Detecting ransomware with Wazuh by monitoring the file system

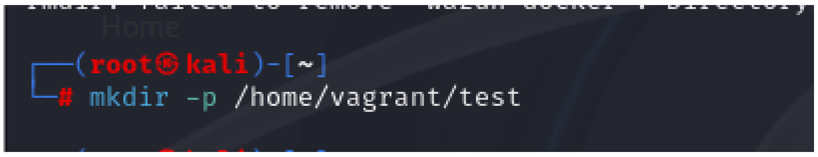
-R.M.NAREN ADITHYA

Let's now run a simple proof of concept using Wazuh file integrity monitoring module. For it, we created a Python script (wazuh-ransomware-poc.py) to simulate a ransomware attack. The script requires Python 3 and the cryptography package.

Step 1: Prepare the test environment

First, we create the /home/vagrant/test directory:

```
# mkdir -p /home/vagrant/test
```

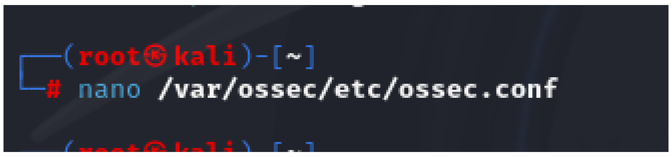


```
┌──(root㉿kali)-[~]
└─# mkdir -p /home/vagrant/test
```

We need to configure the Wazuh agent to monitor the previous directory:

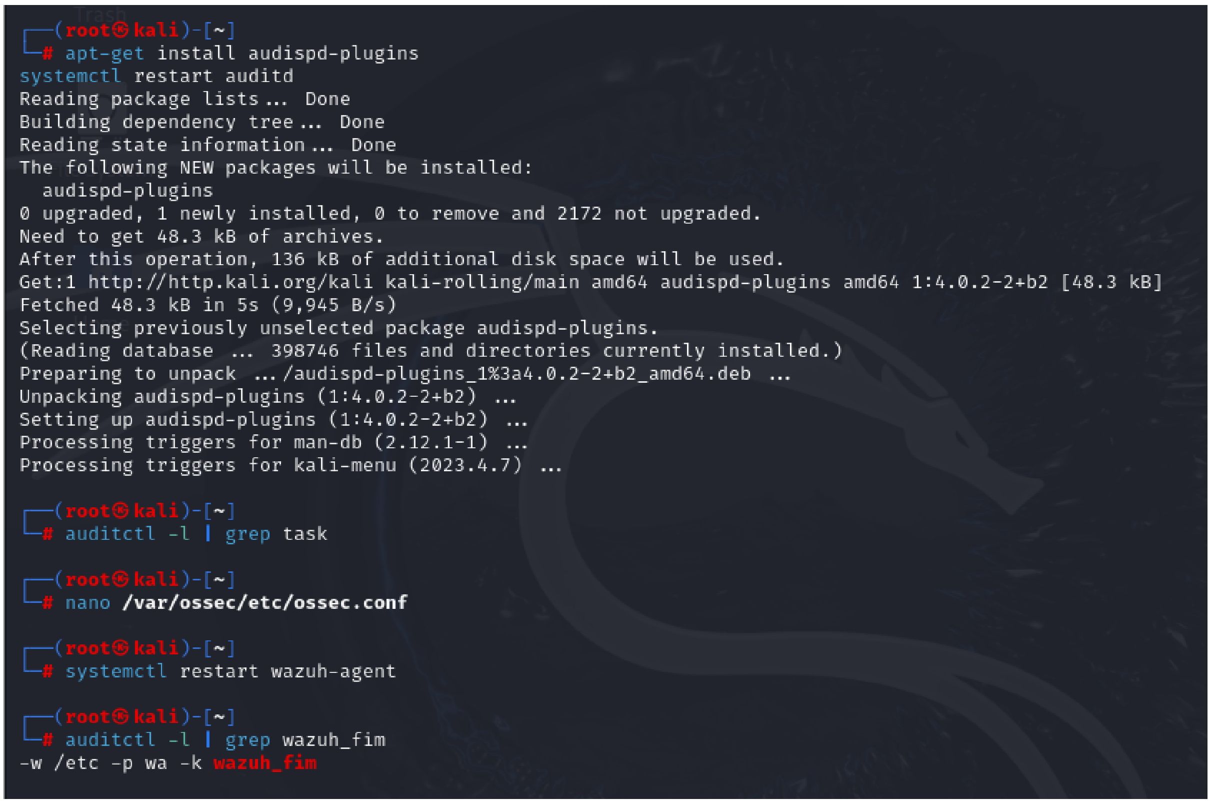
/var/ossec/etc/ossec.conf

```
<syscheck>
  <directories check_all="yes" whodata="yes">/home/vagrant/test</directories>
</syscheck>
```



```
┌──(root㉿kali)-[~]
└─# nano /var/ossec/etc/ossec.conf
```

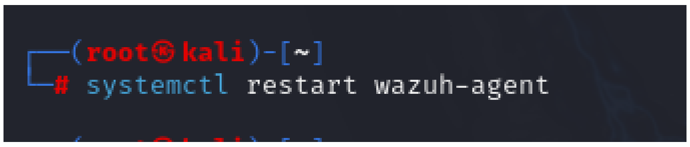
Note that we enabled whodata. This will make the Wazuh agent use an integration with the operating system kernel in order to report file changes in real-time and include details on who and how those changes were made.



```
┌──(root㉿kali)-[~]
└─# apt-get install audispd-plugins
systemctl restart auditd
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  audispd-plugins
0 upgraded, 1 newly installed, 0 to remove and 2172 not upgraded.
Need to get 48.3 kB of archives.
After this operation, 136 kB of additional disk space will be used.
Get:1 http://http.kali.org/kali kali-rolling/main amd64 audispd-plugins amd64 1:4.0.2-2+b2 [48.3 kB]
Fetched 48.3 kB in 5s (9,945 B/s)
Selecting previously unselected package audispd-plugins.
(Reading database ... 398746 files and directories currently installed.)
Preparing to unpack .../audispd-plugins_1%3a4.0.2-2+b2_amd64.deb ...
Unpacking audispd-plugins (1:4.0.2-2+b2) ...
Setting up audispd-plugins (1:4.0.2-2+b2) ...
Processing triggers for man-db (2.12.1-1) ...
Processing triggers for kali-menu (2023.4.7) ...

┌──(root㉿kali)-[~]
└─# auditctl -l | grep task

┌──(root㉿kali)-[~]
└─# nano /var/ossec/etc/ossec.conf

┌──(root㉿kali)-[~]
└─# systemctl restart wazuh-agent

┌──(root㉿kali)-[~]
└─# auditctl -l | grep wazuh_fim
-w /etc -p wa -k wazuh_fim
```

Restart the agent to apply changes:

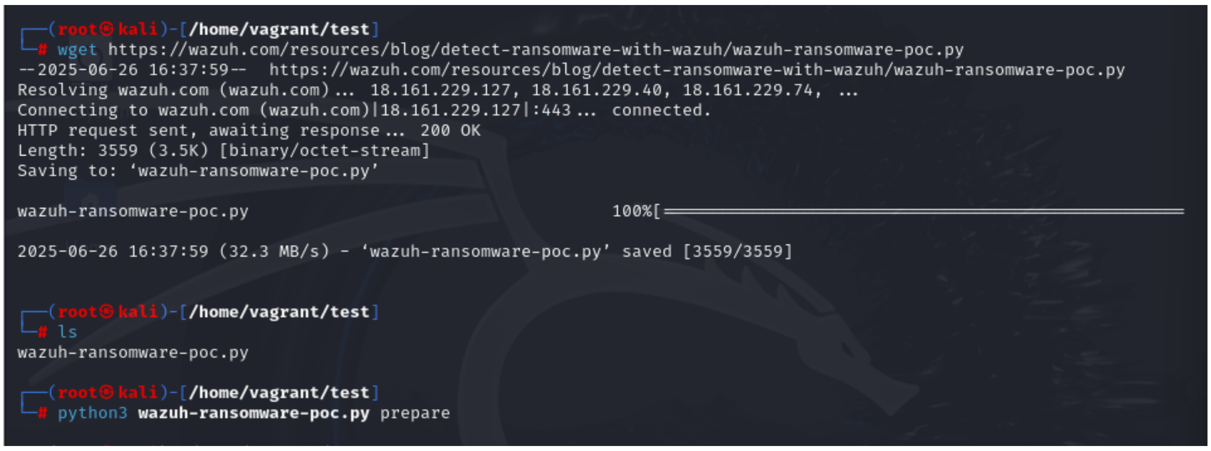
systemctl restart wazuh-agent



```
┌──(root㉿kali)-[~]
└─# systemctl restart wazuh-agent
```

We create several files and subdirectories in our agent. By default, the script will add 10 directories with 20 files each of 1KB in /home/vagrant/test:

python3 wazuh-ransomware-poc.py prepare



```
┌──(root㉿kali)-[/home/vagrant/test]
└─# wget https://wazuh.com/resources/blog/detect-ransomware-with-wazuh/wazuh-ransomware-poc.py
--2025-06-26 16:37:59--  https://wazuh.com/resources/blog/detect-ransomware-with-wazuh/wazuh-ransomware-poc.py
Resolving wazuh.com (wazuh.com)... 18.161.229.127, 18.161.229.40, 18.161.229.74, ...
Connecting to wazuh.com (wazuh.com)|18.161.229.127|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3559 (3.5K) [binary/octet-stream]
Saving to: 'wazuh-ransomware-poc.py'

wazuh-ransomware-poc.py                         100%[==========================================]

2025-06-26 16:37:59 (32.3 MB/s) - 'wazuh-ransomware-poc.py' saved [3559/3559]


┌──(root㉿kali)-[/home/vagrant/test]
└─# ls
wazuh-ransomware-poc.py

┌──(root㉿kali)-[/home/vagrant/test]
└─# python3 wazuh-ransomware-poc.py prepare
```

Now the directories and files created can be listed:

# ls -lRh /home/vagrant/test/

/home/vagrant/test/:

total 40K

drwxr-xr-x. 2 root root 4.0K Nov 28 14:27 Directory_00

drwxr-xr-x. 2 root root 4.0K Nov 28 14:27 Directory_01

drwxr-xr-x. 2 root root 4.0K Nov 28 14:27 Directory_02

drwxr-xr-x. 2 root root 4.0K Nov 28 14:27 Directory_03

drwxr-xr-x. 2 root root 4.0K Nov 28 14:27 Directory_04

drwxr-xr-x. 2 root root 4.0K Nov 28 14:27 Directory_05

drwxr-xr-x. 2 root root 4.0K Nov 28 14:27 Directory_06

drwxr-xr-x. 2 root root 4.0K Nov 28 14:27 Directory_07

drwxr-xr-x. 2 root root 4.0K Nov 28 14:27 Directory_08

drwxr-xr-x. 2 root root 4.0K Nov 28 14:27 Directory_09

/home/vagrant/test/Directory_00:

total 80K

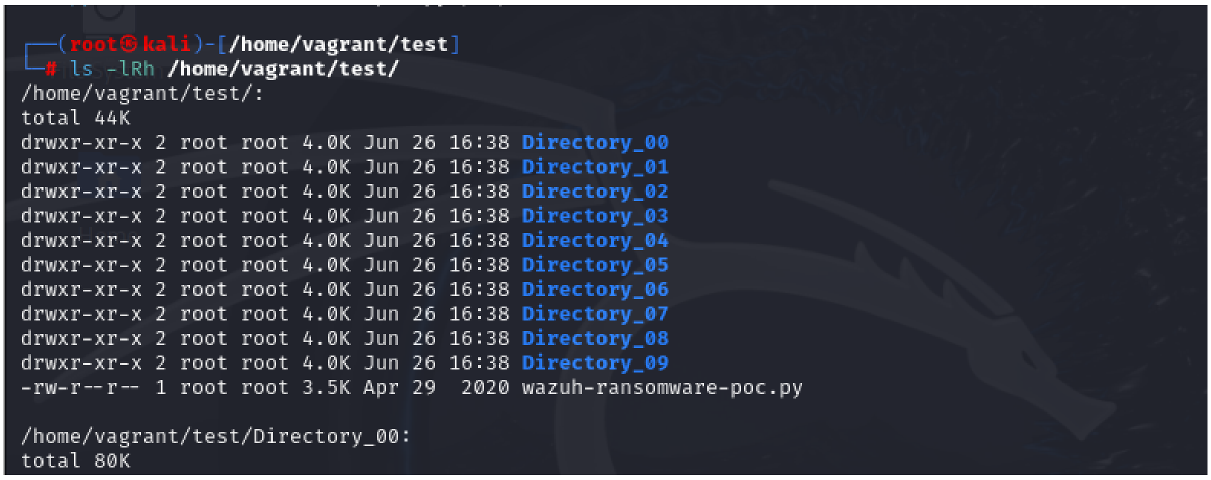
-rw-r--r--. 1 root root 1.0K Nov 28 14:27 File_00.txt

-rw-r--r--. 1 root root 1.0K Nov 28 14:27 File_19.txt

/home/vagrant/test/Directory_01:

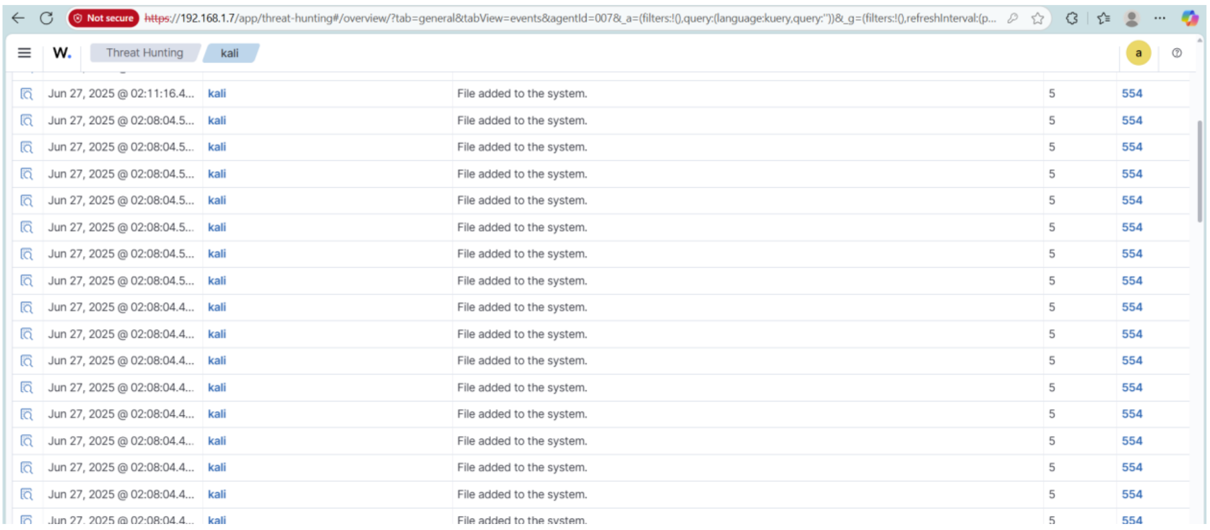
total 80K

-rw-r--r--. 1 root root 1.0K Nov 28 14:27 File_00.txt



```
┌──(root㉿kali)-[/home/vagrant/test]
└─# ls -lRh /home/vagrant/test/
/home/vagrant/test/:
total 44K
drwxr-xr-x 2 root root 4.0K Jun 26 16:38 Directory_00
drwxr-xr-x 2 root root 4.0K Jun 26 16:38 Directory_01
drwxr-xr-x 2 root root 4.0K Jun 26 16:38 Directory_02
drwxr-xr-x 2 root root 4.0K Jun 26 16:38 Directory_03
drwxr-xr-x 2 root root 4.0K Jun 26 16:38 Directory_04
drwxr-xr-x 2 root root 4.0K Jun 26 16:38 Directory_05
drwxr-xr-x 2 root root 4.0K Jun 26 16:38 Directory_06
drwxr-xr-x 2 root root 4.0K Jun 26 16:38 Directory_07
drwxr-xr-x 2 root root 4.0K Jun 26 16:38 Directory_08
drwxr-xr-x 2 root root 4.0K Jun 26 16:38 Directory_09
-rw-r--r-- 1 root root 3.5K Apr 29  2020 wazuh-ransomware-poc.py

/home/vagrant/test/Directory_00:
total 80K
```

From the Wazuh UI, we can see the new files:



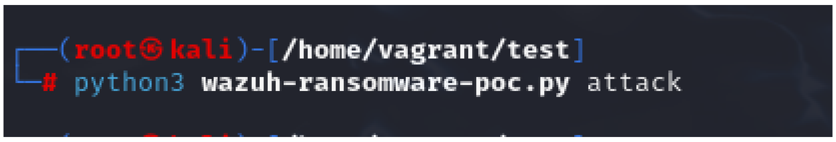
Not secure https://192.168.1.7/app/threat-hunting#/overview/?tab=general&tabView=events&agentId=007&_a=(filters:!(),query:(language:kuery,query:''))&_g=(filters:!(),refreshInterval:(p...

Jun 27, 2025 @ 02:11:16.4...	kali	File added to the system.	5	554
Jun 27, 2025 @ 02:08:04.5...	kali	File added to the system.	5	554
Jun 27, 2025 @ 02:08:04.5...	kali	File added to the system.	5	554
Jun 27, 2025 @ 02:08:04.5...	kali	File added to the system.	5	554
Jun 27, 2025 @ 02:08:04.5...	kali	File added to the system.	5	554
Jun 27, 2025 @ 02:08:04.5...	kali	File added to the system.	5	554
Jun 27, 2025 @ 02:08:04.5...	kali	File added to the system.	5	554
Jun 27, 2025 @ 02:08:04.5...	kali	File added to the system.	5	554
Jun 27, 2025 @ 02:08:04.4...	kali	File added to the system.	5	554
Jun 27, 2025 @ 02:08:04.4...	kali	File added to the system.	5	554
Jun 27, 2025 @ 02:08:04.4...	kali	File added to the system.	5	554
Jun 27, 2025 @ 02:08:04.4...	kali	File added to the system.	5	554
Jun 27, 2025 @ 02:08:04.4...	kali	File added to the system.	5	554
Jun 27, 2025 @ 02:08:04.4...	kali	File added to the system.	5	554
Jun 27, 2025 @ 02:08:04.4...	kali	File added to the system.	5	554
Jun 27, 2025 @ 02:08:04.4...	kali	File added to the system.	5	554
Jun 27, 2025 @ 02:08:04.4...	kali	File added to the system.	5	554

Step 2: Simulating the attack

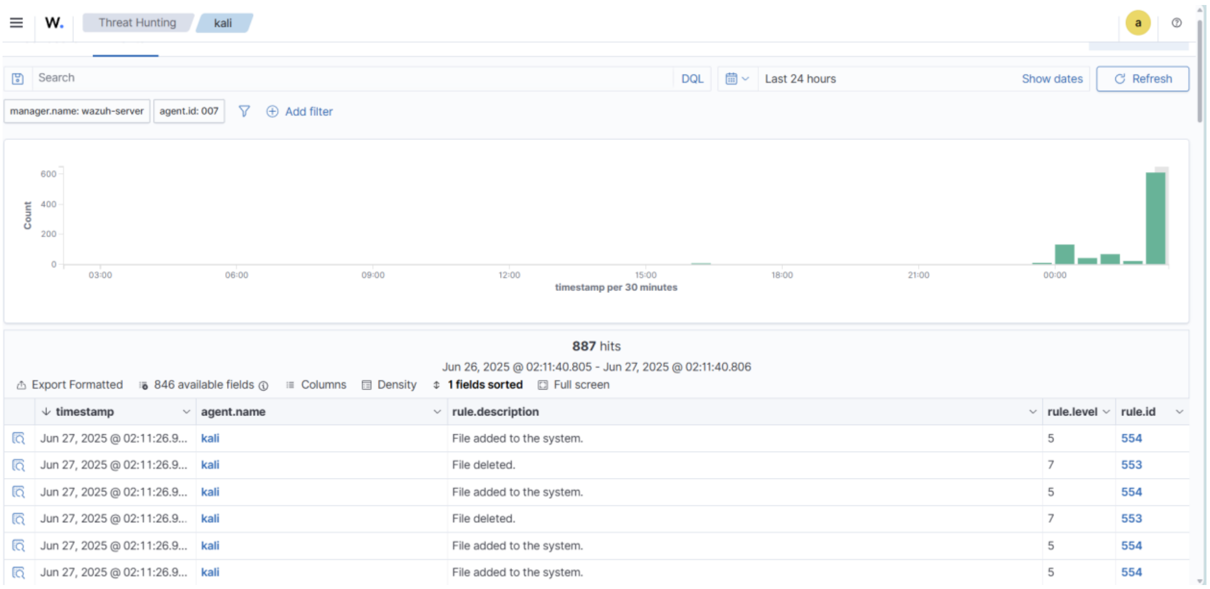
Now it is time to simulate the ransomware attack. The following command will encrypt every file in /home/vagrant/test and will remove the original one:

python3 wazuh-ransomware-poc.py attack



```
┌──(root㉿kali)-[/home/vagrant/test]
└─# python3 wazuh-ransomware-poc.py attack
```

From the Wazuh UI, we see the two types of file integrity monitoring alerts: added and deleted.



887 hits

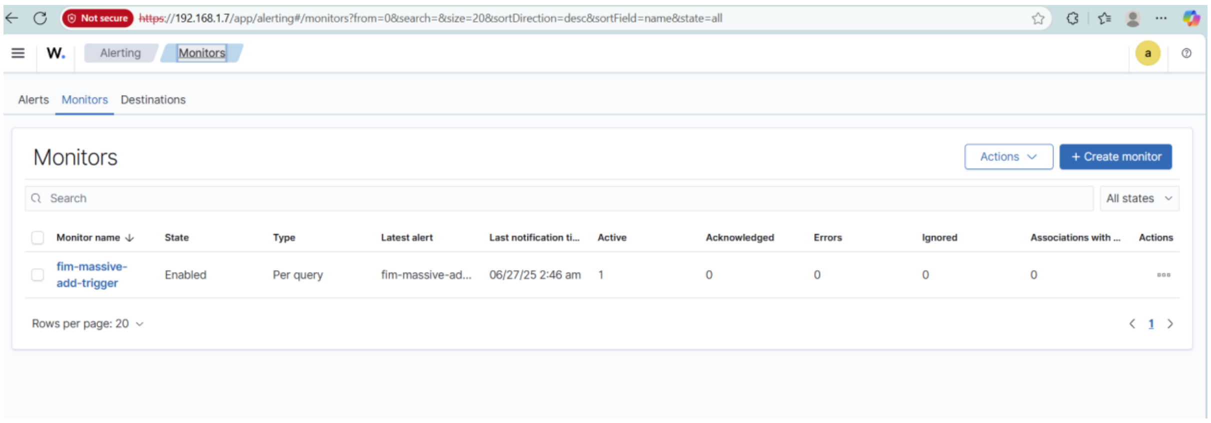
Jun 26, 2025 @ 02:11:40.805 - Jun 27, 2025 @ 02:11:40.806

Export Formatted · 846 available fields · Columns · Density · 1 fields sorted · Full screen

timestamp	agent.name	rule.description	rule.level	rule.id
Jun 27, 2025 @ 02:11:26.9...	kali	File added to the system.	5	554
Jun 27, 2025 @ 02:11:26.9...	kali	File deleted.	7	553
Jun 27, 2025 @ 02:11:26.9...	kali	File added to the system.	5	554
Jun 27, 2025 @ 02:11:26.9...	kali	File deleted.	7	553
Jun 27, 2025 @ 02:11:26.9...	kali	File added to the system.	5	554
Jun 27, 2025 @ 02:11:26.9...	kali	File added to the system.	5	554

Wazuh successfully detected the events that are generated during the attack in this simulation.

Monitoring Wazuh alerts



```
Not secure  https://192.168.1.7/app/alerting#/monitors?from=0&search=&size=20&sortDirection=desc&sortField=name&state=all
```

Monitors

Monitor name	State	Type	Latest alert	Last notification ti...	Active	Acknowledged	Errors	Ignored	Associations with ...	Actions
fim-massive-add-trigger	Enabled	Per query	fim-massive-ad...	06/27/25 2:46 am	1	0	0	0	0	•••

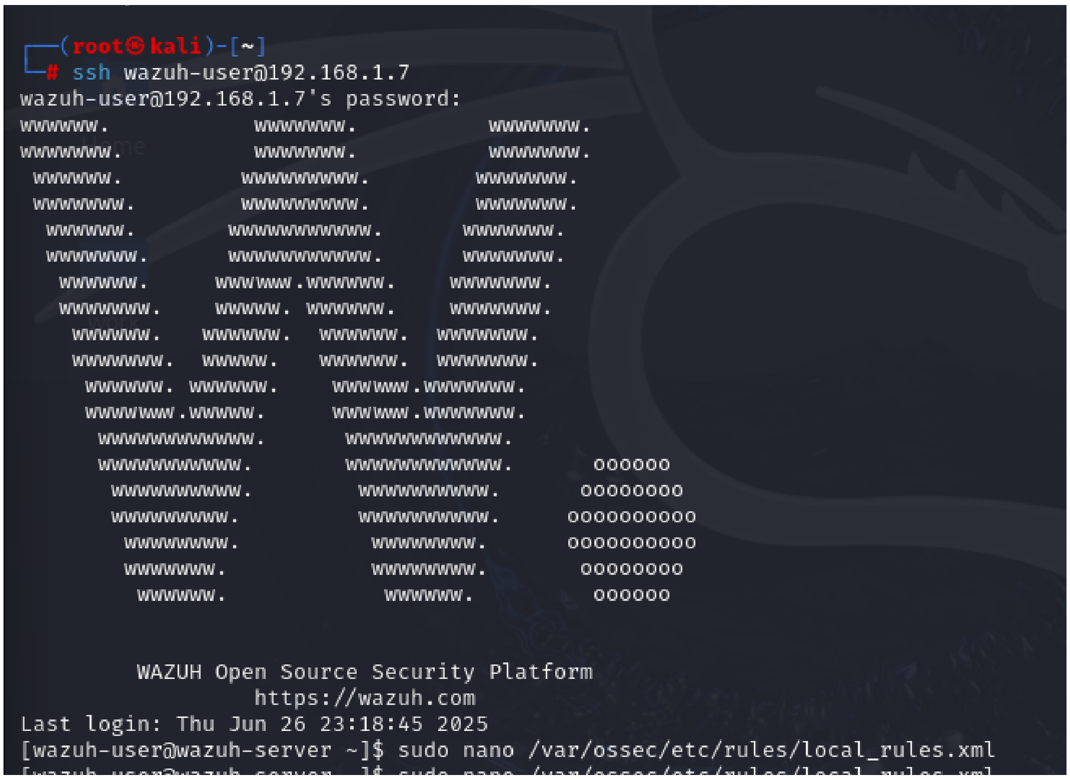
Customize Wazuh Rules

Step 1: Clone an Existing Rule

Wazuh rules are stored in XML files, and custom rules should be placed in /var/ossec/etc/rules/local_rules.xml to avoid being overwritten during upgrades. We'll clone an existing SSH-related rule as a starting point and modify it for a custom log behavior.

1. **Navigate to the Rules Directory**:

sudo nano /var/ossec/etc/rules/local_rules.xml



```
┌──(root㉿kali)-[~]
└─# ssh wazuh-user@192.168.1.7
wazuh-user@192.168.1.7's password:
wwwwww.           wwwwwww.          wwwwwww.
wwwwwww.          wwwwwww.          wwwwwww.
 wwwwww.         wwwwwwwww.        wwwwwww.
 wwwwwww.        wwwwwwwww.        wwwwwww.
  wwwwww.       wwwwwwwwwww.      wwwwwww.
  wwwwwww.      wwwwwwwwwww.      wwwwwww.
   wwwwww.     wwwwww.wwwwww.    wwwwwww.
   wwwwwww.    wwwww. wwwwww.    wwwwwww.
    wwwwww.   wwwwww.  wwwwww.  wwwwwww.
    wwwwwww.  wwwww.   wwwwww.  wwwwwww.
     wwwwww. wwwwww.    wwwwww.wwwwwww.
     wwwwwww.wwwww.     wwwwww.wwwwwww.
      wwwwwwwwwwww.      wwwwwwwwwwww.
      wwwwwwwwwww.       wwwwwwwwwwww.      oooooo
       wwwwwwwwww.        wwwwwwwwww.      oooooooo
       wwwwwwwww.         wwwwwwwwww.     oooooooooo
        wwwwwwww.          wwwwwwww.      oooooooooo
        wwwwwww.           wwwwwwww.       oooooooo
         wwwwww.            wwwwww.         oooooo


         WAZUH Open Source Security Platform
                  https://wazuh.com
Last login: Thu Jun 26 23:18:45 2025
[wazuh-user@wazuh-server ~]$ sudo nano /var/ossec/etc/rules/local_rules.xml
```

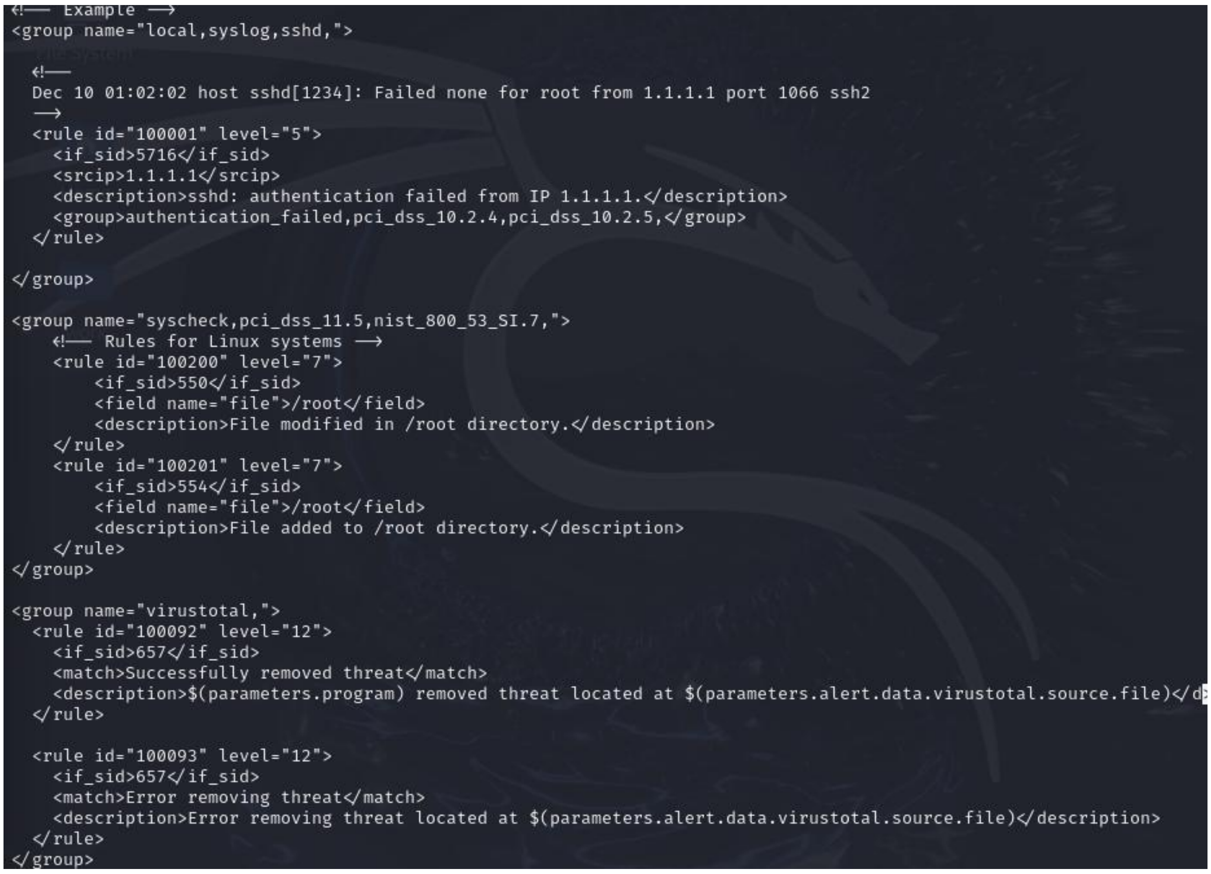
2. **Clone an Existing Rule**: Let's assume you want to detect a custom log line similar to SSH failed login attempts. A relevant default rule for SSH failed logins is rule ID 5710 (from /var/ossec/ruleset/rules/0095-sshd_rules.xml), which detects failed SSH login attempts for non-existent users. We'll clone this rule and modify it.

Example of the original rule (for reference):

```
<rule id="5710" level="5">
  <if_sid>5700</if_sid>
  <match>Invalid user|Failed password</match>
  <description>sshd: Attempt to login using a non-existent user</description>
  <group>syslog,sshd,authentication_failed,invalid_login</group>
</rule>
```

Add a new custom rule to /var/ossec/etc/rules/local_rules.xml. Use a rule ID in the range 100000-120000 for custom rules, as recommended by Wazuh. Here's an example of a cloned and modified rule to detect a custom log line like Failed login attempt for user 'testuser' from 192.168.1.100:

```
<group name="custom,syslog,authentication_failed,">
  <rule id="100001" level="6">
    <if_sid>5700</if_sid>
    <match>Failed login attempt for user</match>
    <regex>Failed login attempt for user '(\w+)' from (\b(?:\d{1,3}\.){3}\d{1,3}\b)</regex>
    <description>Custom: Failed login attempt for user $(dstuser) from $(srcip)</description>
    <group>authentication_failed,invalid_login,custom_rule</group>
  </rule>
</group>
```

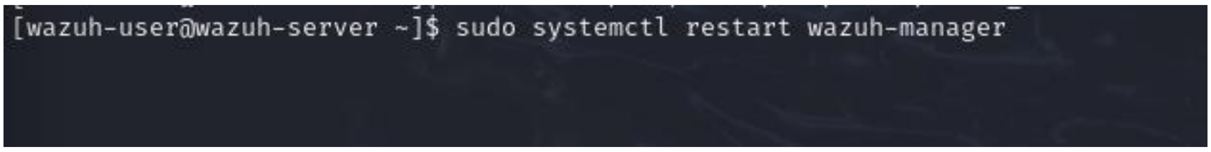


```
<!-- Example -->
<group name="local,syslog,sshd,">

  <!--
  Dec 10 01:02:02 host sshd[1234]: Failed none for root from 1.1.1.1 port 1066 ssh2
  -->
  <rule id="100001" level="5">
    <if_sid>5716</if_sid>
    <srcip>1.1.1.1</srcip>
    <description>sshd: authentication failed from IP 1.1.1.1.</description>
    <group>authentication_failed,pci_dss_10.2.4,pci_dss_10.2.5,</group>
  </rule>

</group>

<group name="syscheck,pci_dss_11.5,nist_800_53_SI.7,">
    <!-- Rules for Linux systems -->
    <rule id="100200" level="7">
        <if_sid>550</if_sid>
        <field name="file">/root</field>
        <description>File modified in /root directory.</description>
    </rule>
    <rule id="100201" level="7">
        <if_sid>554</if_sid>
        <field name="file">/root</field>
        <description>File added to /root directory.</description>
    </rule>
</group>

<group name="virustotal,">
  <rule id="100092" level="12">
    <if_sid>657</if_sid>
    <match>Successfully removed threat</match>
    <description>$(parameters.program) removed threat located at $(parameters.alert.data.virustotal.source.file)</d
  </rule>

  <rule id="100093" level="12">
    <if_sid>657</if_sid>
    <match>Error removing threat</match>
    <description>Error removing threat located at $(parameters.alert.data.virustotal.source.file)</description>
  </rule>
</group>
```

Step 3: Restart Wazuh Manager

To apply the rule and decoder changes, restart the Wazuh manager:

sudo systemctl restart wazuh-manager



```
[wazuh-user@wazuh-server ~]$ sudo systemctl restart wazuh-manager
```

Step 4: Test the Rule with a Custom Log Line

Use the wazuh-logtest tool to simulate a log line and verify that the rule triggers.

1. **Run the Logtest Tool**:

sudo /var/ossec/bin/wazuh-logtest

2. **Input a Custom Log Line**: Enter the following example log line when prompted:

Oct 15 21:07:00 kali-agent sshd[12345]: Failed login attempt for user 'testuser' from 192.168.1.100

Expected output:

**Phase 1: Completed pre-decoding.

full event: 'Oct 15 21:07:00 kali-agent sshd[12345]: Failed login attempt for user 'testuser' from 192.168.1.100'

timestamp: 'Oct 15 21:07:00'

hostname: 'kali-agent'

program_name: 'sshd'

**Phase 2: Completed decoding.

name: 'custom-login'

dstuser: 'testuser'

srcip: '192.168.1.100'

**Phase 3: Completed filtering (rules).

id: '100001'

level: '6'

description: 'Custom: Failed login attempt for user testuser from 192.168.1.100'

groups: '["authentication_failed","invalid_login","custom_rule"]'

firedtimes: '1'

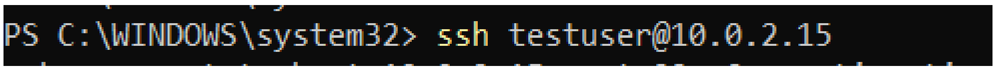
mail: 'false'

Step 5: Trigger the Rule in a Real Scenario

To test the rule with a real log, simulate the log behavior on the Kali Linux system.

1. **Generate a Log Entry**: If your log source is sshd, attempt a failed SSH login to generate a similar log:

ssh testuser@192.168.1.100



```
PS C:\WINDOWS\system32> ssh testuser@10.0.2.15
```

Replace 192.168.1.100 with the actual IP of your Kali system. Enter an incorrect password to simulate a failed login. Note: The actual log format depends on your sshd configuration. If it doesn't match the custom format, adjust the <prematch> and <regex> in the decoder.