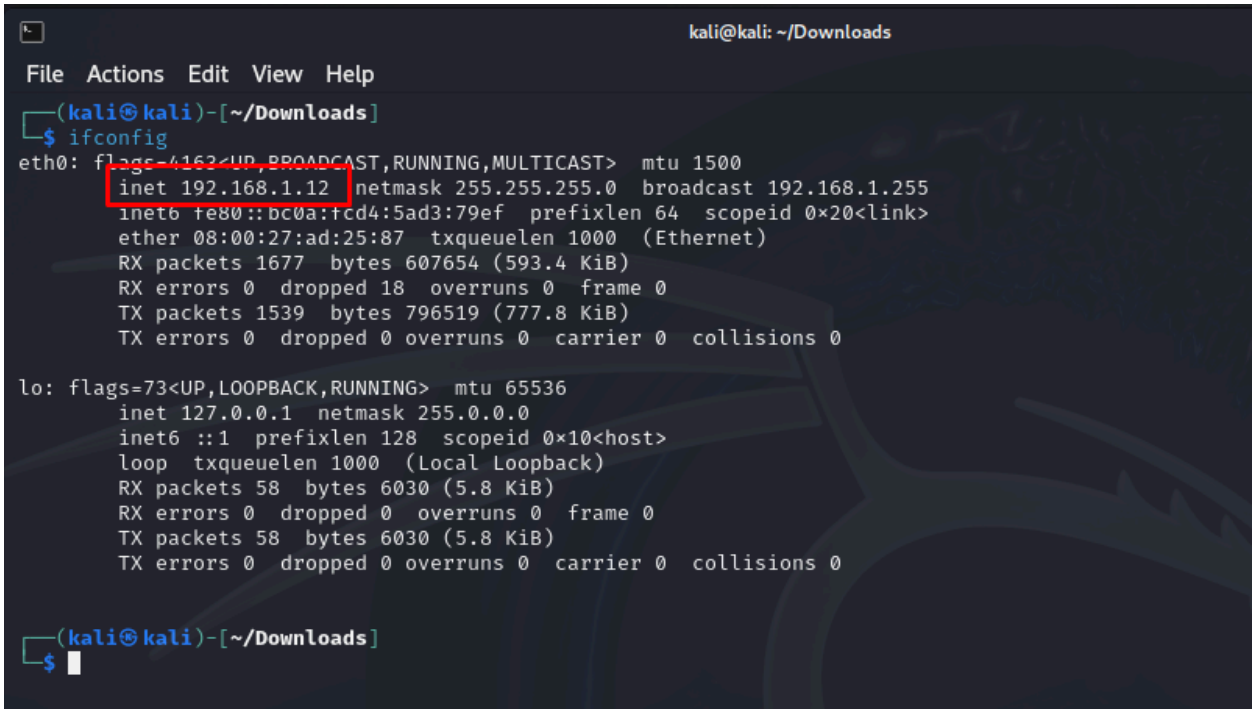


VM1 (Kali VM running on Virtual Box):

Name: Kali

IP: 192.168.1.12

Purpose: Attacking VM2

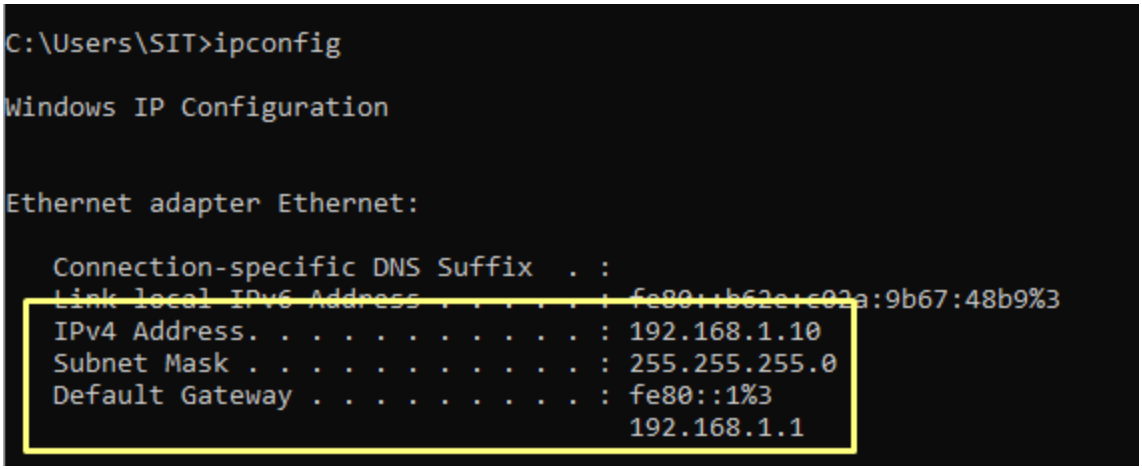


Windows 10 (PC):

Name: SIT

IP: 192.168.1.10

Purpose: Wazuh Agent (id: 001)



Let's proceed with the installation

To set up a SIEM solution for comprehensive log analysis, the implementation involves the utilization of specific tools like, Elasticsearch, Kibana, and Filebeat. Furthermore, The SIEM setup includes Elasticsearch, Kibana and Filebeat version 7.17.13 as it is the compatible version to integrate with our HIDS Wazuh manager version 4.7.4 I am installing it in /otp The procedural steps for this deployment are encapsulated within a script accessible at the following repository:

Install Git & Curl first:

```
sudo apt install git
```

```
sudo apt install curl
```

Navigate to otp:

```
cd /opt
```

Script execution:

1. Clone this repository to your local machine:

```
Git  clone https://github.com/samiul008ghub/sour_setup/
```

2. Navigate to the repository's directory:

```
cd soc_setup
```

3. Make the setup_script.sh executable:

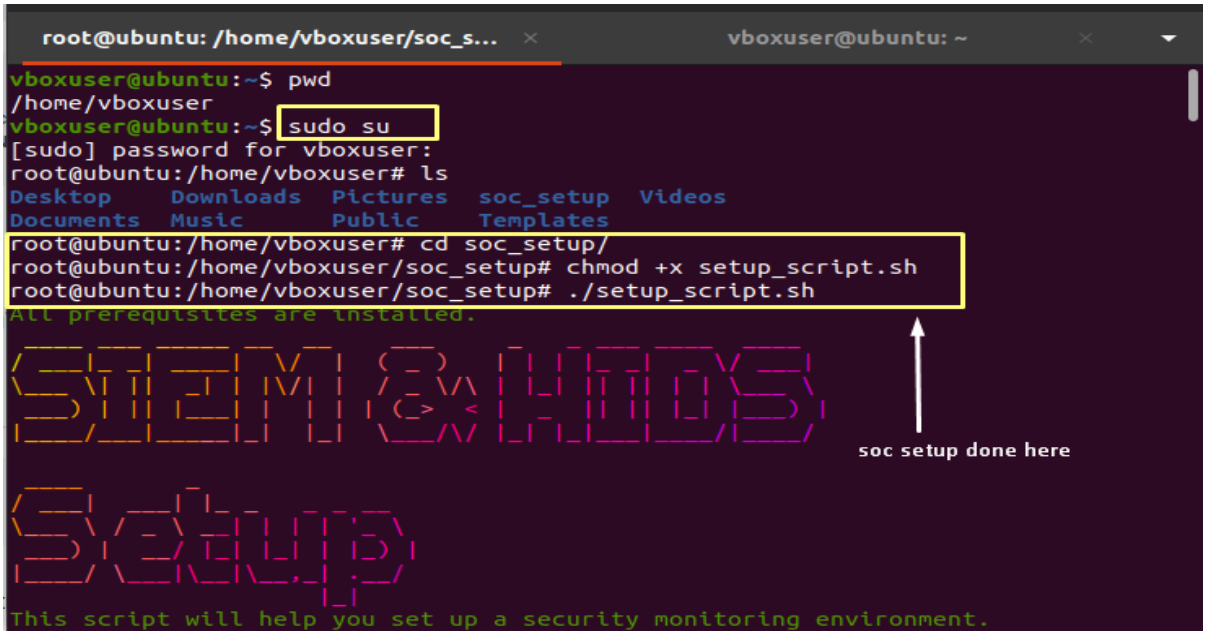
```
chmod +x setup_script.sh
```

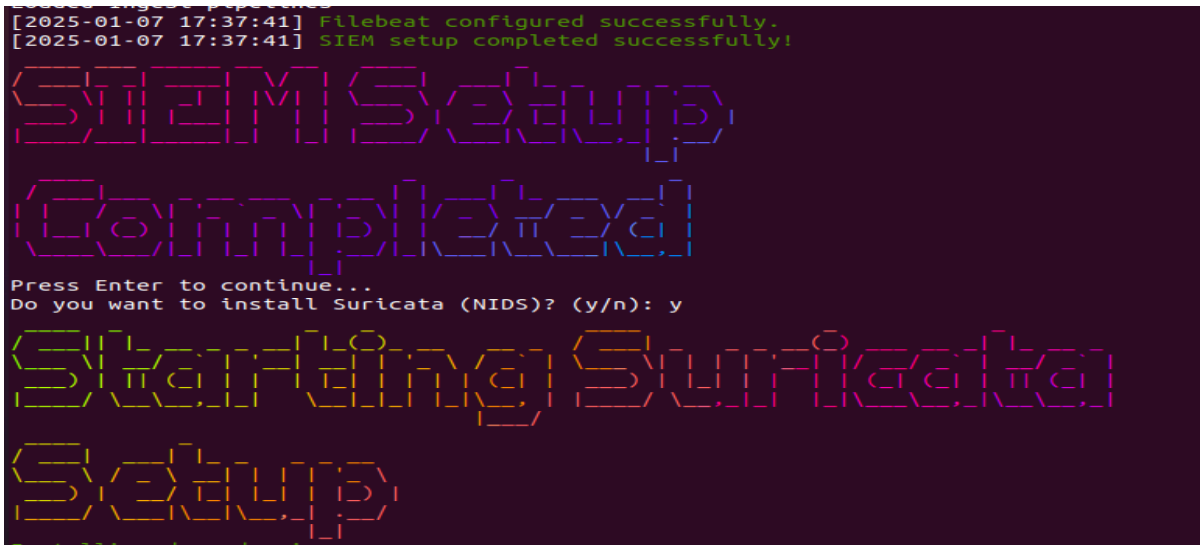
4. Execute the setup_script.sh”

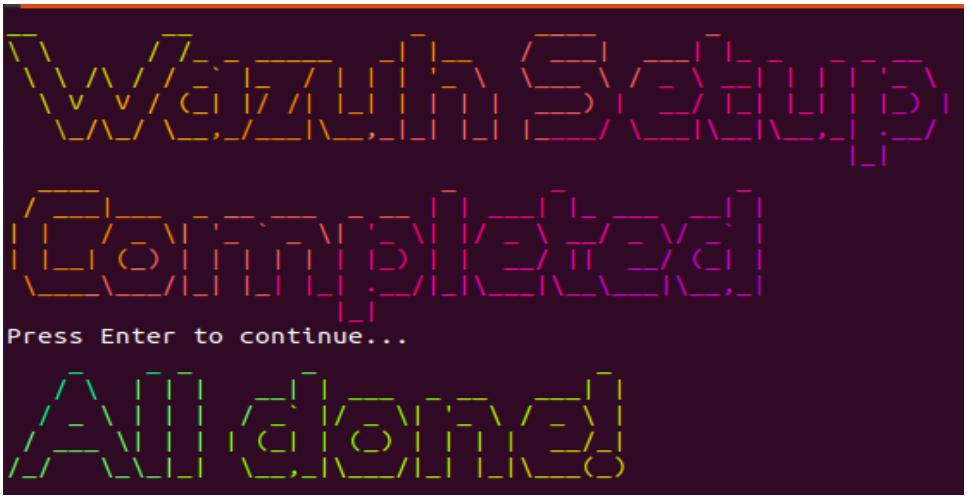
```
./setup_script.sh
```

5. Follow the on-screen prompts to choose which components you want to install and continue with the setup. Post-Installation Steps

Related Screenshots for installation:





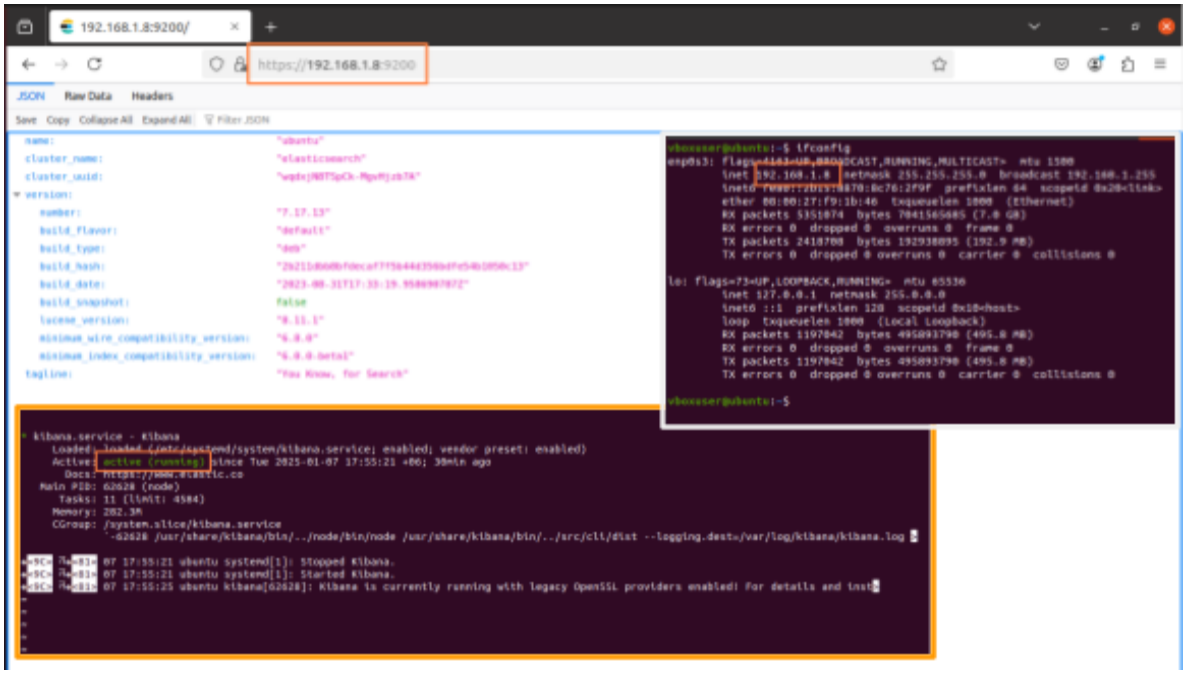


Setup Completion

SIEM

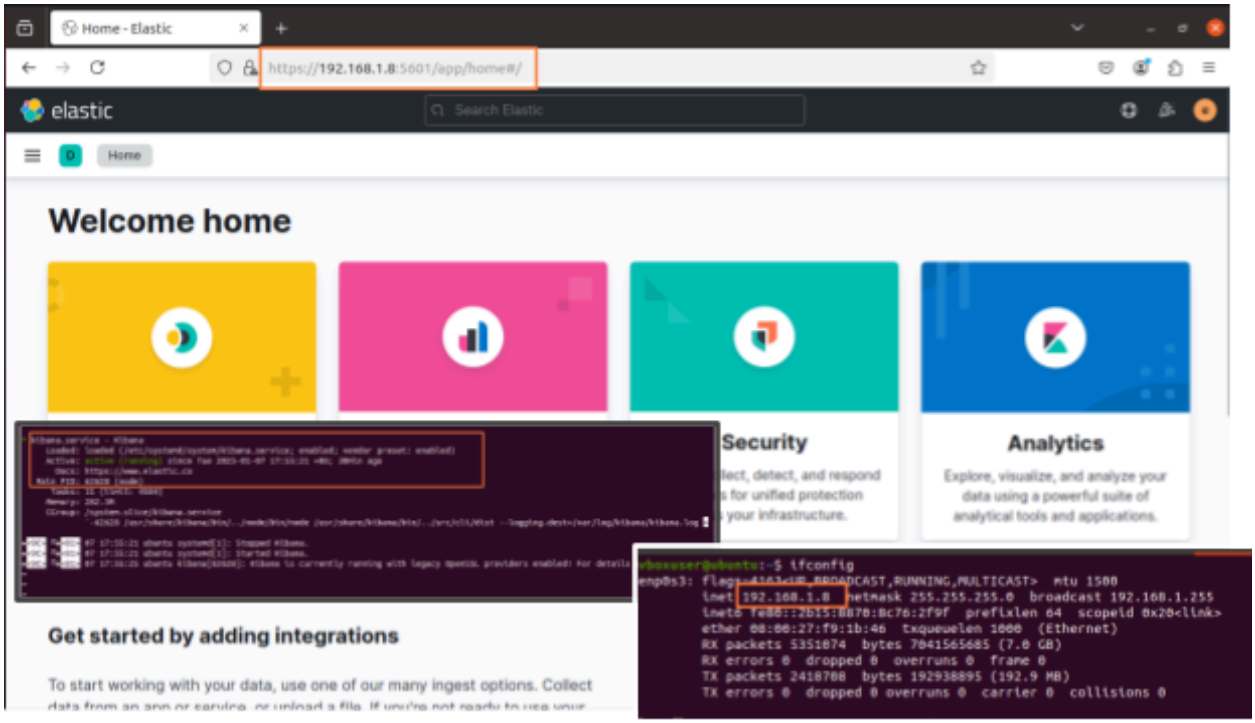
Elasticsearch installation proof

Provided screenshot from browser with ip address and port number.



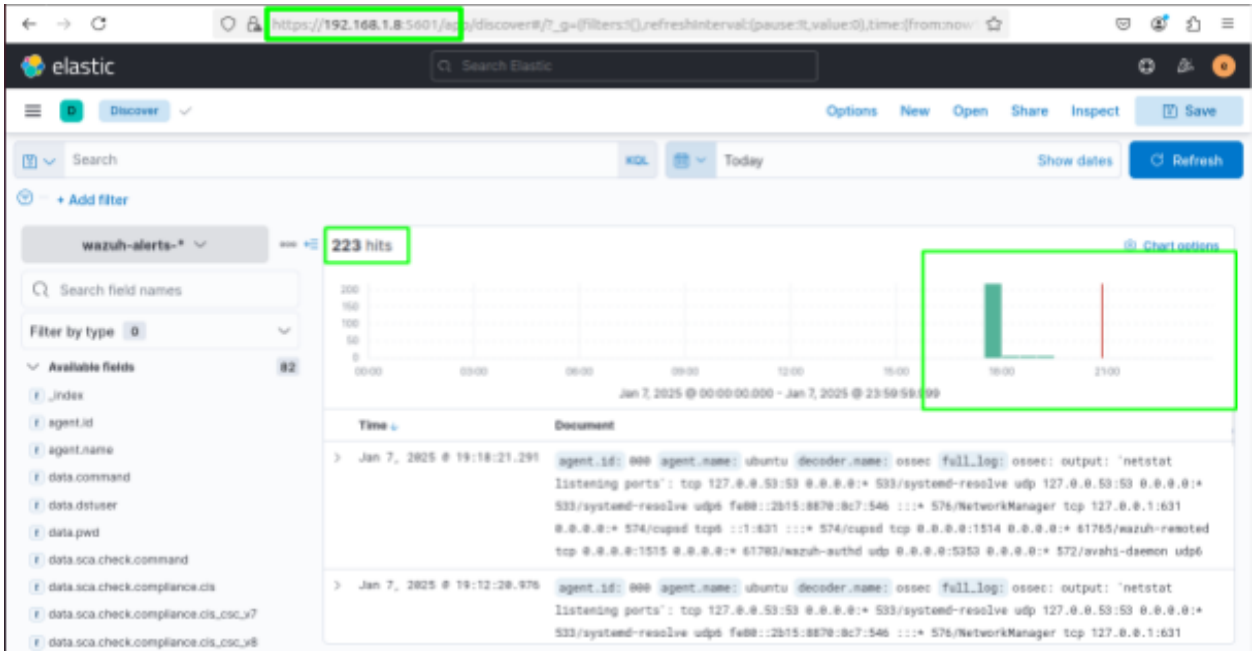
Kibana Installation Proof

Provided screenshot from browser with ip address and port number.



Filebeat Installation Proof

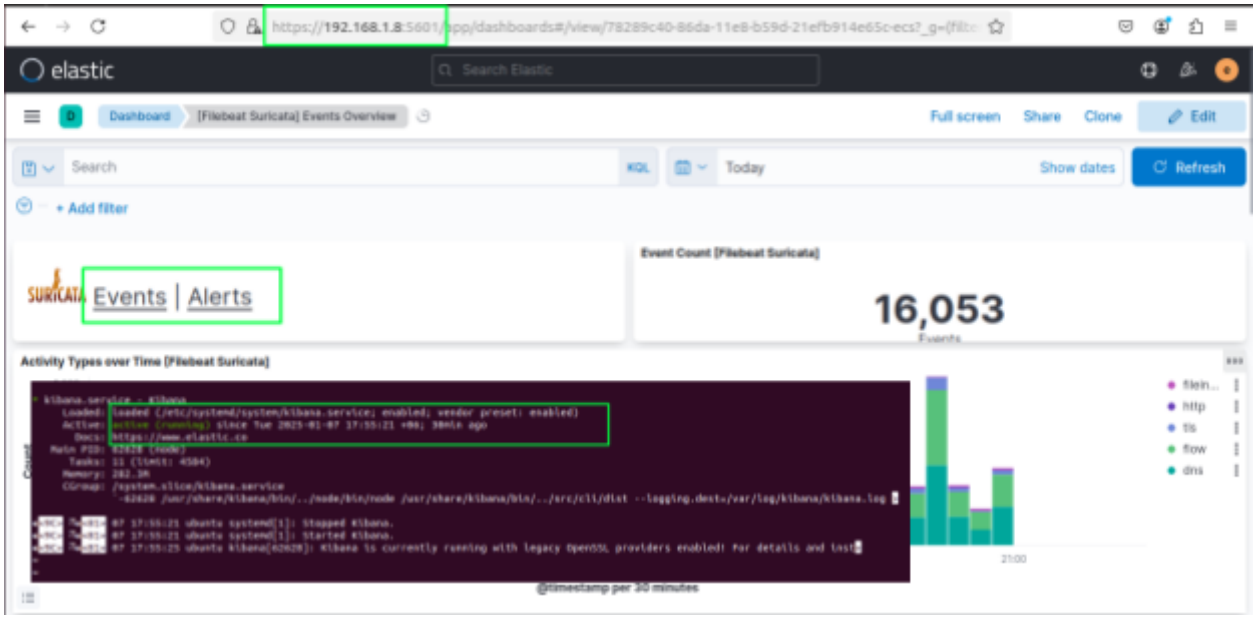
Provide a screenshot from the kibana dashboard which should have logs displayed.



NIDS

Suricata Installation Proof

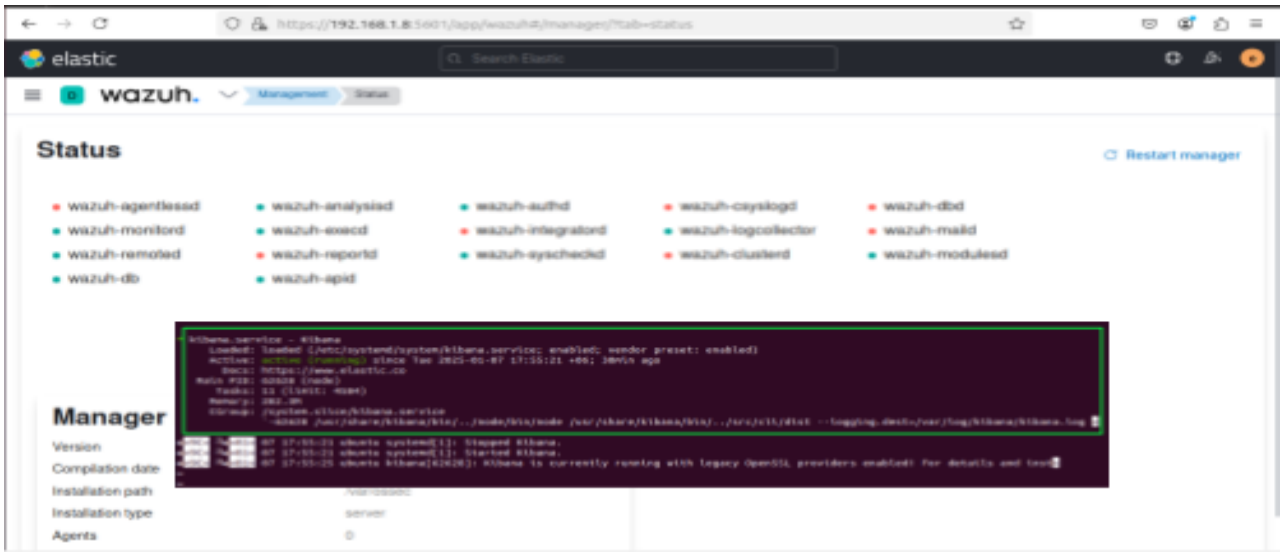
Provide screenshot of Events/Alerts from kibana dashboard of suricata.



HIDS

Wazuh Manager Installation Proof

Provide a screen shot of the wazuh dashboard from the kibana console. It should have all components listed as zero as no agents were not integrated



DLL Hijacking

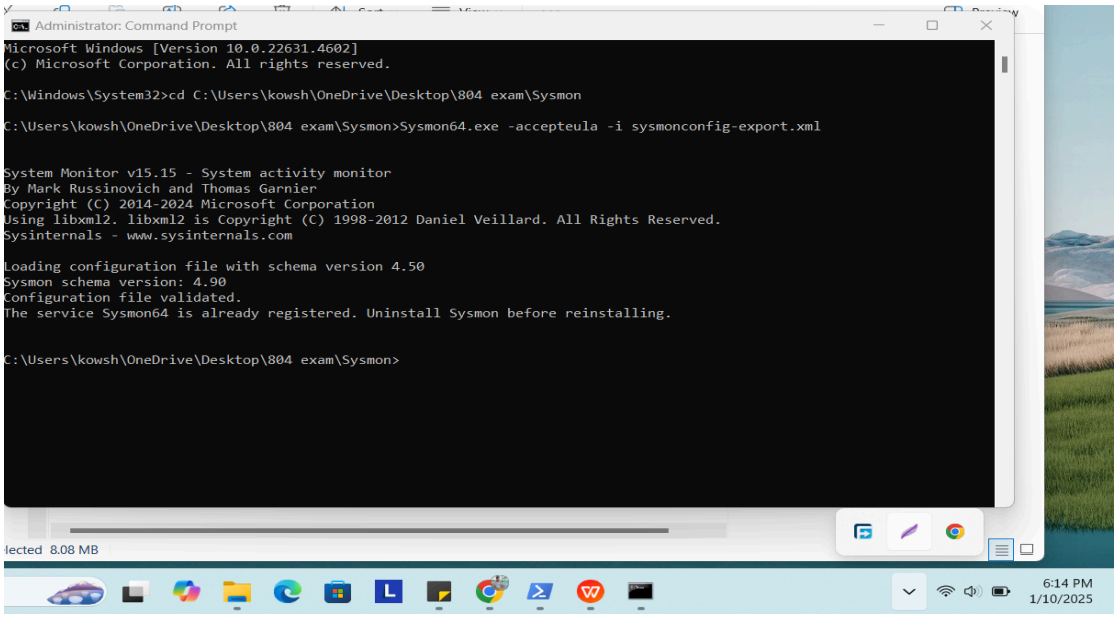
We done this on Machine [Windows-10] [10.42.0.238]

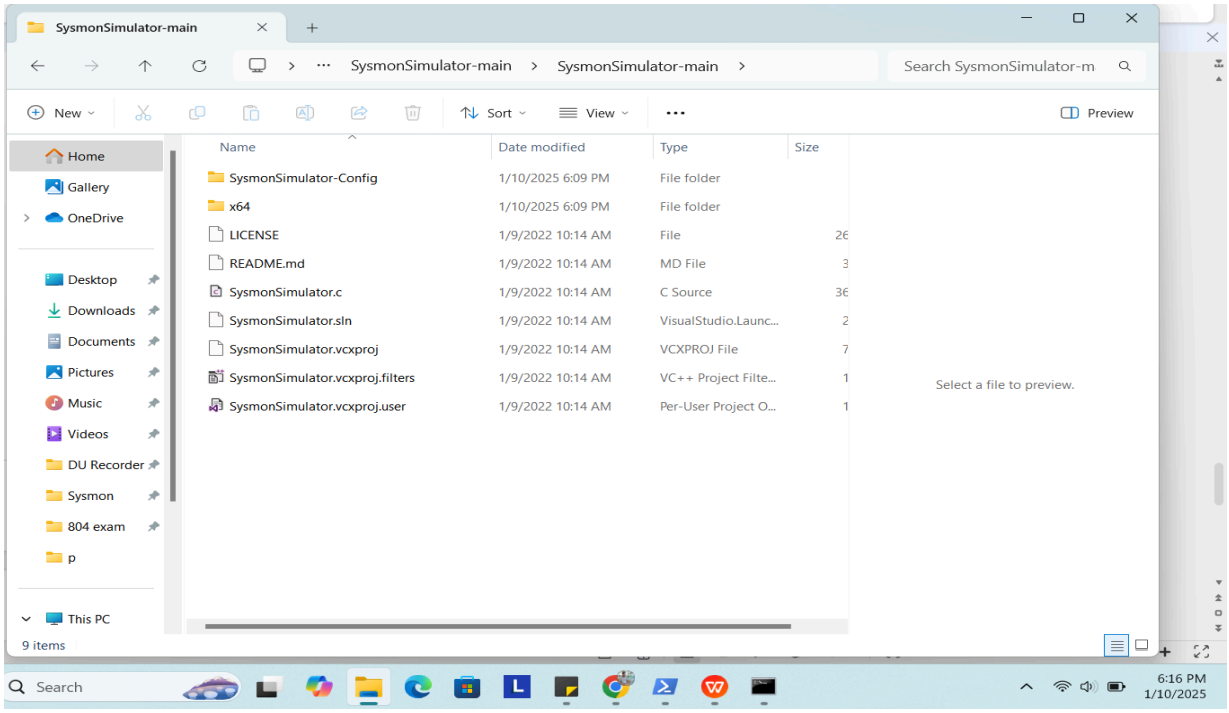
In essence, DLL injection is a common technique employed in malware and fileless adversary tradecraft to circumvent defenses. It entails the insertion of malicious code into running processes by coercing them to load external dynamic link libraries. These injected DLLs facilitate the execution of arbitrary code within the target process, granting attackers the ability to manipulate its behavior, extract sensitive data, or escalate privileges.

There is a tool called Sysmon which is a great tool that logs system activity to the Windows event log. It provides detailed information about process creation, network connections, file creation, registry changes, and more. Leveraging Sysmon, we can effectively detect DLL injection attempts and enhance our security posture.

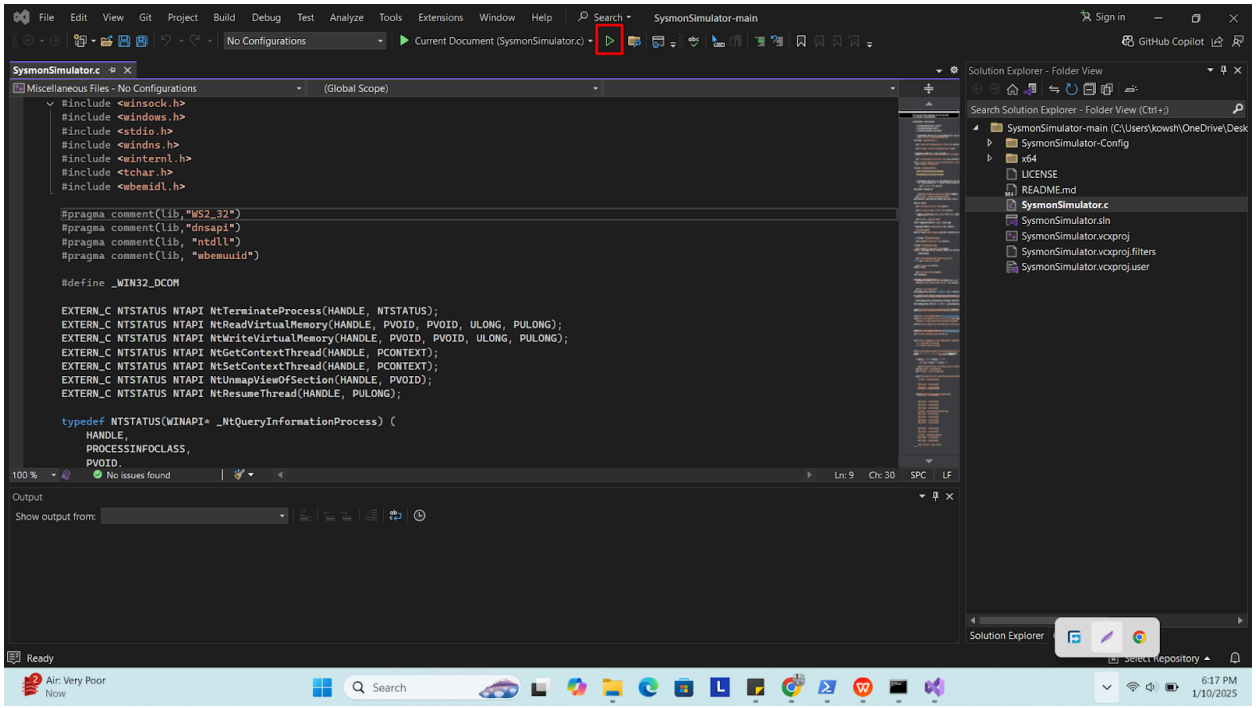
Prerequisite: Need to install Visual studio community 2022

[Visual Studio 2022 Community Edition – Download Latest Free Version (microsoft.com)]

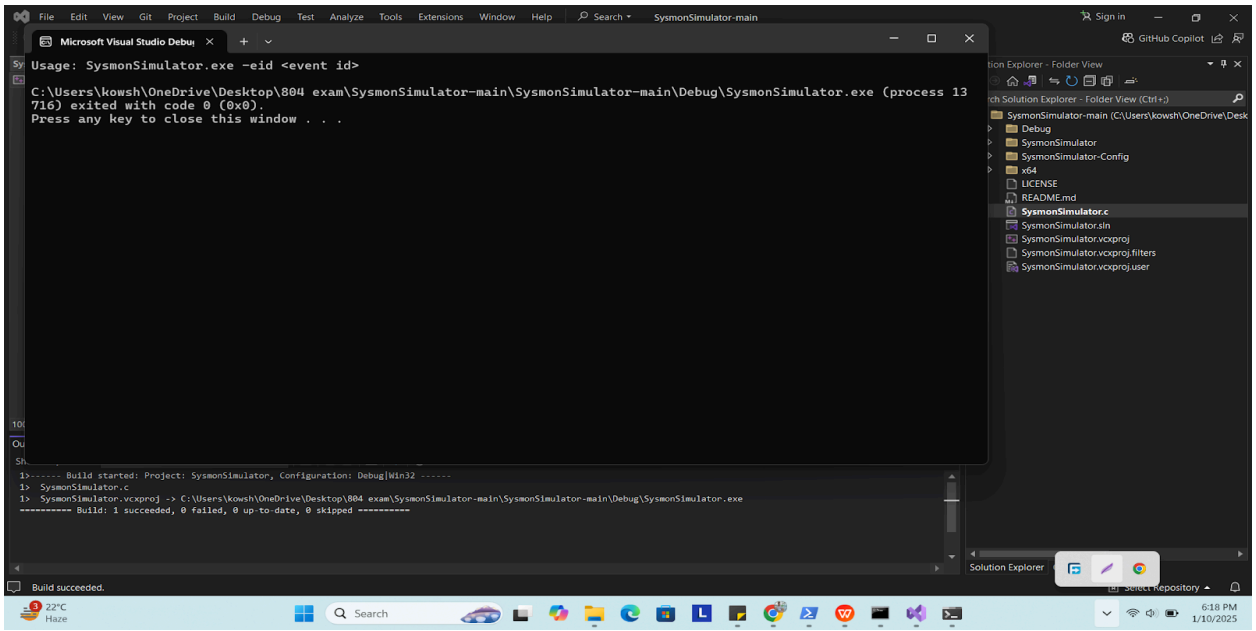




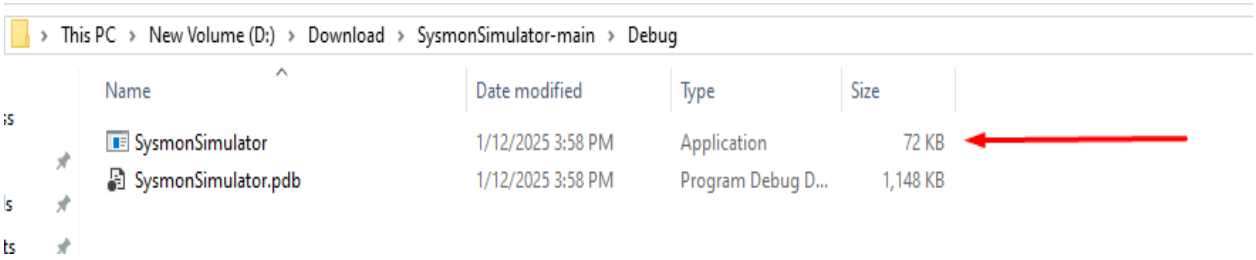
Executed the following command (as administrator) to install the Sysmon driver:



We opened the sysmon simulator folder in **Visual studio community** and ran the **sysmonsimulator.c** file as “Open a project or solution” option.

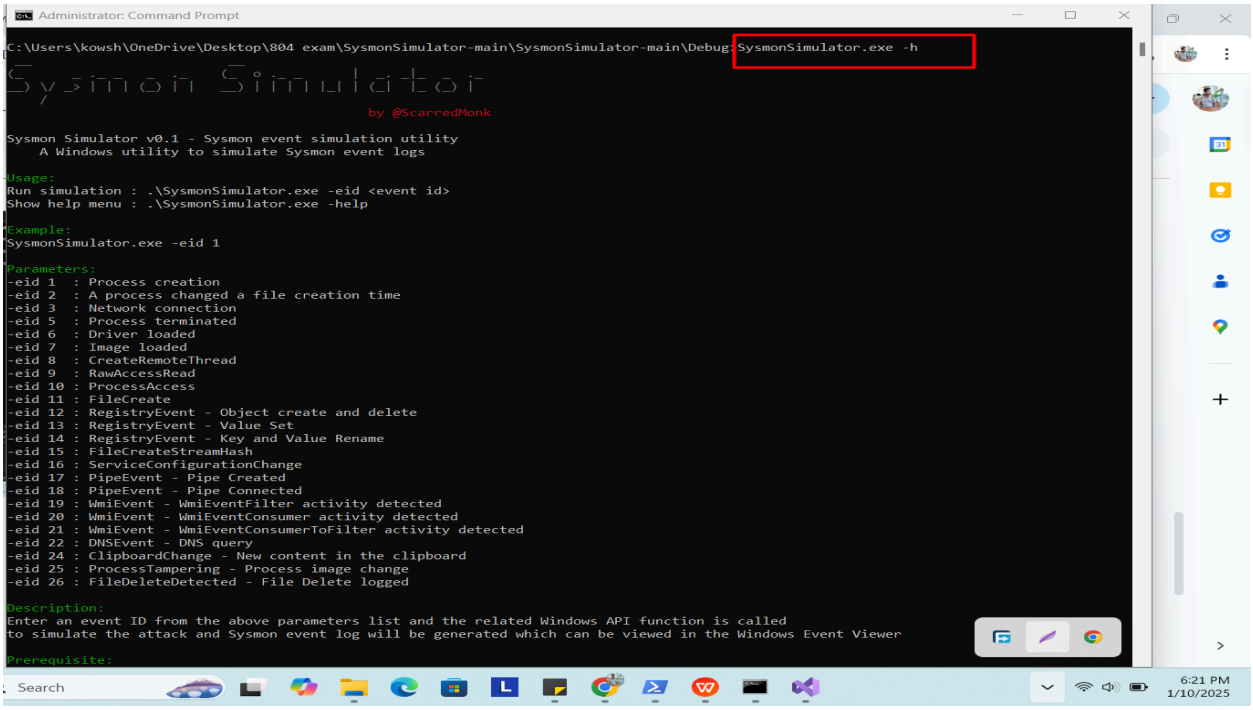


Here the build was successful and we got a **SysmonSimulator.exe** file in debug folder.



This execution gave us a sysmonsimulator.exe file which we used to run the injection to identify whether the sysmon in working or not and also to generate a log for wazuh.

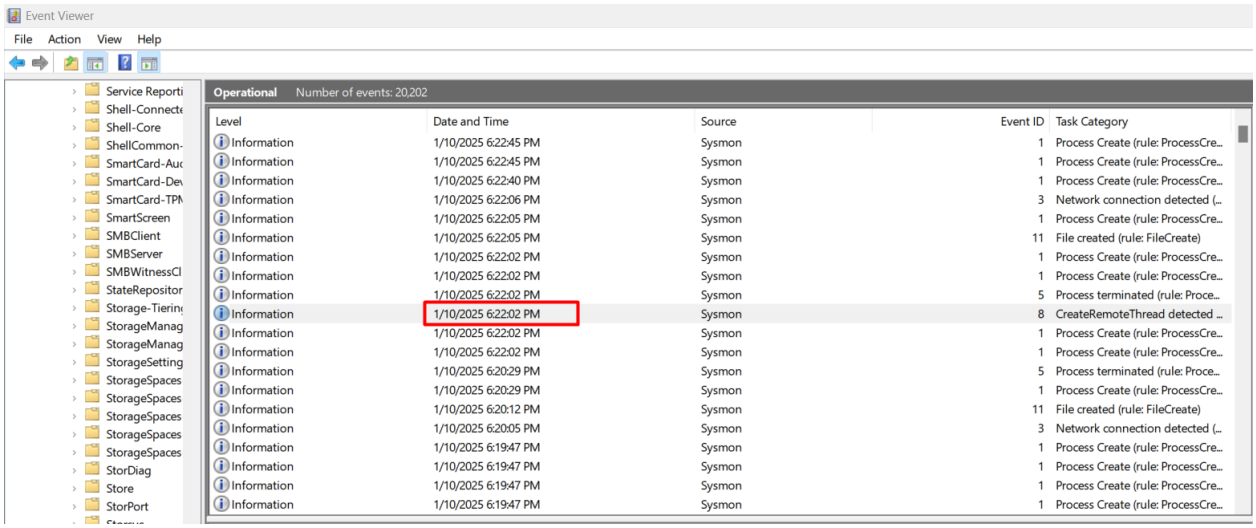
>SysmonSimulator.exe –h



Once Sysmon is installed and configured, let's test it using SysmonSimulator. With SysmonSimulator, we can simulate various attacks using WINAPIs to generate the necessary logs for detection.

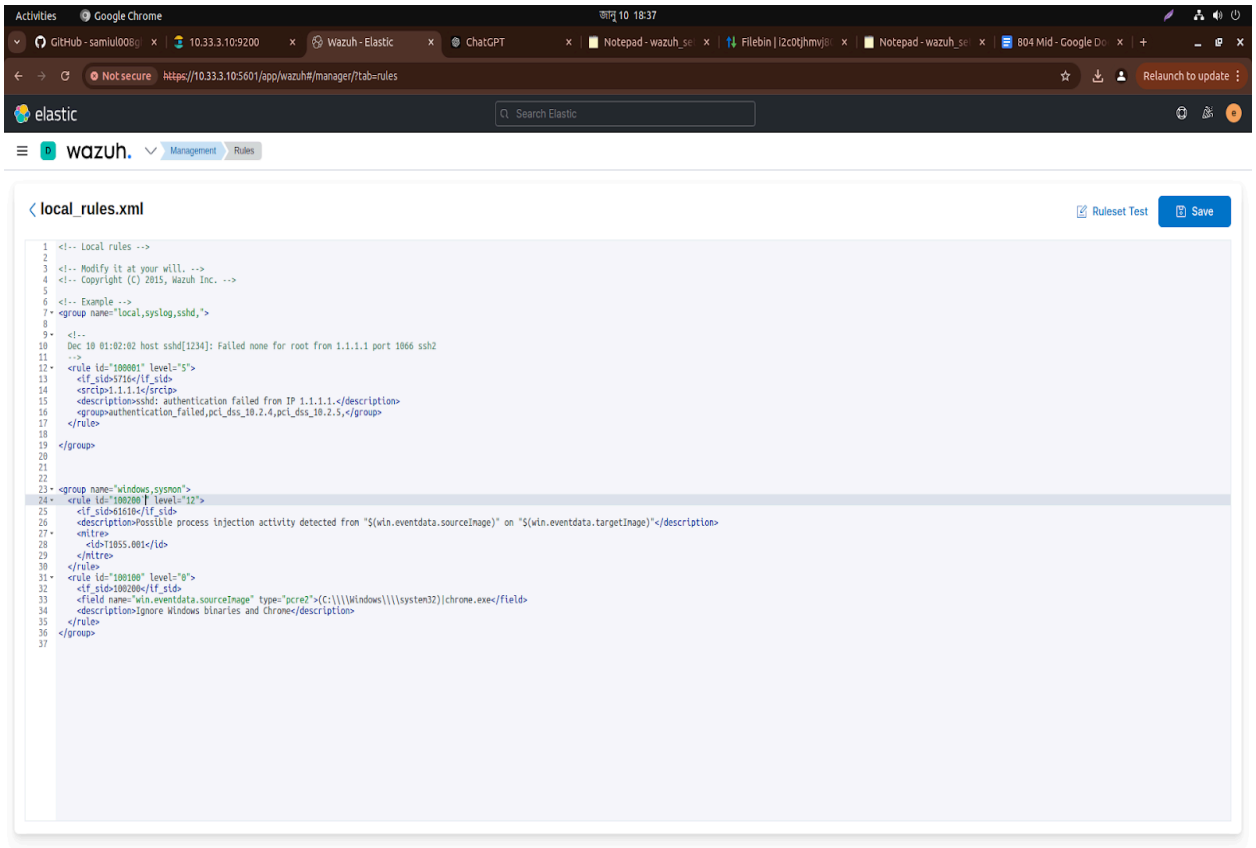
>SysmonSimulator.exe –eid 8

```
C:\Users\kowsh\OneDrive\Desktop\804 exam\SysmonSimulator-main\SysmonSimulator-main\Debug>SysmonSimulator.exe -eid 8

[+] Simulation    : Started successfully
[+] Event ID      : 8
[+] Event Name    : Create Remote Thread Event
[+] Inject into   : PID 10968
[+] Opened process's handle
[+] Created Remote Thread
[+] Closed Handle to the process
[+] Event Viewer  : Check Sysmon Event ID 8 for detection
[+] Event Time    : Event 8 simulation is performed on 10/01/2025 at 18:22:02


C:\Users\kowsh\OneDrive\Desktop\804 exam\SysmonSimulator-main\SysmonSimulator-main\Debug>
```

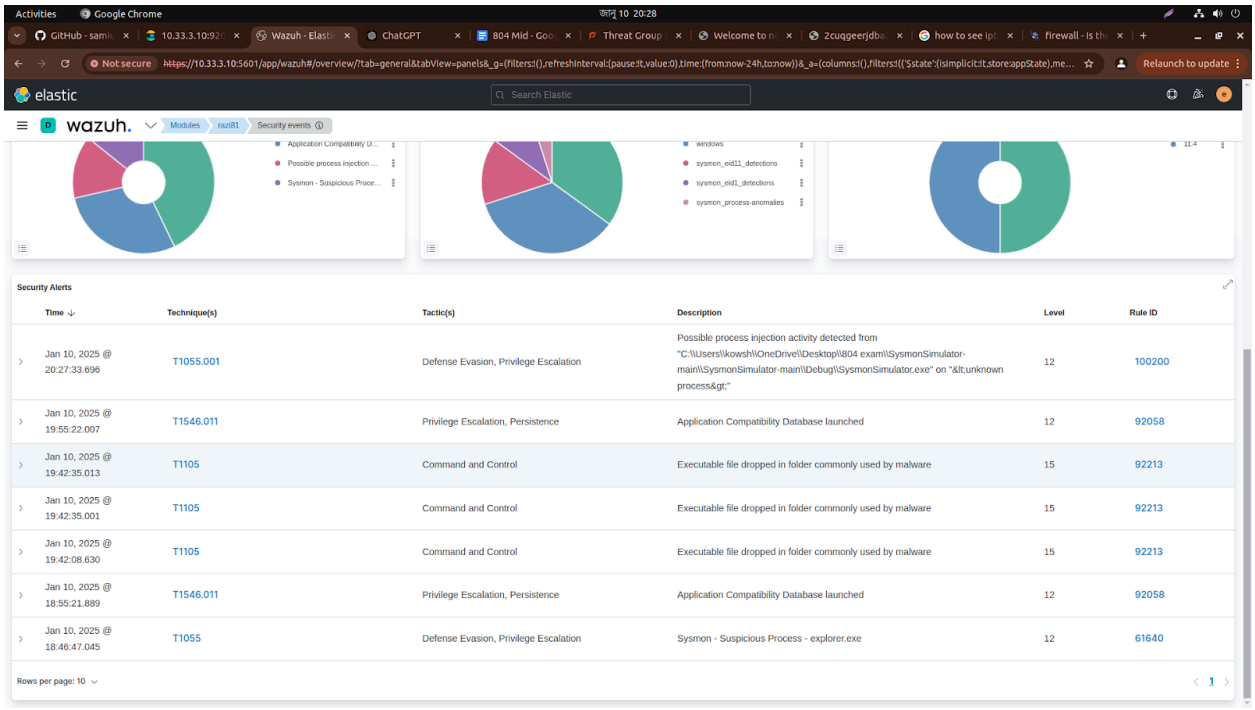


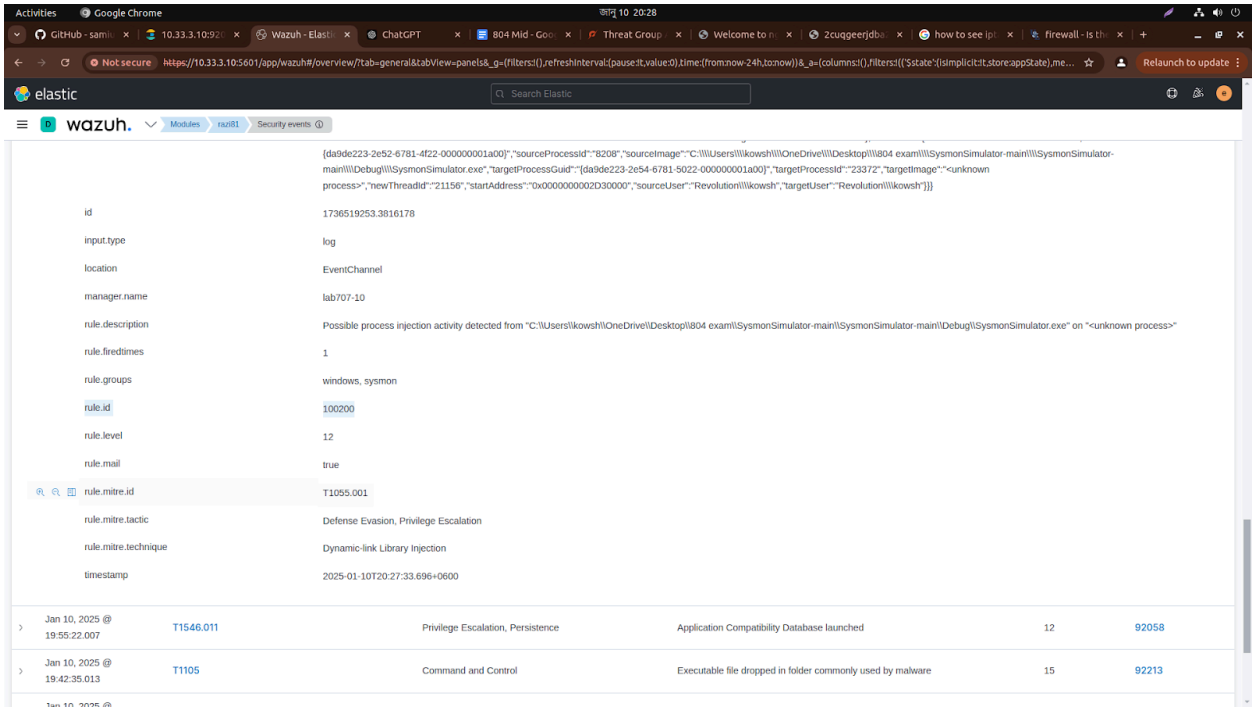
Ubuntu

Change local rules.xml



Alert Generated!





END

SHELL SHOCK ATTACK

What is a Shellshock attack?

Shellshock is a vulnerability that affects Unix-based operating systems, including Linux distributions like Ubuntu, and web servers like Apache. The vulnerability resides in the Bash shell, which is a command-line interpreter widely used on Unix-like operating systems. In a Shellshock attack targeting an Apache2 server on Ubuntu, the attacker exploits this vulnerability to execute arbitrary commands on the server. This can lead to various security breaches, such as data theft, unauthorized access, or the installation of malware. In our shellshock attack we used **HTTP requests** for sending malicious code embedded into the headers.

Let's start, Install Apache2 and start the server:

```
sudo apt update
Sudo apt install apache2
sudo ufw app list                                                    sudo
ufw allow 'Apache'
sudo ufw status
sudo systemctl status apache2
```

```
root@ubuntu:/home/vboxuser# sudo apt update
Hit:1 https://packages.wazuh.com/4.x/apt stable InRelease
Hit:2 https://artifacts.elastic.co/packages/7.x/apt stable InRelease
Hit:3 http://security.ubuntu.com/ubuntu focal-security InRelease
Hit:4 http://ppa.launchpad.net/oisf/suricata-stable/ubuntu focal InRelease
Hit:5 http://bd.archive.ubuntu.com/ubuntu focal InRelease
Get:6 http://bd.archive.ubuntu.com/ubuntu focal-updates InRelease [128 kB]
Hit:7 http://bd.archive.ubuntu.com/ubuntu focal-backports InRelease
Fetched 128 kB in 2s (62.8 kB/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
6 packages can be upgraded. Run 'apt list --upgradable' to see them.
root@ubuntu:/home/vboxuser# sudo apt install apache2
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  apache2-bin apache2-data apache2-utils libapr1 libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-l
Suggested packages:
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom
The following NEW packages will be installed:
  apache2 apache2-bin apache2-data apache2-utils libapr1 libaprutil1 libaprutil1-dbd-sqlite3 libap
0 upgraded, 8 newly installed, 0 to remove and 6 not upgraded.
Need to get 1,723 kB of archives.
After this operation, 7,535 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

```
root@ubuntu:/home/vboxuser# sudo systemctl status apache2
* apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
     Active: active (running) since Sun 2025-01-12 16:10:49 +06; 53s ago
       Docs: https://httpd.apache.org/docs/2.4/
   Main PID: 65503 (apache2)
      Tasks: 55 (limit: 6995)
     Memory: 5.2M
     CGroup: /system.slice/apache2.service
             |-65503 /usr/sbin/apache2 -k start
             |-65504 /usr/sbin/apache2 -k start
             `-65505 /usr/sbin/apache2 -k start
```

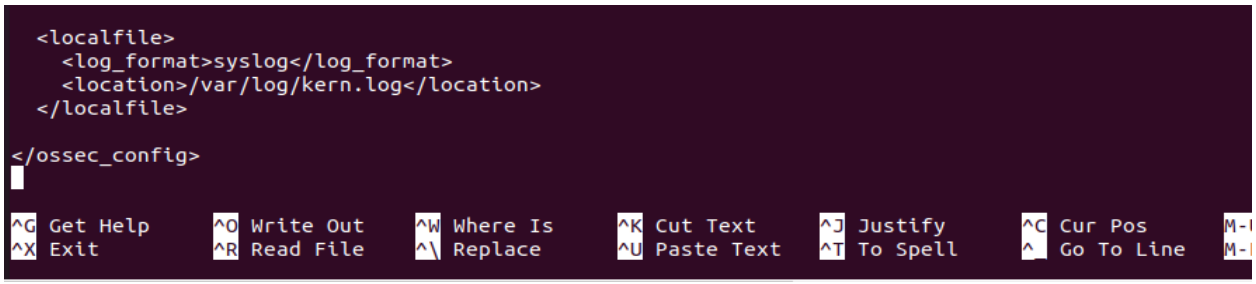
```
root@ubuntu:/home/vboxuser# sudo ufw app list
Available applications:
  Apache
  Apache Full
  Apache Secure
  CUPS
```

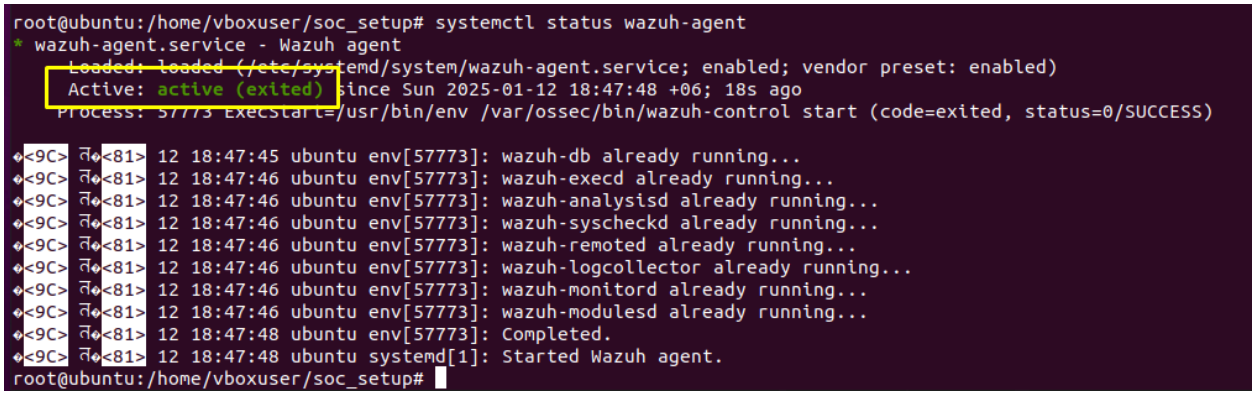
```
root@ubuntu:/home/vboxuser# sudo ufw allow 'Apache'
Rules updated
Rules updated (v6)
```

```
vboxuser@ubuntu:~$ sudo ufw allow 'Apache'
Skipping adding existing rule
Skipping adding existing rule (v6)
```

Add some rules in Wazuh Agent

As we want to collect data of the shellShock attack on apache2 server we need to update some rules in “ossec.conf” file so that our wazuh agent sends us the log.





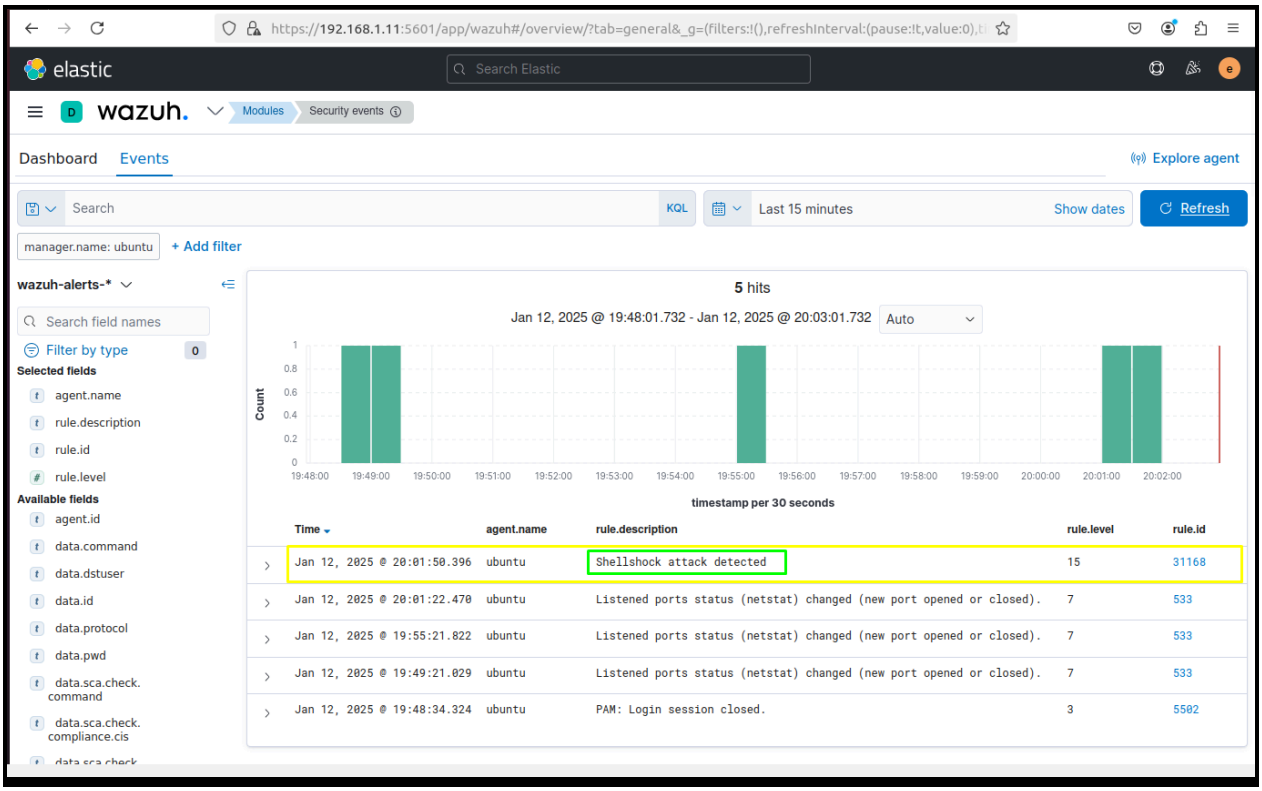
Launch The Attack

sudo curl-H "User-Agent: () { :; }; /bin/cat /etc/passwd" 192.168.1.11

```
┌──(kali㉿kali)-[~/Downloads]
└─$ sudo curl -H "User-Agent: () { :; }; /bin/cat /etc/passwd" 192.168.1.11
[sudo] password for kali:
```

```
                Reporting Problems
        </div>
        <div class="content_section_text">
          <p>
                Please use the <tt>ubuntu-bug</tt> tool to report bugs in the
                Apache2 package with Ubuntu. However, check <a
                href="https://bugs.launchpad.net/ubuntu/+source/apache2"
                rel="nofollow">existing bug reports</a> before reporting a new bug.
          </p>
          <p>
                Please report bugs specific to modules (such as PHP and others)
                to respective packages, not to the web server itself.
          </p>
        </div>



      </div>
    </div>
    <div class="validator">
    </div>
  </body>
</html>


┌──(kali㉿kali)-[~/Downloads]
└─$
```

Shell Shock Attack Got Detected!



Let's view and analyze the attack with wazuh manager:

Wazuh employs rulesets that define specific conditions indicative of a Shellshock attack. These rules analyze system logs, network traffic, or other relevant data sources for patterns or behaviors associated with the exploit.

This is steps the wazuh HTTP decoder took

Packet Capture: Wazuh Can Capture network traffic using tools like tcpdump. When HTTP traffic passes through the network interface being monitored, Wazuh captures the packets containing HTTP requests and responses.

Packet Analysis: Once captured, Wazuh analyzes the captured packets to identify HTTP traffic. The HTTP decoderexaminesthepacketheadersto determine if the packet contains an HTTP request or response.

Header Parsing: WhenanHTTPpacketis identified, the HTTP decoder parses the headers of the request or response. It extracts various components from the HTTP header, such as the request method, URL, headers (like User-Agent), and payload.

Payload Inspection: After parsing the headers, the HTTP decoder inspects the payload of the HTTP request or response. It looks for any anomalies,suspicious patterns, or known attack signatures within the payload.

Rule Matching: Based on the parsed information and payload inspection, Wazuh compares the HTTP traffic against a set of predefined rules designed to detect suspicious or malicious activities. These rules can cover a wide range of threats, including injection attacks, command injection, directory traversal, and more.

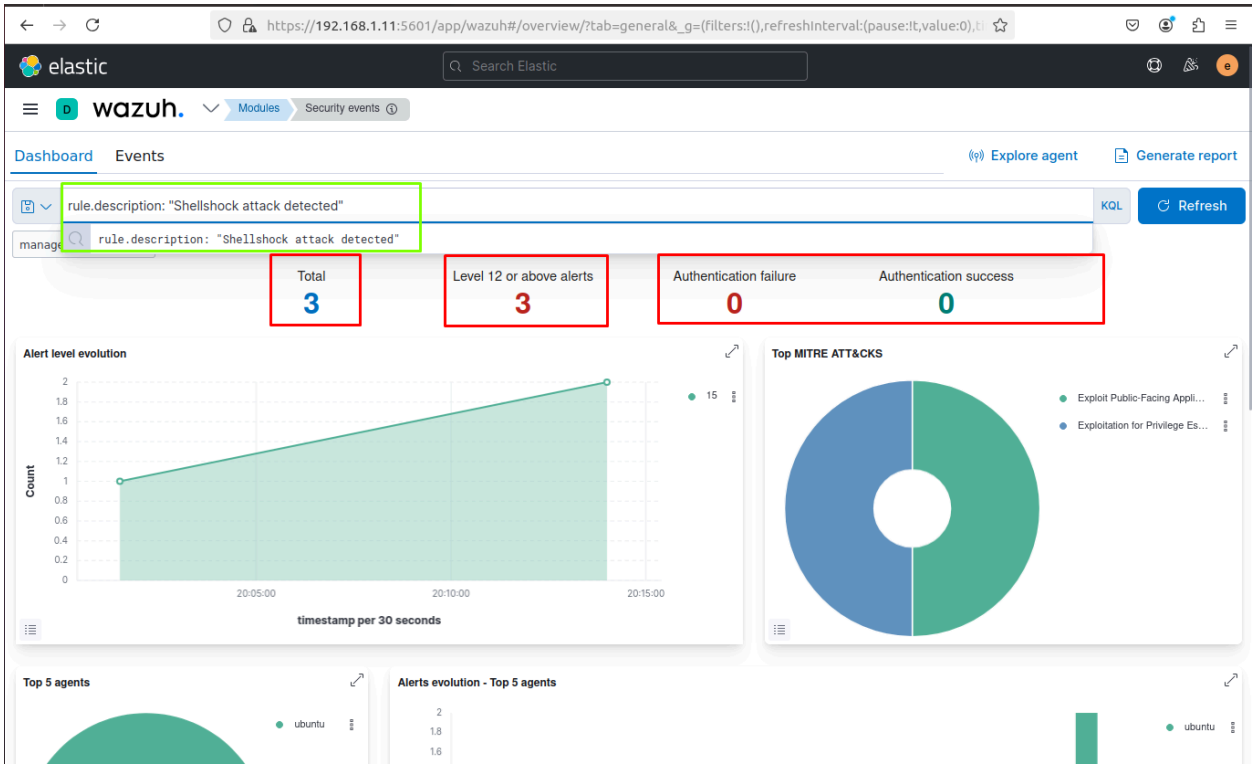
Alert Generation: Upon detecting the Shellshock attack payload in the HTTP request, Wazuh generates an alert for each time I launched the attack, indicating a potential Shellshock attack attempt originating from the source IP address of the HTTP request.

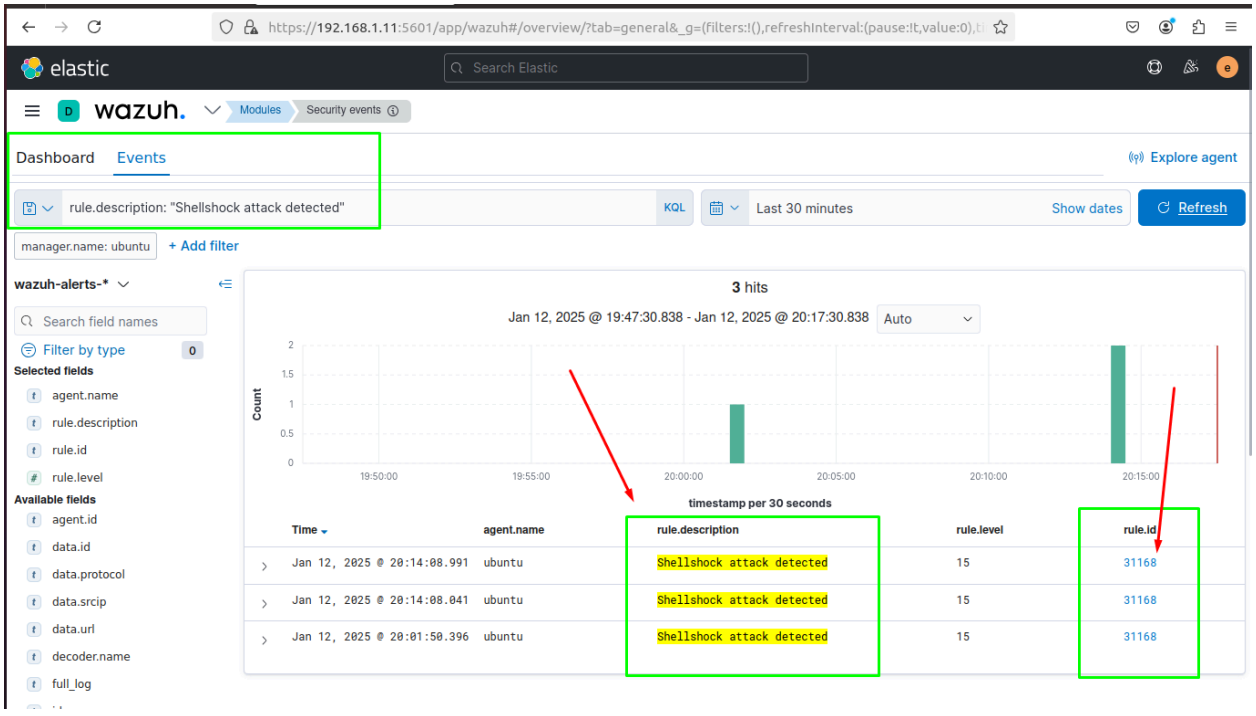
In our attack scenario wazuh used HTTP decoder. It examines HTTP request headers for evidence of the Shellshock payload. So I believe, Wazuh would use an HTTP decoder to parse the web server logs and extract details about the incoming HTTP request, including the User-Agent header containing the potential Shellshock exploit payload
I have launched this attack 3 times thus the number is 3 when we apply rule

```
rule.description:"Shellshock attack detected"
```

Here we are looking for a shellshock attack detected match in the log description. As I launched this attack on Agent 002 this is also added in the filters. We could also filter this by rule.id. In case of shellshock the ID will be 31168. Wealso customize the columns for a better view.

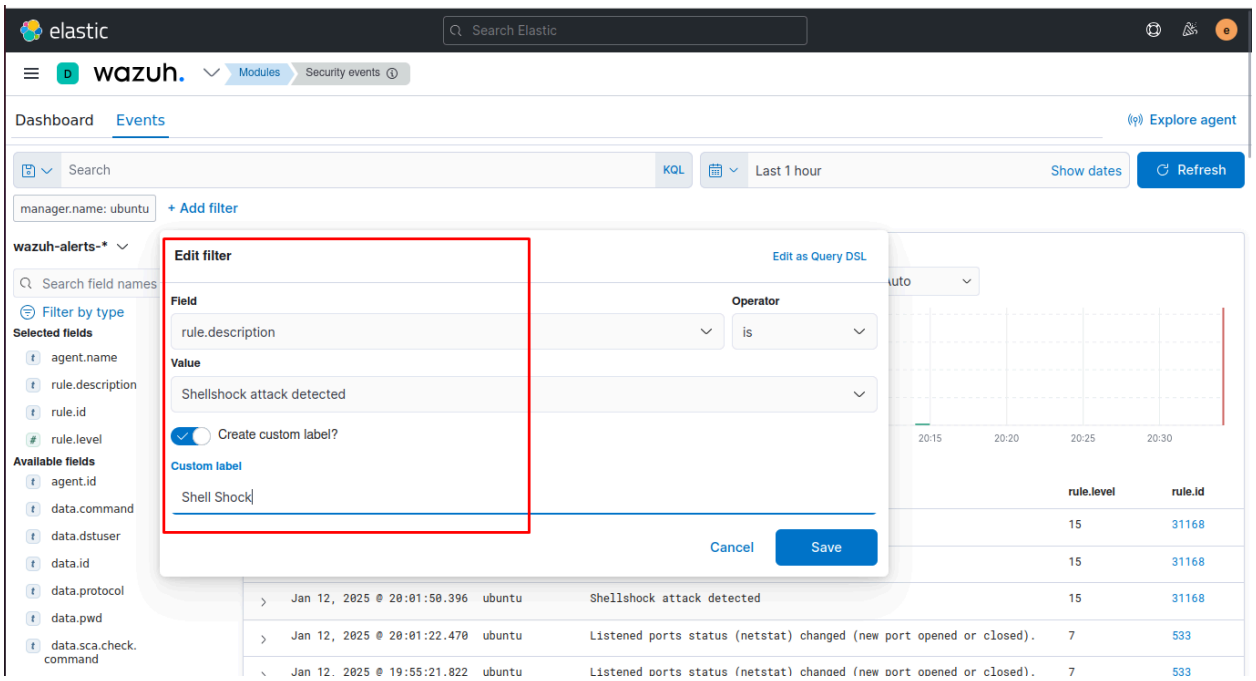
We also customize the columns for a better view.

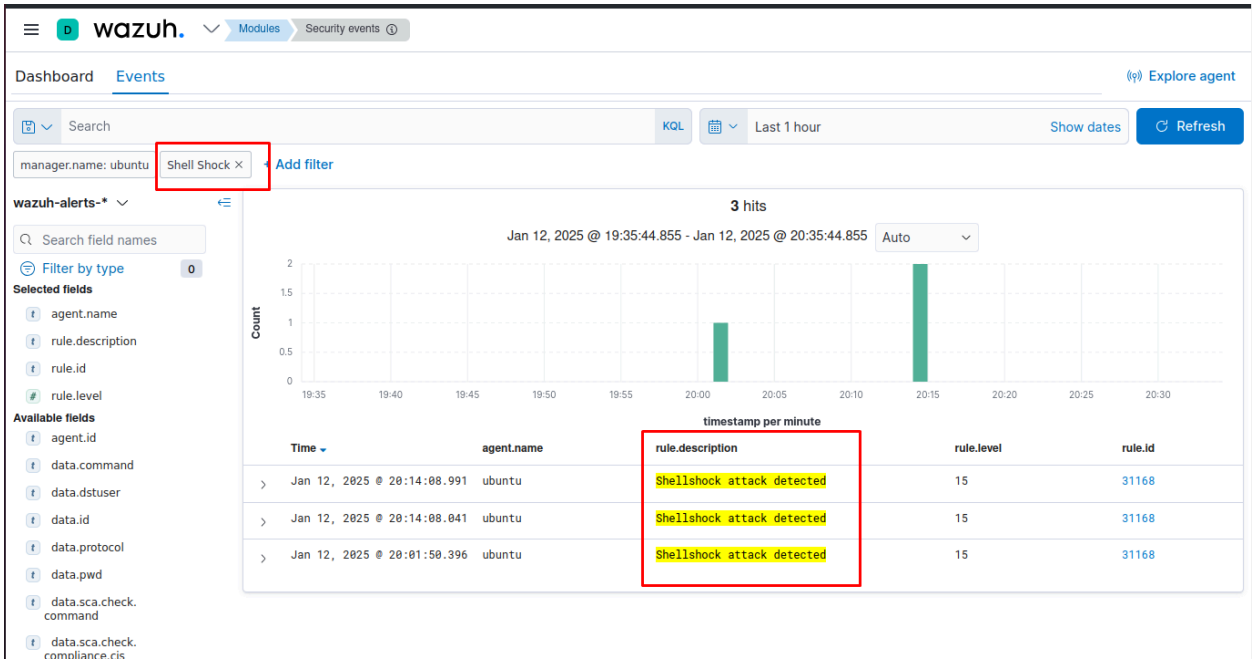

https://192.168.1.11:5601/app/wazuh#/overview/?tab=general&_g=(filters:!(),refreshInterval:(pause:!t,value:0),t
elastic
Search Elastic
wazuh.
Modules
Security events
Dashboard
Events
Explore agent
Generate report
rule.description: "Shellshock attack detected"
KQL
Refresh
Total
3
Level 12 or above alerts
3
Authentication failure
0
Authentication success
0
Alert level evolution
Count
timestamp per 30 seconds
Top MITRE ATT&CKS
Exploit Public-Facing Appli...
Exploitation for Privilege Es...
Top 5 agents
ubuntu
Alerts evolution - Top 5 agents
ubuntu

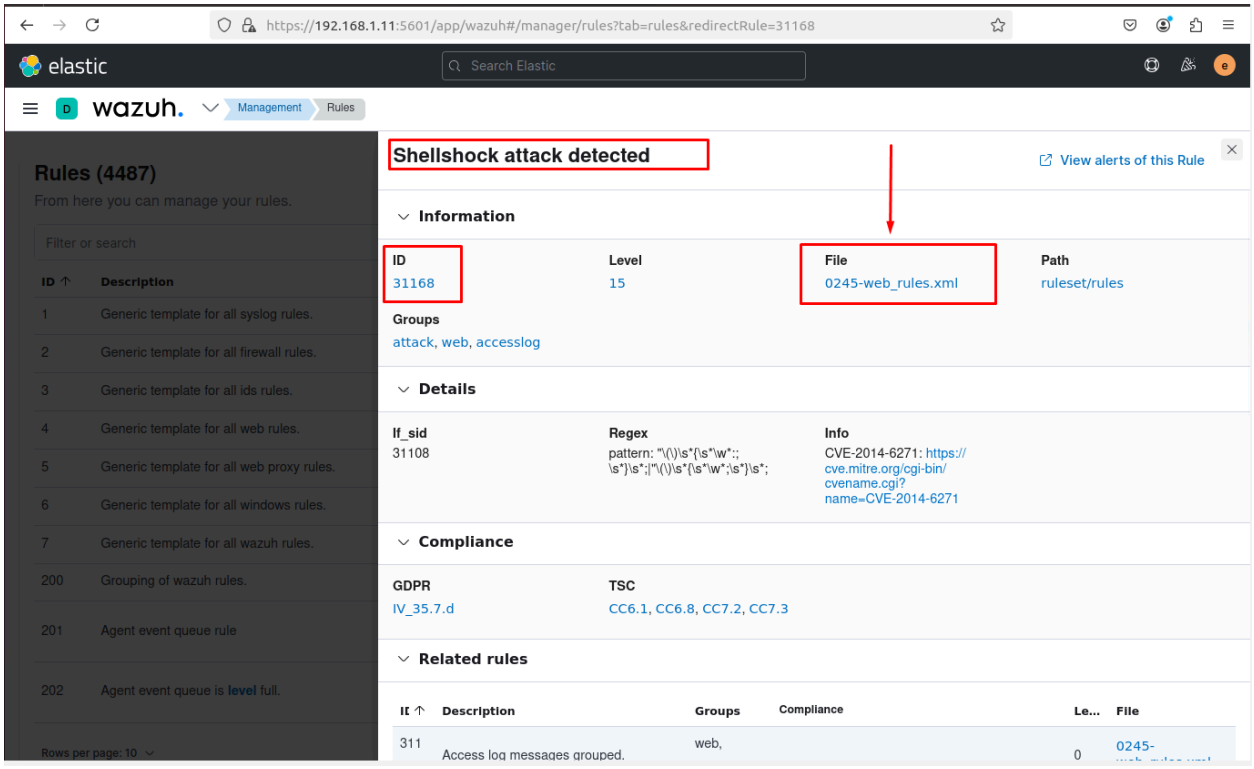

https://192.168.1.11:5601/app/wazuh#/overview/?tab=general&_g=(filters:!(),refreshInterval:(pause:!t,value:0),t
elastic
Search Elastic
wazuh.
Modules
Security events
Dashboard
Events
Explore agent
rule.description: "Shellshock attack detected"
KQL
Last 30 minutes
Show dates
Refresh
manager.name: ubuntu
+ Add filter
wazuh-alerts-*
Search field names
Filter by type
0
Selected fields
agent.name
rule.description
rule.id
rule.level
Available fields
agent.id
data.id
data.protocol
data.srcip
data.url
decoder.name
full_log
3 hits
Jan 12, 2025 @ 19:47:30.838 - Jan 12, 2025 @ 20:17:30.838
Auto
Count
timestamp per 30 seconds
Time
agent.name
rule.description
rule.level
rule.id
Jan 12, 2025 @ 20:14:08.991
ubuntu
Shellshock attack detected
15
31168
Jan 12, 2025 @ 20:14:08.041
ubuntu
Shellshock attack detected
15
31168
Jan 12, 2025 @ 20:01:50.396
ubuntu
Shellshock attack detected
15
31168

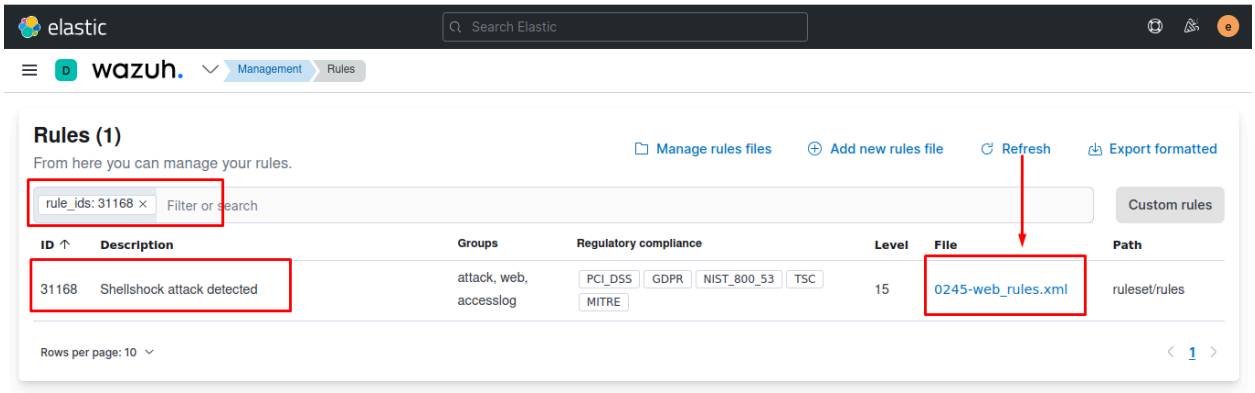
Wazuh rule and Decoder:

This is the process I followed for rule & decoder detection.






https://192.168.1.11:5601/app/wazuh#/manager/rules?tab=rules&redirectRule=31168
elastic
Search Elastic
wazuh.
Management
Rules
Rules (4487)
From here you can manage your rules.
Filter or search
ID
Description
1 Generic template for all syslog rules.
2 Generic template for all firewall rules.
3 Generic template for all ids rules.
4 Generic template for all web rules.
5 Generic template for all web proxy rules.
6 Generic template for all windows rules.
7 Generic template for all wazuh rules.
200 Grouping of wazuh rules.
201 Agent event queue rule
202 Agent event queue is level full.
Rows per page: 10
Shellshock attack detected
View alerts of this Rule
Information
ID
31168
Level
15
File
0245-web_rules.xml
Path
ruleset/rules
Groups
attack, web, accesslog
Details
If_sid
31108
Regex
pattern: "\(\)\s*{\s*\w*:; \s*}\s*;|"\(\)\s*{\s*\w*;\s*}\s*;
Info
CVE-2014-6271: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6271
Compliance
GDPR
IV_35.7.d
TSC
CC6.1, CC6.8, CC7.2, CC7.3
Related rules
ID
Description
Groups
Compliance
Le...
File
311
Access log messages grouped.
web,
0
0245-


elastic
Search Elastic
wazuh.
Management
Rules
Rules (1)
From here you can manage your rules.
Manage rules files
Add new rules file
Refresh
Export formatted
rule_ids: 31168 ×
Filter or search
Custom rules
ID
Description
Groups
Regulatory compliance
Level
File
Path
31168
Shellshock attack detected
attack, web, accesslog
PCI_DSS GDPR NIST_800_53 TSC MITRE
15
0245-web_rules.xml
ruleset/rules
Rows per page: 10
1

0245-web_rules.xml

```
313     Shellshock detected
314     Code: 2xx, 3xx
315   -->
316   <rule id="31168" level="15">
317     <if_sid>31108</if_sid>
318     <regex>"\(\)\s*{\s*\w*:;\s*}\s*;|"\(\)\s*{\s*\w*;\s*}\s*;</regex>
319     <description>Shellshock attack detected</description>
320     <mitre>
321       <id>T1068</id>
322       <id>T1190</id>
323     </mitre>
324     <info type="cve">CVE-2014-6271</info>
325     <info type="link">https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6271</info>
326     <group>attack,pci_dss_11.4,gdpr_IV_35.7.d,nist_800_53_SI.4,tsc_CC6.1,tsc_CC6.8,tsc_CC7.2,tsc_CC7.3,</group>
327   </rule>
328
329   <rule id="31169" level="15">
330     <if_sid>31108</if_sid>
331     <regex>"\(\)\s*{\s*_;\.*}\s*>_[\$\(\$\(\)\)]\s*{</regex>
332     <description>Shellshock attack detected</description>
333     <mitre>
334       <id>T1068</id>
335       <id>T1190</id>
336     </mitre>
337     <info type="cve">CVE-2014-6278</info>
338     <info type="link">https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6278</info>
339     <group>attack,pci_dss_11.4,gdpr_IV_35.7.d,nist_800_53_SI.4,tsc_CC6.1,tsc_CC6.8,tsc_CC7.2,tsc_CC7.3,</group>
340   </rule>
341
342   <rule id="31170" level="6">
343     <if_sid>31100</if_sid>
344     <url>%2csleep|sysdate()|nslookup%20dns.sqli</url>
345     <description>SQL injection attempt.</description>
346     <group>attack,sqlinjection,pci_dss_6.5,pci_dss_11.4,pci_dss_6.5.1,gdpr_IV_35.7.d,</group>
347   </rule>
348
```

Rules for Shell Shock Attack

END