Integrate with Threat Intelligence and Malware Detection

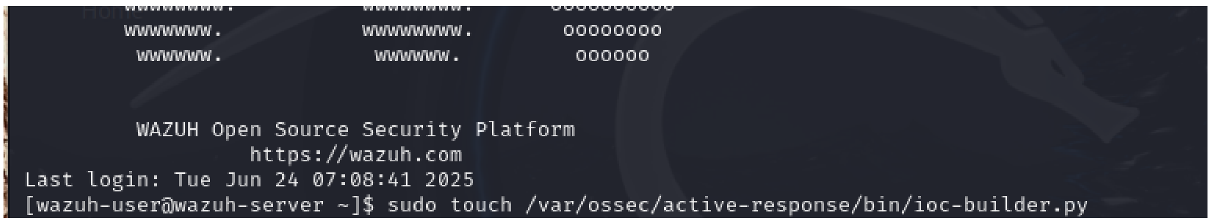
-R.M.Naren Adithya

Building IoC files for threat intelligence

Perform the following steps on the Wazuh server to configure it.

1. Create a Python script, ioc-builder.py, in the /var/ossec/active-response/bin/ directory with the following command:

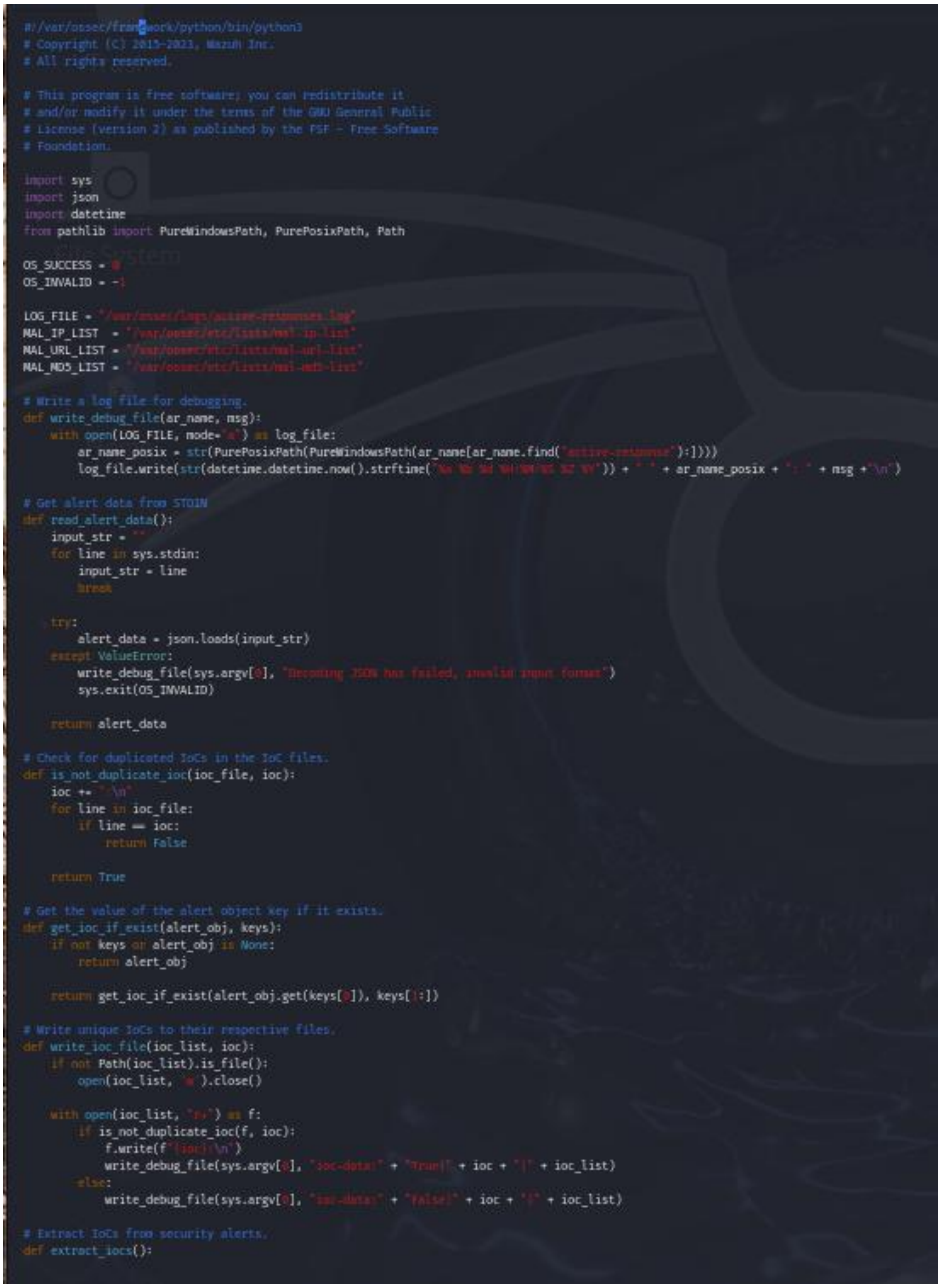
$ sudo touch /var/ossec/active-response/bin/ioc-builder.py



```
        wwwwwww.           wwwwwwww.        oooooooo
         wwwwww.            wwwwww.          oooooo


         WAZUH Open Source Security Platform
                  https://wazuh.com
Last login: Tue Jun 24 07:08:41 2025
[wazuh-user@wazuh-server ~]$ sudo touch /var/ossec/active-response/bin/ioc-builder.py
```

2. Add the following content to the /var/ossec/active-response/bin/ioc-builder.py script file. This script extracts the values of the fields that contain IoCs which Wazuh decodes as srcip, url, md5_after, md5, and id in an alert.

```
#!/var/ossec/framework/python/bin/python3

# Copyright (C) 2015-2023, Wazuh Inc.

# All rights reserved.

# This program is free software; you can redistribute it

# and/or modify it under the terms of the GNU General Public

# License (version 2) as published by the FSF - Free Software

# Foundation.

import sys

import json

import datetime

from pathlib import PureWindowsPath, PurePosixPath, Path

OS_SUCCESS = 0

OS_INVALID = -1

LOG_FILE = "/var/ossec/logs/active-responses.log"

MAL_IP_LIST  = "/var/ossec/etc/lists/mal-ip-list"

MAL_URL_LIST = "/var/ossec/etc/lists/mal-url-list"

MAL_MD5_LIST = "/var/ossec/etc/lists/mal-md5-list"

# Write a log file for debugging.

def write_debug_file(ar_name, msg):
    with open(LOG_FILE, mode="a") as log_file:

        ar_name_posix = str(PurePosixPath(PureWindowsPath(ar_name[ar_name.find("active-response"):])))

        log_file.write(str(datetime.datetime.now().strftime("%a %b %d %H:%M:%S %Z %Y")) + " " + ar_name_posix + ": " + msg +"\n")

# Get alert data from STDIN

def read_alert_data():

    input_str = ""

    for line in sys.stdin:

        input_str = line

        break

    try:

        alert_data = json.loads(input_str)

    except ValueError:

        write_debug_file(sys.argv[0], "Decoding JSON has failed, invalid input format")

        sys.exit(OS_INVALID)

    return alert_data

# Check for duplicated IoCs in the IoC files.

def is_not_duplicate_ioc(ioc_file, ioc):

    ioc += ":\n"

    for line in ioc_file:

        if line == ioc:

            return False

    return True

# Get the value of the alert object key if it exists.

def get_ioc_if_exist(alert_obj, keys):

    if not keys or alert_obj is None:

        return alert_obj

    return get_ioc_if_exist(alert_obj.get(keys[0]), keys[1:])

# Write unique IoCs to their respective files.

def write_ioc_file(ioc_list, ioc):

    if not Path(ioc_list).is_file():

        open(ioc_list, 'w').close()

    with open(ioc_list, "r+") as f:

        if is_not_duplicate_ioc(f, ioc):

            f.write(f"{ioc}:\n")

            write_debug_file(sys.argv[0], "ioc-data:" + "True|" + ioc + "|" + ioc_list)

        else:

            write_debug_file(sys.argv[0], "ioc-data:" + "False|" + ioc + "|" + ioc_list)

# Extract IoCs from security alerts.

def extract_iocs():

    alert_obj = read_alert_data()

    # Uniquely add the url to mal-url-list.

    url = get_ioc_if_exist(alert_obj, ["parameters", "alert", "data", "url"])

    if url is not None:

        write_ioc_file(MAL_URL_LIST, url)
    # Uniquely add the srcip to mal-ip-list.

    srcip = get_ioc_if_exist(alert_obj, ["parameters", "alert", "data", "srcip"])

    if srcip is not None:

        write_ioc_file(MAL_IP_LIST , srcip)

    # Uniquely add the file hash to mal-md5-list.

    md5_after = get_ioc_if_exist(alert_obj, ["parameters", "alert", "syscheck", "md5_after"])

    if md5_after is not None: # For FIM alert format.

        write_ioc_file(MAL_MD5_LIST, md5_after)

    md5 = get_ioc_if_exist(alert_obj, ["parameters", "alert", "data", "virustotal", "source", "md5"])

    if md5 is not None: # For VT alert format.

        write_ioc_file(MAL_MD5_LIST, md5)

    id = get_ioc_if_exist(alert_obj, ["parameters", "alert", "data", "id"])

    if id is not None and len(id) == 32: # For ClamAV alert format. The id field represents the md5 hash.

        write_ioc_file(MAL_MD5_LIST, id)

def main(argv):

    write_debug_file(argv[0], "Started")

    extract_iocs()

    write_debug_file(argv[0], "Ended")

    sys.exit(OS_SUCCESS)

if __name__ == "__main__":

    main(sys.argv)
```

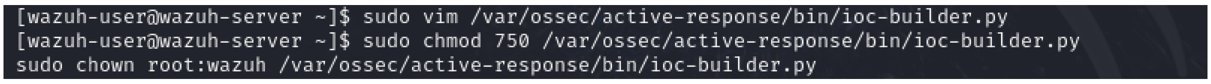


```python
#!/var/ossec/framework/python/bin/python3
# Copyright (C) 2015-2023, Wazuh Inc.
# All rights reserved.

# This program is free software; you can redistribute it
# and/or modify it under the terms of the GNU General Public
# License (version 2) as published by the FSF - Free Software
# Foundation.

import sys
import json
import datetime
from pathlib import PureWindowsPath, PurePosixPath, Path

OS_SUCCESS = 0
OS_INVALID = -1

LOG_FILE = "/var/ossec/logs/active-responses.log"
MAL_IP_LIST  = "/var/ossec/etc/lists/mal-ip-list"
MAL_URL_LIST = "/var/ossec/etc/lists/mal-url-list"
MAL_MD5_LIST = "/var/ossec/etc/lists/mal-md5-list"

# Write a log file for debugging.
def write_debug_file(ar_name, msg):
    with open(LOG_FILE, mode="a") as log_file:
        ar_name_posix = str(PurePosixPath(PureWindowsPath(ar_name[ar_name.find("active-response"):])))
        log_file.write(str(datetime.datetime.now().strftime('%a %b %d %H:%M:%S %Z %Y')) + " " + ar_name_posix + ": " + msg +"\n")

# Get alert data from STDIN
def read_alert_data():
    input_str = ""
    for line in sys.stdin:
        input_str = line
        break

    try:
        alert_data = json.loads(input_str)
    except ValueError:
        write_debug_file(sys.argv[0], "Decoding JSON has failed, invalid input format")
        sys.exit(OS_INVALID)

    return alert_data

# Check for duplicated IoCs in the IoC files.
def is_not_duplicate_ioc(ioc_file, ioc):
    ioc += ":\n"
    for line in ioc_file:
        if line == ioc:
            return False

    return True

# Get the value of the alert object key if it exists.
def get_ioc_if_exist(alert_obj, keys):
    if not keys or alert_obj is None:
        return alert_obj

    return get_ioc_if_exist(alert_obj.get(keys[0]), keys[1:])

# Write unique IoCs to their respective files.
def write_ioc_file(ioc_list, ioc):
    if not Path(ioc_list).is_file():
        open(ioc_list, 'w').close()

    with open(ioc_list, "r+") as f:
        if is_not_duplicate_ioc(f, ioc):
            f.write(f"{ioc}:\n")
            write_debug_file(sys.argv[0], "ioc-data: " + "True|" + ioc + "|" + ioc_list)
        else:
            write_debug_file(sys.argv[0], "ioc-data: " + "False|" + ioc + "|" + ioc_list)

# Extract IoCs from security alerts.
def extract_iocs():
```

3. Change the permissions and ownership of the ioc-builder.py script to 750 and root:wazuh respectively using the following commands:

$ sudo chmod 750 /var/ossec/active-response/bin/ioc-builder.py

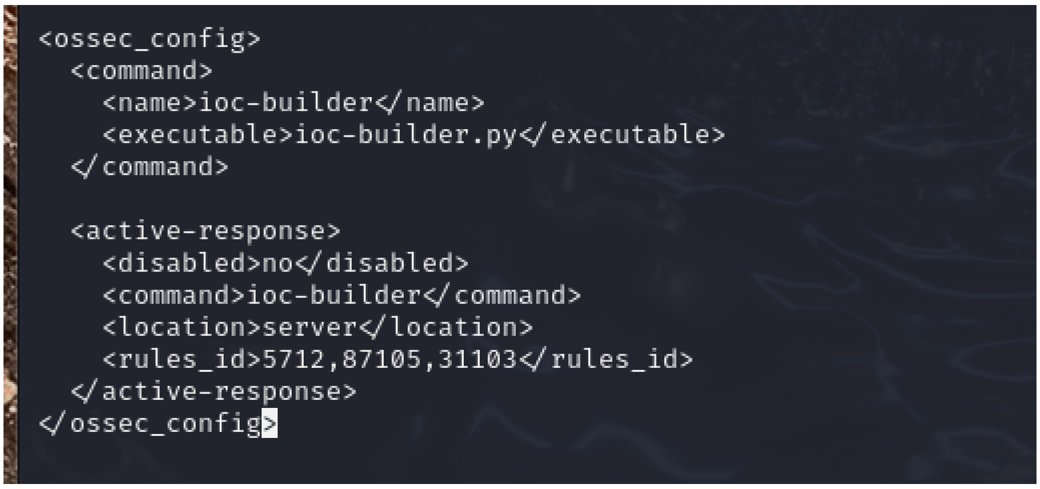
$ sudo chown root:wazuh /var/ossec/active-response/bin/ioc-builder.py



```
[wazuh-user@wazuh-server ~]$ sudo vim /var/ossec/active-response/bin/ioc-builder.py
[wazuh-user@wazuh-server ~]$ sudo chmod 750 /var/ossec/active-response/bin/ioc-builder.py
sudo chown root:wazuh /var/ossec/active-response/bin/ioc-builder.py
```

4. Append the following configuration to the /var/ossec/etc/ossec.conf file enabling the Wazuh active response module to automatically execute the ioc-builder.py script. This automatic execution only occurs after an endpoint triggers a rule with an ID defined in the <rules_id> tag as highlighted in the configuration below.

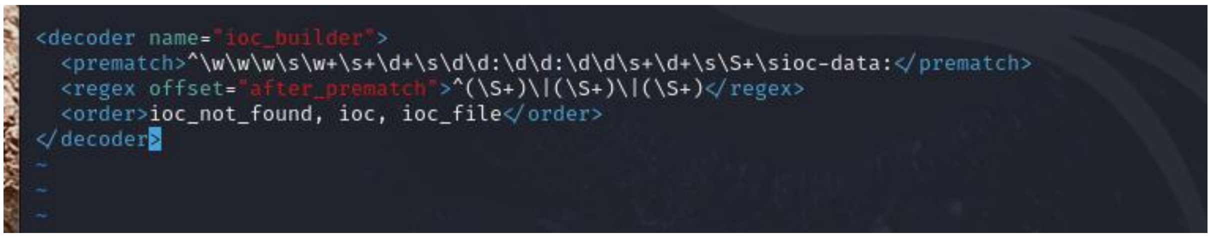
```
<ossec_config>
  <command>
    <name>ioc-builder</name>
    <executable>ioc-builder.py</executable>
  </command>

  <active-response>
    <disabled>no</disabled>
    <command>ioc-builder</command>
    <location>server</location>
    <rules_id>5712,87105,31103</rules_id>
  </active-response>
</ossec_config>
```



```
<ossec_config>
  <command>
    <name>ioc-builder</name>
    <executable>ioc-builder.py</executable>
  </command>

  <active-response>
    <disabled>no</disabled>
    <command>ioc-builder</command>
    <location>server</location>
    <rules_id>5712,87105,31103</rules_id>
  </active-response>
</ossec_config>
```

5. Append the following decoder to the /var/ossec/etc/decoders/local_decoder.xml file to extract the fields in the log event produced by the ioc-builder.py script:

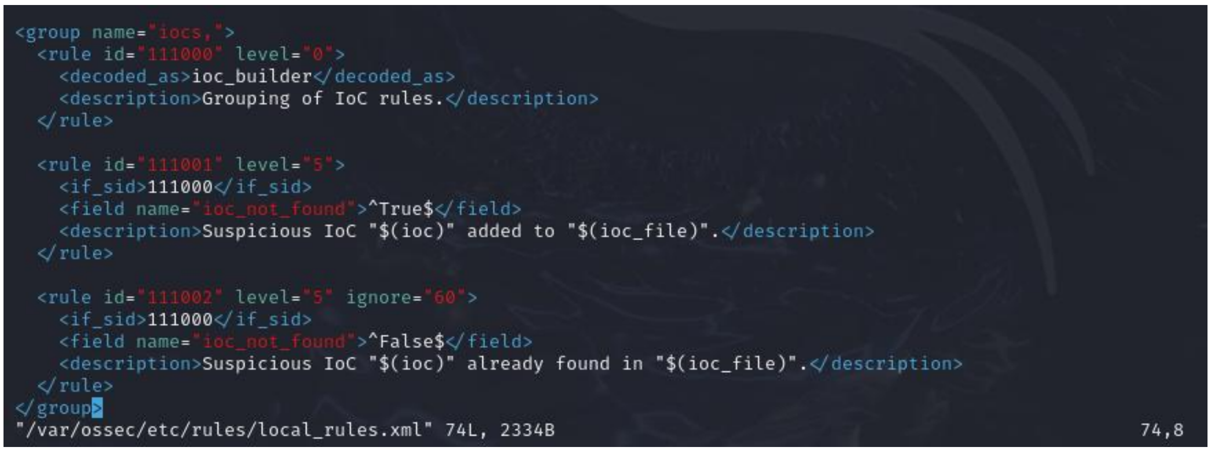
```
<decoder name="ioc_builder">
  <prematch>^\w\w\w\s\w+\s+\d+\s\d\d:\d\d:\d\d\s+\d+\s\S+\sioc-data:</prematch>
  <regex offset="after_prematch">^(\S+)\|(\S+)\|(\S+)</regex>

  <order>ioc_not_found, ioc, ioc_file</order>

</decoder>
```



```
<decoder name="ioc_builder">
  <prematch>^\w\w\w\s\w+\s+\d+\s\d\d:\d\d:\d\d\s+\d+\s\S+\sioc-data:</prematch>
  <regex offset="after_prematch">^(\S+)\|(\S+)\|(\S+)</regex>
  <order>ioc_not_found, ioc, ioc_file</order>
</decoder>
~
~
~
```

6. Append the following rules to the /var/ossec/etc/rules/local_rules.xml file. Wazuh triggers the rules when it verifies or adds an entry into the appropriate IoC files.

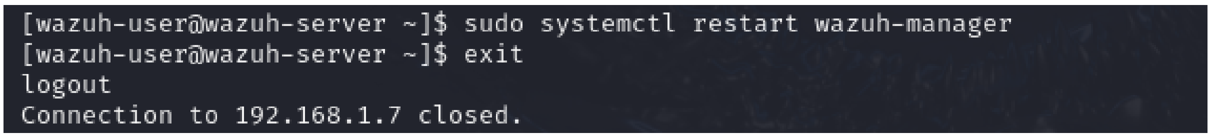
```
<group name="iocs,">

 <rule id="111000" level="0">

   <decoded_as>ioc_builder</decoded_as>

   <description>Grouping of IoC rules.</description>

 </rule>

 <rule id="111001" level="5">

   <if_sid>111000</if_sid>

   <field name="ioc_not_found">^True$</field>

   <description>Suspicious IoC "$(ioc)" added to "$(ioc_file)".</description>

 </rule>

 <rule id="111002" level="5" ignore="60">

   <if_sid>111000</if_sid>

   <field name="ioc_not_found">^False$</field>

   <description>Suspicious IoC "$(ioc)" already found in "$(ioc_file)".</description>

 </rule>

</group>
```



```
<group name="iocs,">
  <rule id="111000" level="0">
    <decoded_as>ioc_builder</decoded_as>
    <description>Grouping of IoC rules.</description>
  </rule>

  <rule id="111001" level="5">
    <if_sid>111000</if_sid>
    <field name="ioc_not_found">^True$</field>
    <description>Suspicious IoC "$(ioc)" added to "$(ioc_file)".</description>
  </rule>

  <rule id="111002" level="5" ignore="60">
    <if_sid>111000</if_sid>
    <field name="ioc_not_found">^False$</field>
    <description>Suspicious IoC "$(ioc)" already found in "$(ioc_file)".</description>
  </rule>
</group>
"/var/ossec/etc/rules/local_rules.xml" 74L, 2334B                                    74,8
```

7. Restart the Wazuh manager to apply the configuration changes:

$ sudo systemctl restart wazuh-manager



```
[wazuh-user@wazuh-server ~]$ sudo systemctl restart wazuh-manager
[wazuh-user@wazuh-server ~]$ exit
logout
Connection to 192.168.1.7 closed.
```

Attack emulation

Malware attack

Configuring the FIM module and VirusTotal integration

Ubuntu endpoint

Perform the following steps to configure FIM to monitor for file changes.

1. Append the following configuration to the /var/ossec/etc/ossec.conf file to monitor the /home directory for file changes in real-time:

2. Restart the Wazuh agent to apply the configuration changes:

$ sudo systemctl restart wazuh-agent

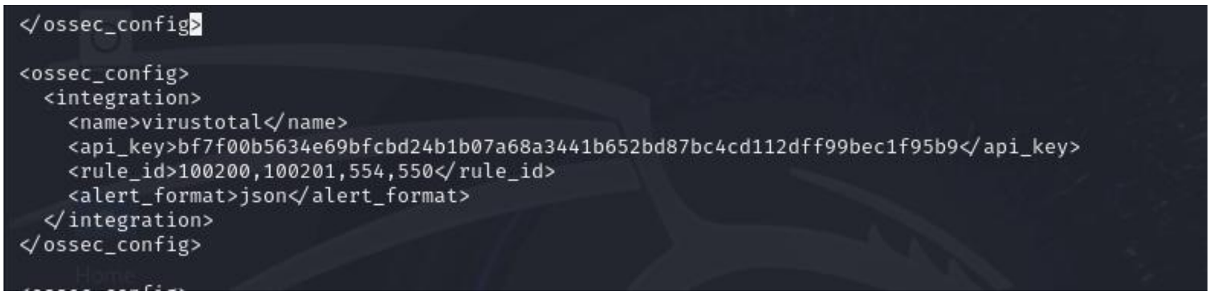
Wazuh server

Perform the following steps to configure the VirusTotal integration to scan for malicious files.

1. Follow the instructions from the VirusTotal API key page to obtain a free key for scanning files if you don’t have one already.

2. Append the following settings to the /var/ossec/etc/ossec.conf file and replace <YOUR_VIRUS_TOTAL_API_KEY> with your VirusTotal API key obtained in step 1 above:

```
<ossec_config>
  <integration>
    <name>virustotal</name>
    <api_key><YOUR_VIRUS_TOTAL_API_KEY></api_key>
    <rule_id>554,550</rule_id>
    <alert_format>json</alert_format>
  </integration>
</ossec_config>
```



```
</ossec_config>

<ossec_config>
  <integration>
    <name>virustotal</name>
    <api_key>bf7f00b5634e69bfcbd24b1b07a68a3441b652bd87bc4cd112dff99bec1f95b9</api_key>
    <rule_id>100200,100201,554,550</rule_id>
    <alert_format>json</alert_format>
  </integration>
</ossec_config>
```

3. Restart the Wazuh manager to apply the configuration changes:

$ sudo systemctl restart wazuh-manager

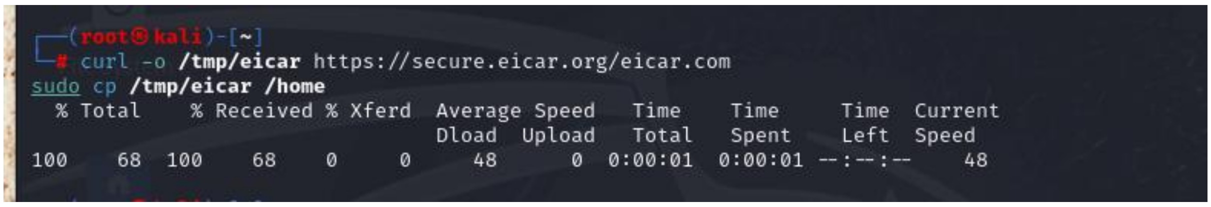
Launching the attack

For testing purposes, we download the EICAR anti-malware test file as shown below. We recommend testing in a sandbox, not in a production environment.

1. Download the malware, eicar, to the /tmp directory and copy it to the /home directory of the Ubuntu endpoint using the following commands:

$ curl -o /tmp/eicar https://secure.eicar.org/eicar.com

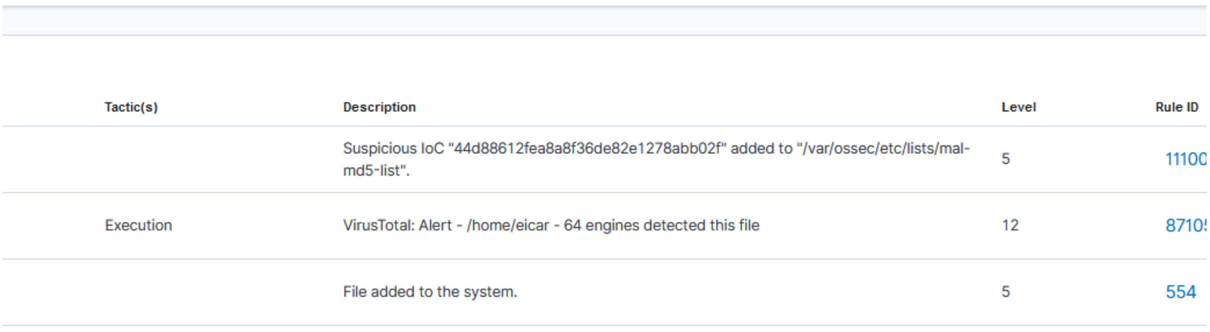
$ sudo cp /tmp/eicar /home



```
┌──(root㉿kali)-[~]
└─# curl -o /tmp/eicar https://secure.eicar.org/eicar.com
sudo cp /tmp/eicar /home
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    68  100    68    0     0     48      0  0:00:01  0:00:01 --:--:--    48
```

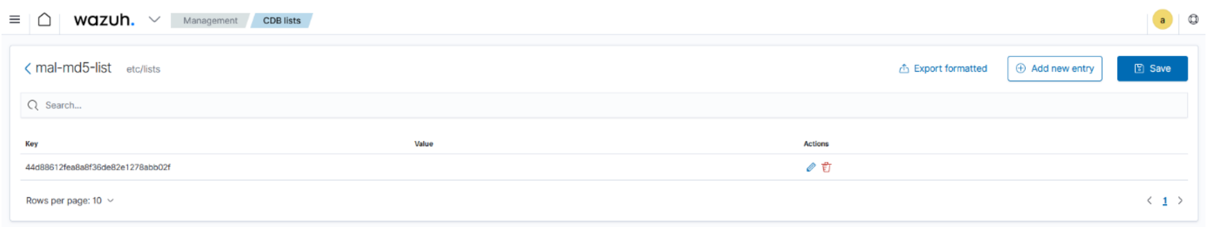
Perform the following steps on the Wazuh dashboard to view the generated alerts and stored IoCs.

1. Navigate to Modules > Security events tab to view the generated alerts.



Tactic(s)	Description	Level	Rule ID
	Suspicious IoC "44d88612fea8a8f36de82e1278abb02f" added to "/var/ossec/etc/lists/mal-md5-list".	5	1110
Execution	VirusTotal: Alert - /home/eicar - 64 engines detected this file	12	8710
	File added to the system.	5	554

2. Navigate to Management > CDB lists and click the mal-md5-list CDB list to see the stored suspicious MD5 hash of the eicar malware.



Key	Value	Actions
44d88612fea8a8f36de82e1278abb02f		

Detecting and removing malware using VirusTotal integration

Configuration for the Ubuntu endpoint

Configure your environment as follows to test the use case for the Ubuntu endpoint. These steps work for other Linux distributions as well.

Ubuntu endpoint

Perform the following steps to configure Wazuh to monitor near real-time changes in the /root directory of the Ubuntu endpoint. These steps also install the necessary packages and create the active response script that removes malicious files.

1. Search for the <syscheck> block in the Wazuh agent configuration file /var/ossec/etc/ossec.conf. Make sure that <disabled> is set to no. This enables the Wazuh FIM to monitor for directory changes.

2. Add an entry within the <syscheck> block to configure a directory to be monitored in near real-time. In this case, you are monitoring the /root directory:

   ```
   <directories realtime="yes">/root</directories>
   ```

3. Install jq, a utility that processes JSON input from the active response script.

   ```
   $ sudo apt update
   ```

   ```
   $ sudo apt -y install jq
   ```

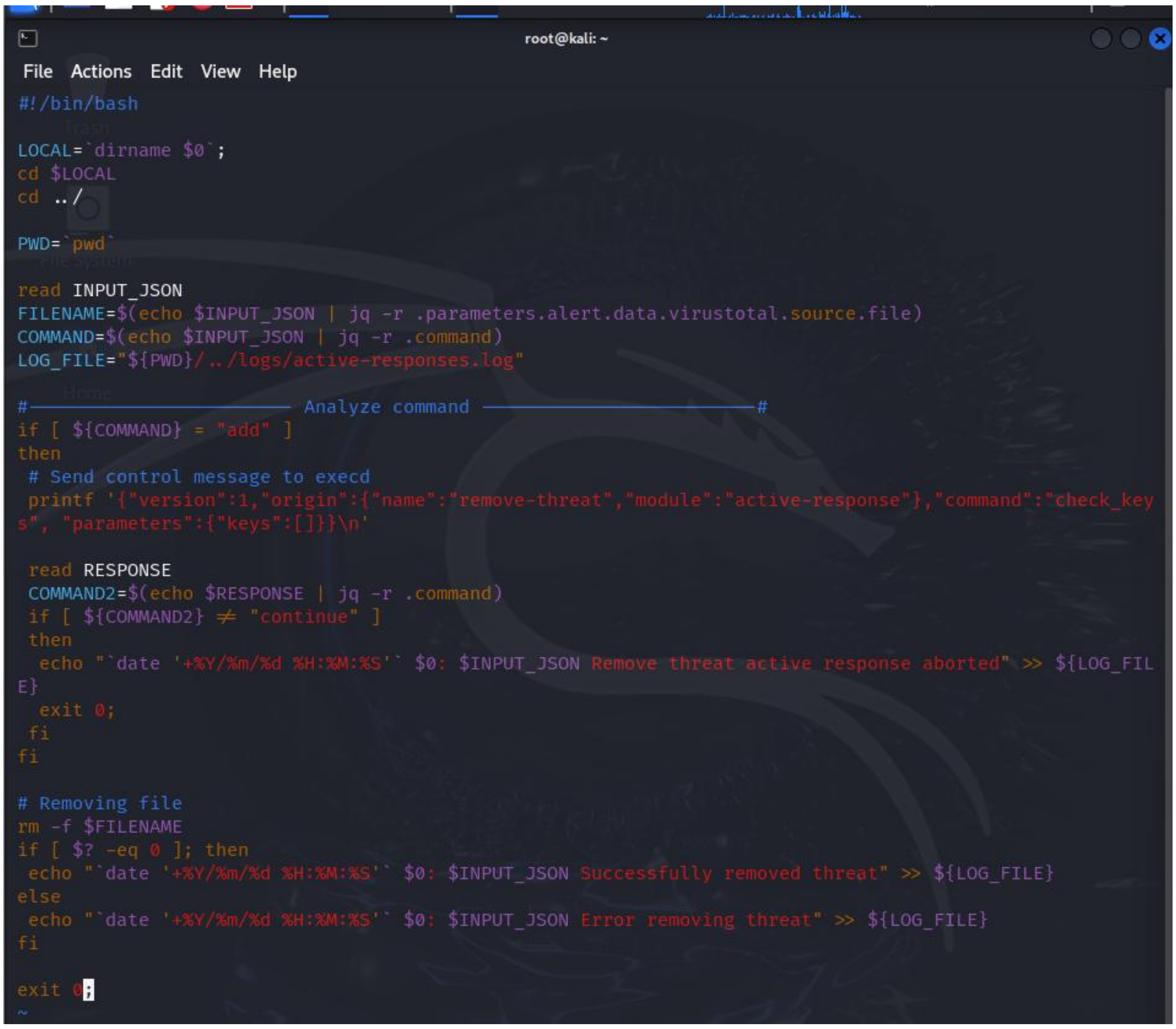
4. Create the /var/ossec/active-response/bin/remove-threat.sh active response script to remove malicious files from the endpoint:

   ```
   #!/bin/bash
   ```

```
LOCAL=`dirname $0`;
cd $LOCAL
cd ../

PWD=`pwd`

read INPUT_JSON
FILENAME=$(echo $INPUT_JSON | jq -r .parameters.alert.data.virustotal.source.file)
COMMAND=$(echo $INPUT_JSON | jq -r .command)
LOG_FILE="${PWD}/../logs/active-responses.log"

#------------------------ Analyze command -------------------------#
if [ ${COMMAND} = "add" ]
then
 # Send control message to execd
 printf '{"version":1,"origin":{"name":"remove-threat","module":"active-
response"},"command":"check_keys", "parameters":{"keys":[]}}\n'

 read RESPONSE
 COMMAND2=$(echo $RESPONSE | jq -r .command)
 if [ ${COMMAND2} != "continue" ]
 then
  echo "`date '+%Y/%m/%d %H:%M:%S'` $0: $INPUT_JSON Remove threat active response aborted" >>
${LOG_FILE}
  exit 0;
 fi
fi

# Removing file
rm -f $FILENAME
if [ $? -eq 0 ]; then
 echo "`date '+%Y/%m/%d %H:%M:%S'` $0: $INPUT_JSON Successfullyremoved threat" >> ${LOG_FILE}
else
 echo "`date '+%Y/%m/%d %H:%M:%S'` $0: $INPUT_JSON Error removing threat" >> ${LOG_FILE}
```

fi

exit 0;



root@kali: ~

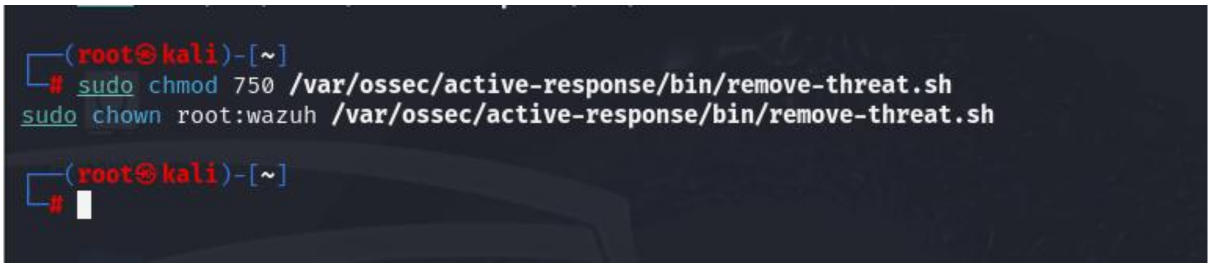
File Actions Edit View Help

```
#!/bin/bash

LOCAL=`dirname $0`;
cd $LOCAL
cd ../

PWD=`pwd`

read INPUT_JSON
FILENAME=$(echo $INPUT_JSON | jq -r .parameters.alert.data.virustotal.source.file)
COMMAND=$(echo $INPUT_JSON | jq -r .command)
LOG_FILE="${PWD}/../logs/active-responses.log"

#------------------------ Analyze command -------------------------#
if [ ${COMMAND} = "add" ]
then
 # Send control message to execd
 printf '{"version":1,"origin":{"name":"remove-threat","module":"active-response"},"command":"check_key
s", "parameters":{"keys":[]}}\n'

 read RESPONSE
 COMMAND2=$(echo $RESPONSE | jq -r .command)
 if [ ${COMMAND2} ≠ "continue" ]
 then
  echo "`date '+%Y/%m/%d %H:%M:%S'` $0: $INPUT_JSON Remove threat active response aborted" >> ${LOG_FIL
E}
  exit 0;
 fi
fi

# Removing file
rm -f $FILENAME
if [ $? -eq 0 ]; then
 echo "`date '+%Y/%m/%d %H:%M:%S'` $0: $INPUT_JSON Successfully removed threat" >> ${LOG_FILE}
else
 echo "`date '+%Y/%m/%d %H:%M:%S'` $0: $INPUT_JSON Error removing threat" >> ${LOG_FILE}
fi

exit 0;
~
```

5. Change the /var/ossec/active-response/bin/remove-threat.sh file ownership, and permissions:

   $ sudo chmod 750 /var/ossec/active-response/bin/remove-threat.sh

   $ sudo chown root:wazuh /var/ossec/active-response/bin/remove-threat.sh



```
┌──(root㉿kali)-[~]
└─# sudo chmod 750 /var/ossec/active-response/bin/remove-threat.sh
sudo chown root:wazuh /var/ossec/active-response/bin/remove-threat.sh

┌──(root㉿kali)-[~]
└─# 
```

6. Restart the Wazuh agent to apply the changes:

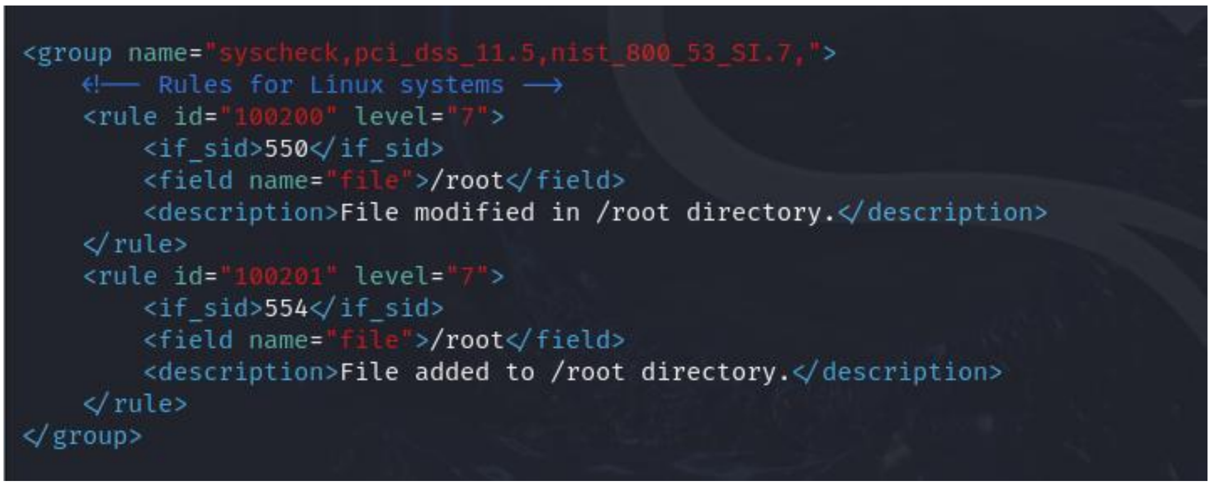
   $ sudo systemctl restart wazuh-agent

Wazuh server

Perform the following steps on the Wazuh server to alert for changes in the endpoint directory and enable the VirusTotal integration. These steps also enable and trigger the active response script whenever a suspicious file is detected.

1. Add the following rules to the /var/ossec/etc/rules/local_rules.xml file on the Wazuh server. These rules alert about changes in the /root directory that are detected by FIM scans:

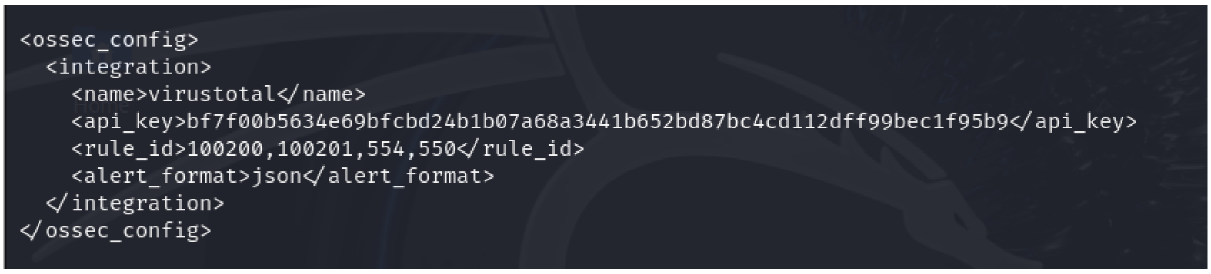
```
<group name="syscheck,pci_dss_11.5,nist_800_53_SI.7,">
  <!-- Rules for Linux systems -->
  <rule id="100200" level="7">
    <if_sid>550</if_sid>
    <field name="file">/root</field>
    <description>File modified in /root directory.</description>
  </rule>
  <rule id="100201" level="7">
    <if_sid>554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



```
<group name="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>
```

2. Add the following configuration to the Wazuh server /var/ossec/etc/ossec.conf file to enable the Virustotal integration. Replace <YOUR_VIRUS_TOTAL_API_KEY> with your VirusTotal API key. This allows to trigger a VirusTotal query whenever any of the rules 100200 and 100201 are triggered:

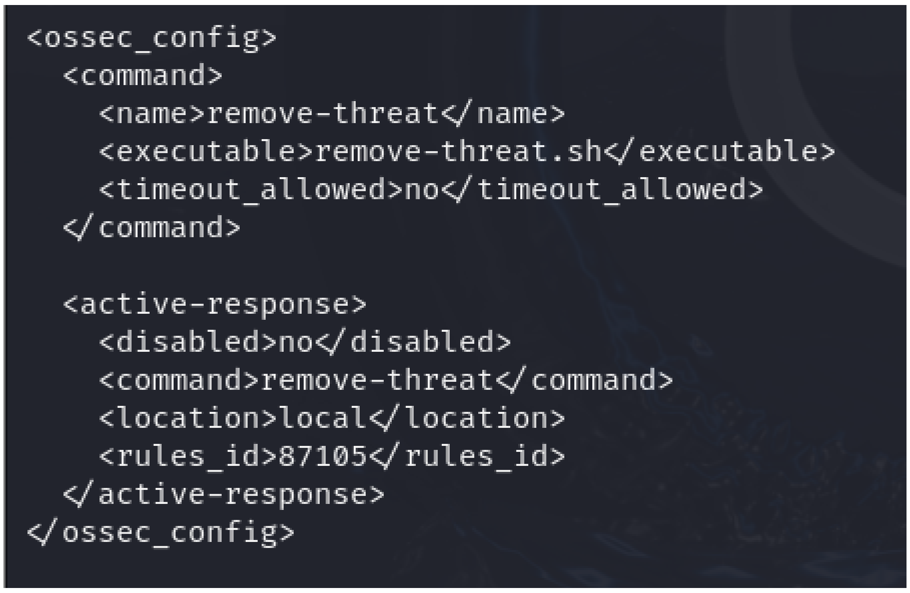
```
<ossec_config>
  <integration>
    <name>virustotal</name>
    <api_key><YOUR_VIRUS_TOTAL_API_KEY></api_key> <!-- Replace with your VirusTotal API key -->
    <rule_id>100200,100201</rule_id>
    <alert_format>json</alert_format>
  </integration>
</ossec_config>
```



```
<ossec_config>
  <integration>
    <name>virustotal</name>
    <api_key>bf7f00b5634e69bfcbd24b1b07a68a3441b652bd87bc4cd112dff99bec1f95b9</api_key>
    <rule_id>100200,100201,554,550</rule_id>
    <alert_format>json</alert_format>
  </integration>
</ossec_config>
```

3. Append the following blocks to the Wazuh server /var/ossec/etc/ossec.conf file. This enables Active Response and triggers the remove-threat.sh script when VirusTotal flags a file as malicious:

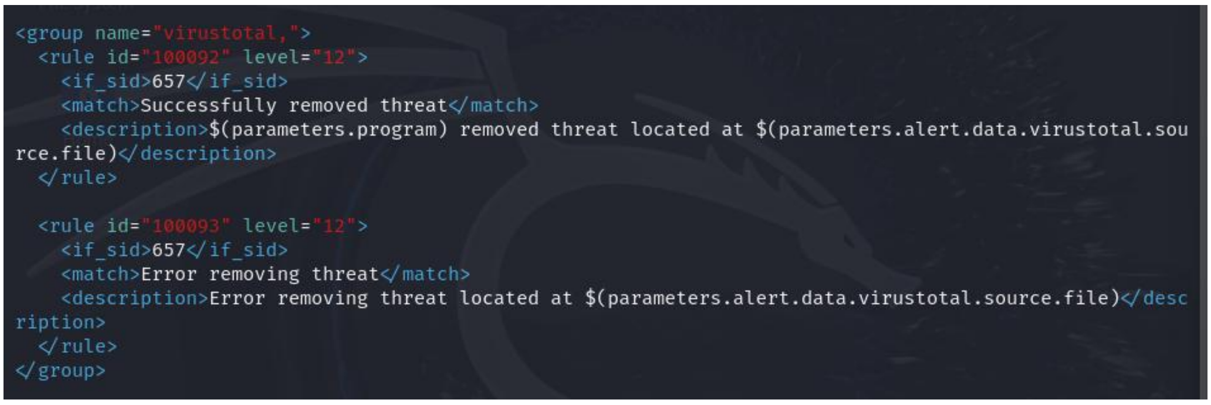
```
<ossec_config>
  <command>
    <name>remove-threat</name>
    <executable>remove-threat.sh</executable>
    <timeout_allowed>no</timeout_allowed>
  </command>

  <active-response>
    <disabled>no</disabled>
    <command>remove-threat</command>

    <location>local</location>

    <rules_id>87105</rules_id>

  </active-response>

</ossec_config>
```



```
<ossec_config>
  <command>
    <name>remove-threat</name>
    <executable>remove-threat.sh</executable>
    <timeout_allowed>no</timeout_allowed>
  </command>

  <active-response>
    <disabled>no</disabled>
    <command>remove-threat</command>
    <location>local</location>
    <rules_id>87105</rules_id>
  </active-response>
</ossec_config>
```

4. Add the following rules to the Wazuh server /var/ossec/etc/rules/local_rules.xml file to alert about the Active Response results:

```
<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)</description>

  </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>
```


```
<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.sou
rce.file)</description>
  </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)</desc
ription>
  </rule>
</group>
```

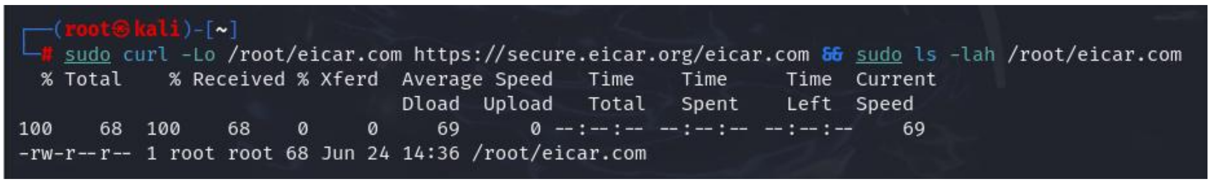
5. Restart the Wazuh manager to apply the configuration changes:

   $ sudo systemctl restart wazuh-manager

Attack emulation

1. Download an EICAR test file to the /root directory on the Ubuntu endpoint:

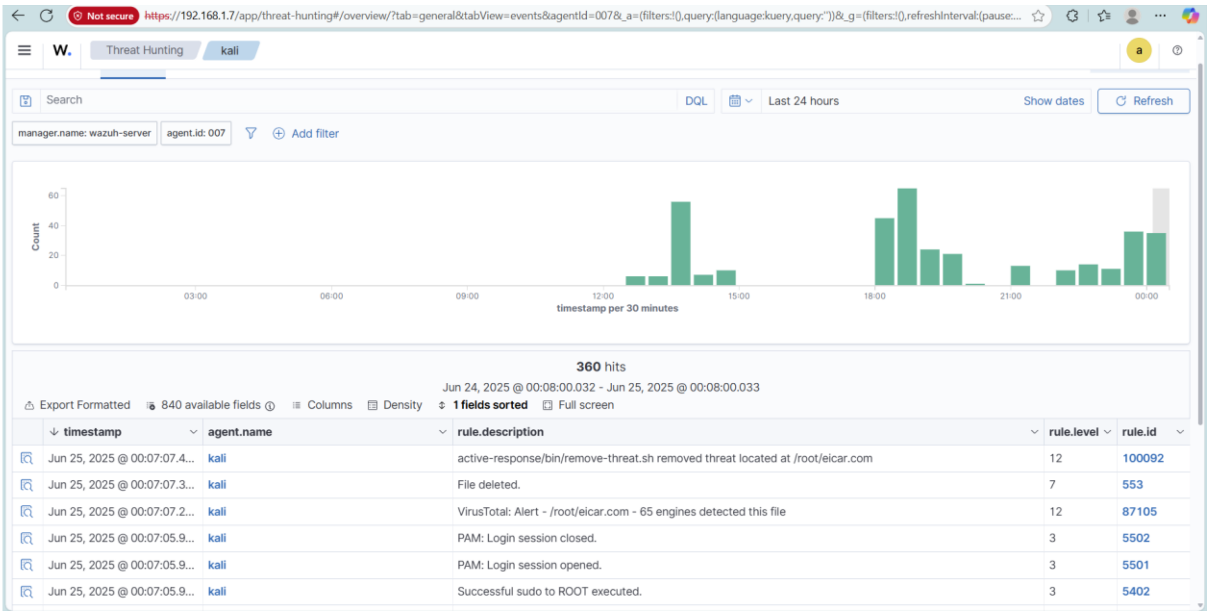
   $ sudo curl -Lo /root/eicar.com https://secure.eicar.org/eicar.com && sudo ls -lah /root/eicar.com


```
┌──(root㉿kali)-[~]
└─# sudo curl -Lo /root/eicar.com https://secure.eicar.org/eicar.com && sudo ls -lah /root/eicar.com
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    68  100    68    0     0     69      0 --:--:-- --:--:-- --:--:--    69
-rw-r--r-- 1 root root 68 Jun 24 14:36 /root/eicar.com
```

Visualize the alerts

You can visualize the alert data in the Wazuh dashboard. To do this, go to the **Threat Hunting** module and add the filters in the search bar to query the alerts.

- Linux - rule.id: is one of 553,100092,87105,100201



360 hits

Jun 24, 2025 @ 00:08:00.032 - Jun 25, 2025 @ 00:08:00.033

timestamp	agent.name	rule.description	rule.level	rule.id
Jun 25, 2025 @ 00:07:07.4...	kali	active-response/bin/remove-threat.sh removed threat located at /root/eicar.com	12	100092
Jun 25, 2025 @ 00:07:07.3...	kali	File deleted.	7	553
Jun 25, 2025 @ 00:07:07.2...	kali	VirusTotal: Alert - /root/eicar.com - 65 engines detected this file	12	87105
Jun 25, 2025 @ 00:07:05.9...	kali	PAM: Login session closed.	3	5502
Jun 25, 2025 @ 00:07:05.9...	kali	PAM: Login session opened.	3	5501
Jun 25, 2025 @ 00:07:05.9...	kali	Successful sudo to ROOT executed.	3	5402