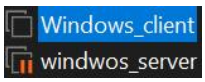
Security Engineering and Security Monitoring

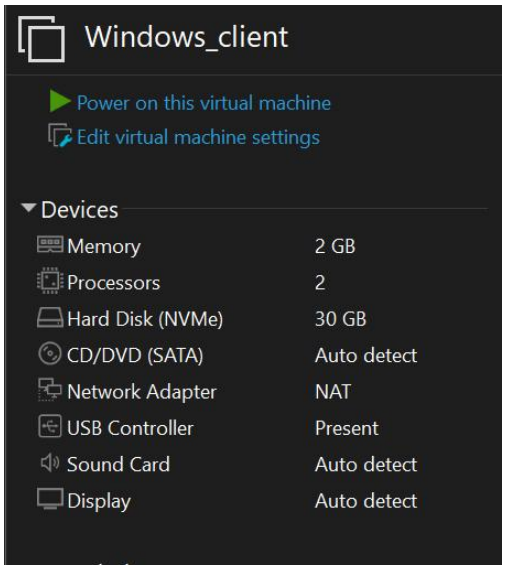
Date: 29.03.2024
Project: Security Engineering and Security Monitoring
Author: Sanan Rahimli, Sabir Eyvazlı

Table of Contents

Windows client and server deployment

We deployed Windows 10 pro as a client machine and Windows Server 2019 as a domain controller over local virtualization environment VMWare Workstation pro.



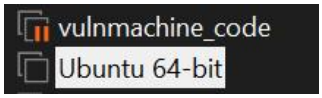


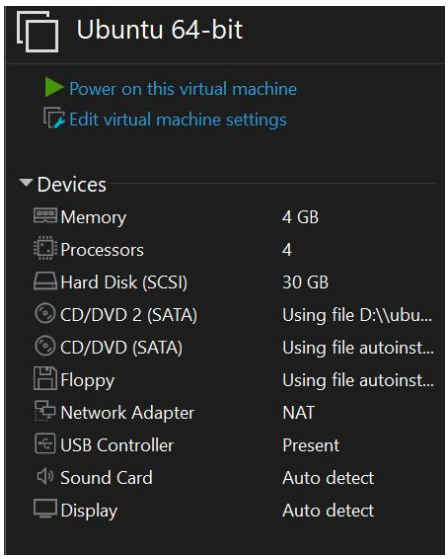
We raised up domain controller over Windows server 2019 and added Windows client machine to the domain controller environment. Note: These two are the just examples, totally we have 2 Active directory environment seperatly. One of them raised up in Sabir's computer and another one raised up in Sanan's computer. Then we both created users.

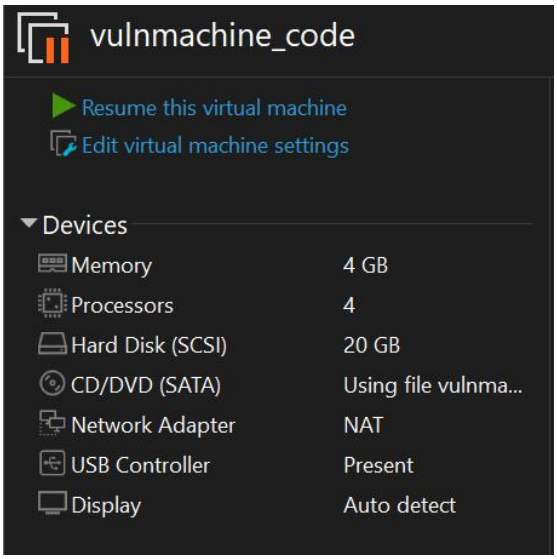
Again these users are the examples.

Ubuntu based vulnerable machine deployment

We deployed Ubuntu 22.04 LTS and DVWA for web vulnerabilities. And we also used vulnerable machine which has given to us for penetration testing. At the end of the penetration testing phase we gained persistence by adding SSH key to the .ssh directory under home directory of root user. And then connected to the machine as a root user via SSH and then we added Wazuh agent to the both of them. Note: Again we have our lab environment separately therefore you will see there're more agent's in the list of Wazuh agents in the next pages.

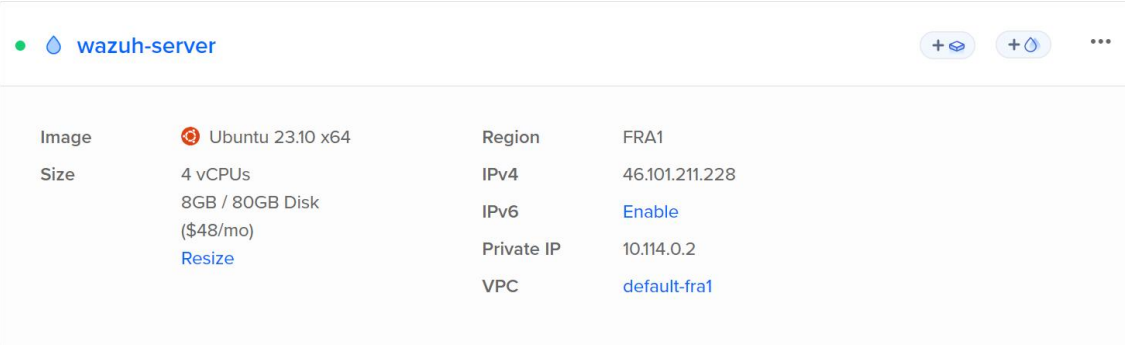






Wazuh Installation

We deployed Wazuh over Digital Ocean Cloud platform.
Here is the ip address of the instance. You can connect the instance via ssh.
Password is: !123Salam
URL: https://46.101.211.228 Username: admin Password: SecretPassword



Clone the Wazuh repository to your system:

git clone https://github.com/wazuh/wazuh-docker.git -b v4.7.3

Then enter into the single-node directory to execute all the commands described below within this directory.
Execute the following command to get the desired certificates:

docker-compose -f generate-indexer-certs.yml run --rm generator

This saves the certificates into the config/wazuh_indexer_ssl_certs directory.
Start the Wazuh single-node deployment using docker-compose:
Background:

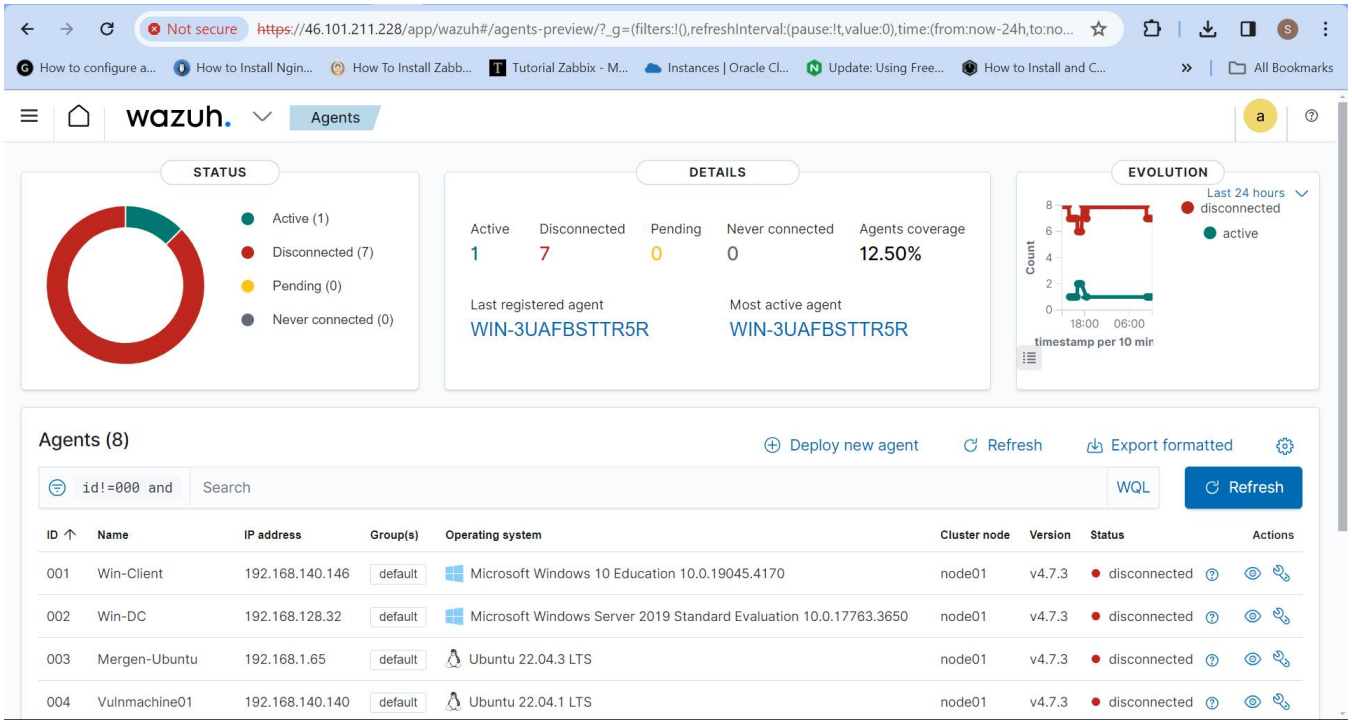
docker-compose up –d

```
root@wazuh-server:~# docker ps
CONTAINER ID   IMAGE                          COMMAND                  CREATED       STATUS       PORTS
                 NAMES
ea2356a44b43   wazuh/wazuh-dashboard:4.7.3    "/entrypoint.sh"         2 weeks ago   Up 12 days   443/tcp, 0.0.0.0:443->5601/tcp, :::443->5601/tcp
                 single-node_wazuh.dashboard_1
5d0f57950927   wazuh/wazuh-manager:4.7.3      "/init"                  2 weeks ago   Up 12 days   0.0.0.0:1514-1515->1514-1515/tcp, :::1514-1515->1514-1515/tcp, 0.0.0.0:514->514/udp, :::514->514/udp, 0.0.0.0:55000->55000/tcp, :::55000->55000/tcp, 1516/tcp   single-node_wazuh.manager_1
a338d6e09a6d   wazuh/wazuh-indexer:4.7.3      "/entrypoint.sh open…"   2 weeks ago   Up 12 days   0.0.0.0:9200->9200/tcp, :::9200->9200/tcp
                 single-node_wazuh.indexer_1
```

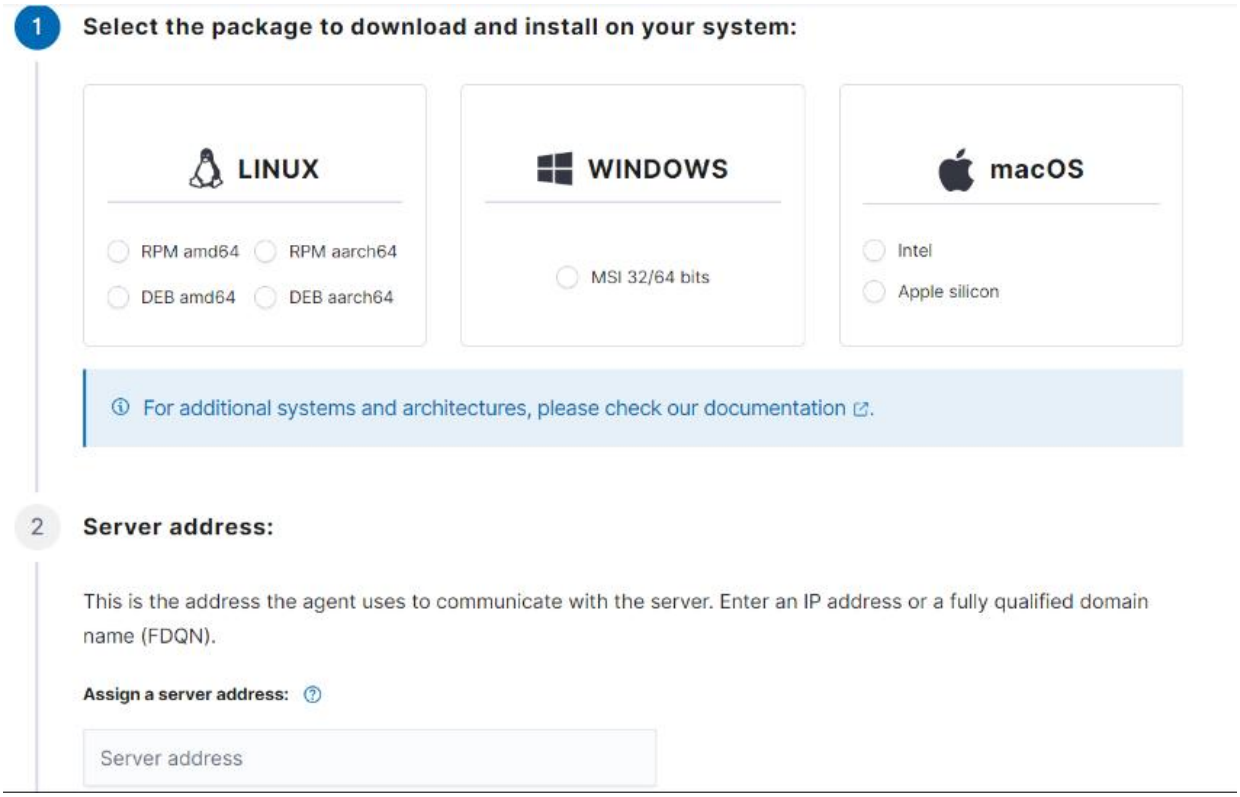
Use this command to enter inside of the Docker.

docker exec –it dockername bash

```
root@wazuh-server:~# docker exec -it single-node_wazuh.manager_1 bash
root@wazuh:/# whoami
root
```



Wazuh Agent generation and installing agent to machines



You choose your OS in the 1st then add server address in the 2nd phase and name for the agent in the 3rd. It will generate a code in the 4th phase, you will copy that code and paste it to the terminal then it will automatically start installing Wazuh-agent service, then you will start the service itself. That's it.

Above one is just example.

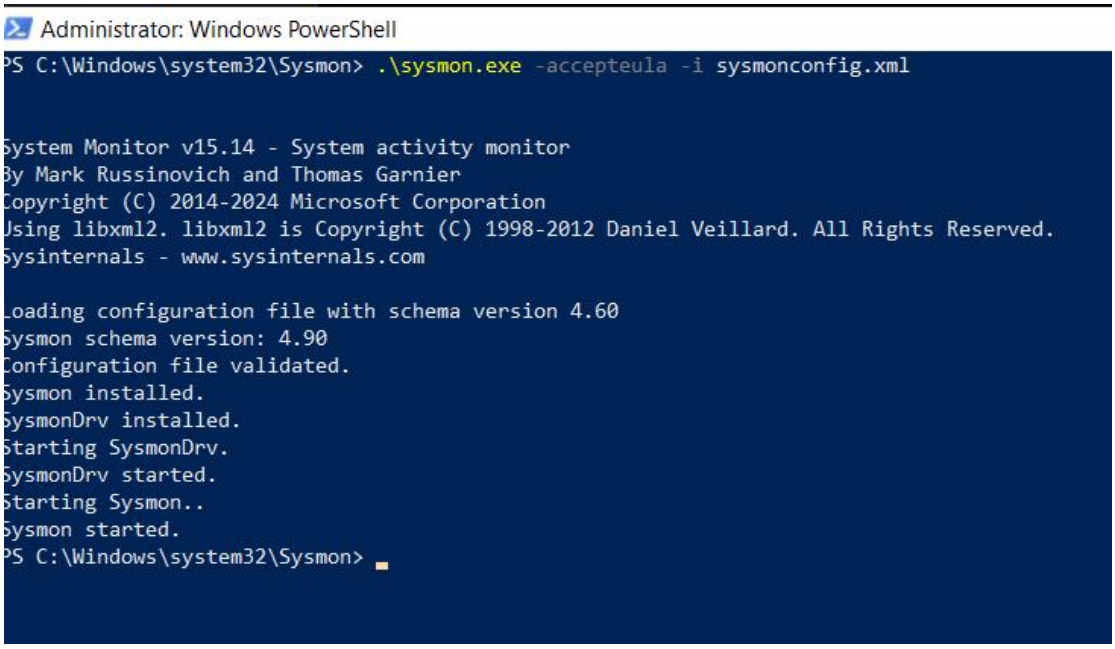
Active Directory attack simulations and alerts in Wazuh

Detection rules

To detect AD attacks, we create rules on the Wazuh server to detect IoCs in Windows security events and system events monitored by Sysmon.

Sysmon integration

Using the configuration file sysmonconfig.xml on the Windows 2019 domain controller and the attacked Windows 10 endpoint, download Sysmon from https://learn.microsoft.com/en-us/sysinternals/downloads/sysmon.



3. Add the following parameters to the agent configuration file located at "C:\Program Files (x86)\ossec-agent\ossec.conf" to set up the Wazuh agents to collect Sysmon events:

```
<ossec_config>
  <localfile>
    <location>Microsoft-Windows-Sysmon/Operational</location>
    <log_format>eventchannel</log_format>
  </localfile>
</ossec_config>
```

4. Apply the changes by restarting the agents using this PowerShell command: Restart-Service -Name wazuh

Wazuh server configuration

To create alerts on the Wazuh dashboard anytime an attacker executes any of the Active Directory attacks, add the following rules to the Wazuh server's /var/ossec/etc/rules/local_rules.xml file:

```
<group name="security_event, windows,">

  <!-- This rule detects DCSync attacks using windows security event on the domain controller -->
  <rule id="110001" level="12">
    <if_sid>60103</if_sid>
    <field name="win.system.eventID">^4662$</field>
    <field name="win.eventdata.properties" type="pcre2">{1131f6aa-9c07-11d1-f79f-
00c04fc2dcd2}|{19195a5b-6da0-11d0-afd3-00c04fd930c9}</field>
    <options>no_full_log</options>
    <description>Directory Service Access. Possible DCSync attack</description>
  </rule>

 <!-- This rule ignores Directory Service Access originating from machine accounts containing $ -->
 <rule id="110009" level="0">
    <if_sid>60103</if_sid>
    <field name="win.system.eventID">^4662$</field>
    <field name="win.eventdata.properties" type="pcre2">{1131f6aa-9c07-11d1-f79f-
00c04fc2dcd2}|{19195a5b-6da0-11d0-afd3-00c04fd930c9}</field>
    <field name="win.eventdata.SubjectUserName" type="pcre2">\$$</field>
    <options>no_full_log</options>
    <description>Ignore all Directory Service Access that is originated from a machine account
containing $</description>
  </rule>

  <!-- This rule detects Keberoasting attacks using windows security event on the domain controller -
->
  <rule id="110002" level="12">
    <if_sid>60103</if_sid>
    <field name="win.system.eventID">^4769$</field>
    <field name="win.eventdata.TicketOptions" type="pcre2">0x40810000</field>
    <field name="win.eventdata.TicketEncryptionType" type="pcre2">0x17</field>
    <options>no_full_log</options>
    <description>Possible Keberoasting attack</description>
  </rule>

  <!-- This rule detects Golden Ticket attacks using windows security events on the domain controller
-->
  <rule id="110003" level="12">
    <if_sid>60103</if_sid>
```

```
    <field name="win.system.eventID">^4624$</field>
    <field name="win.eventdata.LogonGuid" type="pcre2">{00000000-0000-0000-0000-
000000000000}</field>
    <field name="win.eventdata.logonType" type="pcre2">3</field>
    <options>no_full_log</options>
      <description>Possible Golden Ticket attack</description>
<group name="security_event, windows,">
  <!-- This rule detects when PsExec is launched remotely to perform lateral movement within the
domain. The rule uses Sysmon events collected from the domain controller. -->
  <rule id="110004" level="12">
    <if_sid>61600</if_sid>
    <field name="win.system.eventID" type="pcre2">17|18</field>
    <field name="win.eventdata.PipeName" type="pcre2">\\PSEXESVC</field>
    <options>no_full_log</options>
    <description>PsExec service launched for possible lateral movement within the
domain</description>
  </rule>
  <!-- This rule detects NTDS.dit file extraction using a sysmon event captured on the domain
controller -->
  <rule id="110006" level="12">
    <if_group>sysmon_event1</if_group>
    <field name="win.eventdata.commandLine" type="pcre2">NTDSUTIL</field>
    <description>Possible NTDS.dit file extraction using ntdsutil.exe</description>
  </rule>
  <!-- This rule detects Pass-the-ash (PtH) attacks using windows security event 4624 on the
compromised endpoint -->
  <rule id="110007" level="12">
    <if_sid>60103</if_sid>
    <field name="win.system.eventID">^4624$</field>
    <field name="win.eventdata.LogonProcessName" type="pcre2">seclogo</field>
    <field name="win.eventdata.LogonType" type="pcre2">9</field>
    <field name="win.eventdata.AuthenticationPackageName" type="pcre2">Negotiate</field>
    <field name="win.eventdata.LogonGuid" type="pcre2">{00000000-0000-0000-0000-
000000000000}</field>
    <options>no_full_log</options>
    <description>Possible Pass the hash attack</description>
  </rule>

  <!-- This rule detects credential dumping when the command sekurlsa::logonpasswords is run on
mimikatz -->
  <rule id="110008" level="12">
    <if_sid>61612</if_sid>
    <field name="win.eventdata.TargetImage" type="pcre2">(?i)\\\\system32\\\\lsass.exe</field>
    <field name="win.eventdata.GrantedAccess" type="pcre2">(?i)0x1010</field>
    <description>Possible credential dumping using mimikatz</description>
  </rule> </group>
```

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

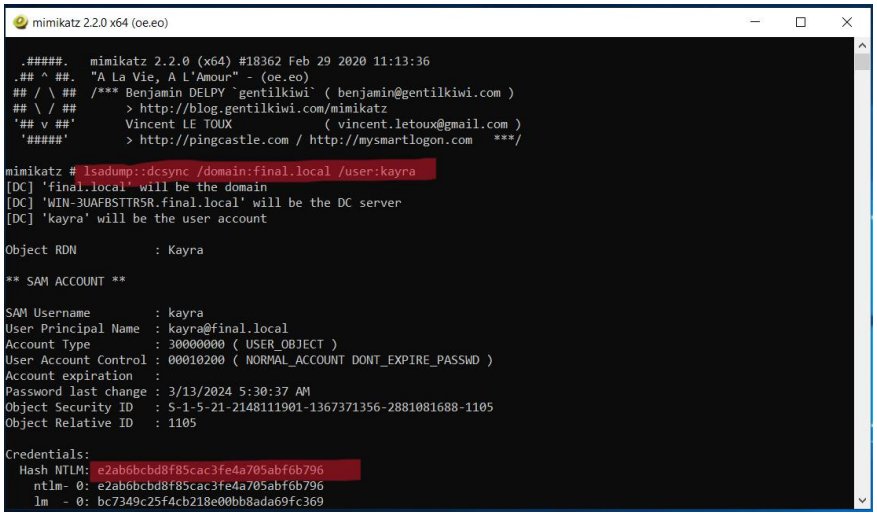
service wazuh-manager restart

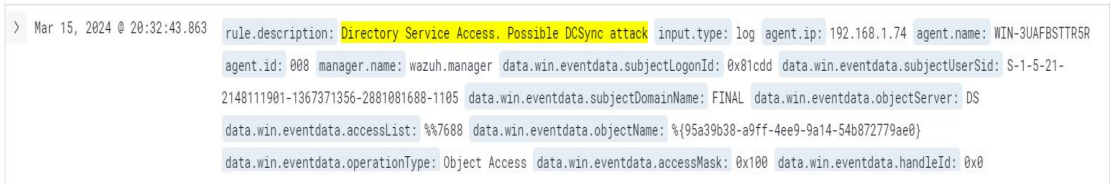
In this section we show how to simulate some common active directory attacks.

DCSync attack simulation

Threat actors use the credential dumping technique known as DCSync to steal the credentials of domain users. The remote protocol domain controllers for the Directory Replication Service (DRS), which are used for replication and synchronization, are abused in this attack. A threat actor needs access to a domain user account with the permissions "Replicating Directory Changes" and "Replicating Directory Changes All" in order to carry out this attack successfully. The steps to carry out a DCSync attack are as follows:

1. Run mimikatz as administrator and run the following command in the mimikatz console to replicate KRBTGT credentials from the Active Directory.

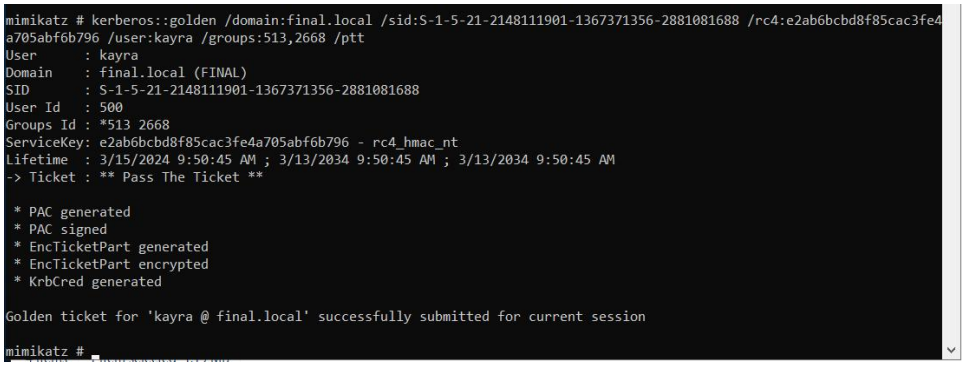




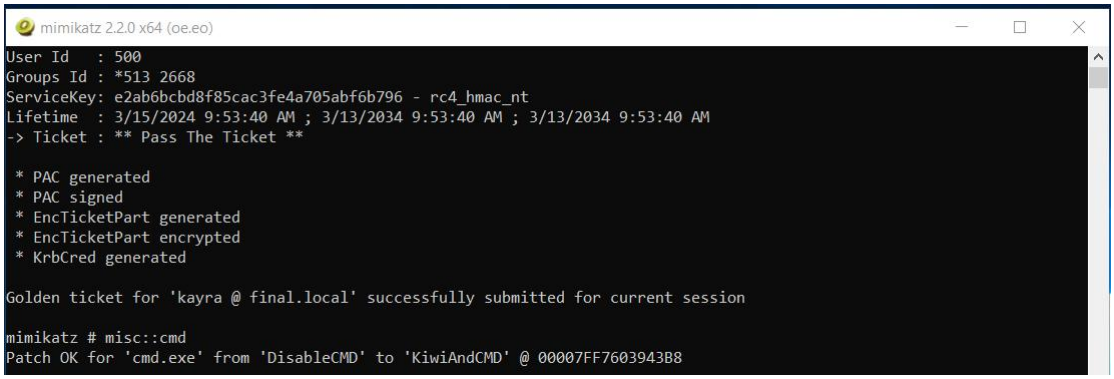
Golden ticket attack simulation

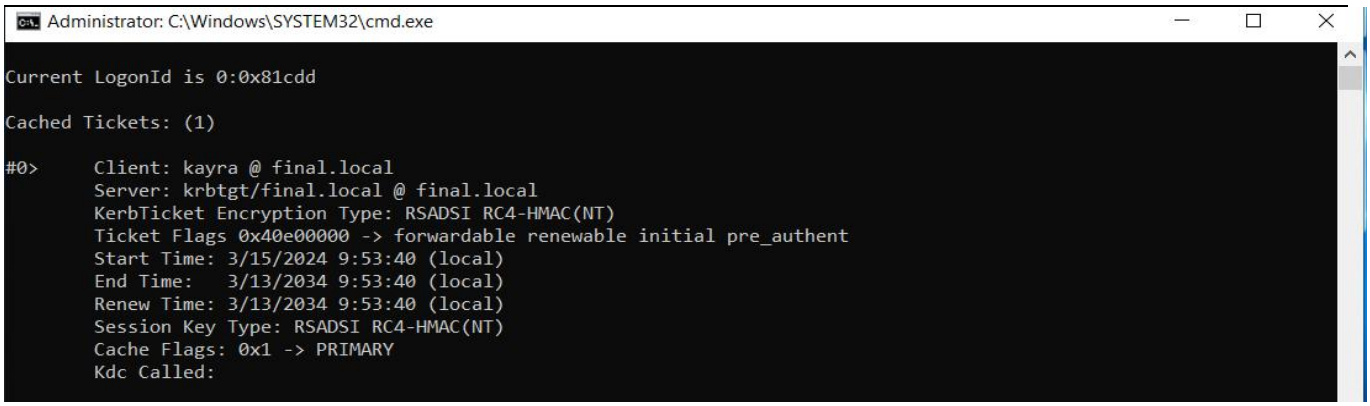
A Golden Ticket attack abuses the Kerberos protocol, which depends on the use of shared secrets to encrypt and sign messages. One of these secrets is known only to the Key Distribution Center (KDC): the password hash for the KRBTGT user, which is used to issue the Kerberos tickets required to access IT systems and data.

1. Run mimikatz as administrator and run the following command to forge Kerberos tickets using the NTLM hash of the KRBTGT account obtained during the DCSync attack.



2.Run the misc::cmd to open a command prompt session authenticated with the forged Kerberos ticket. With klist command to verify the forged ticket is currently loaded into memory for the current session.

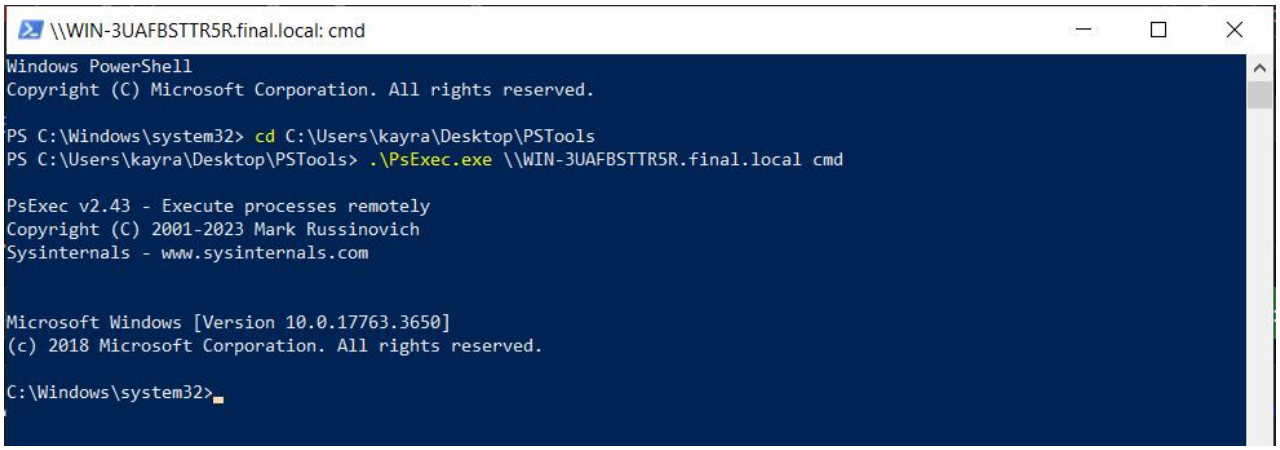




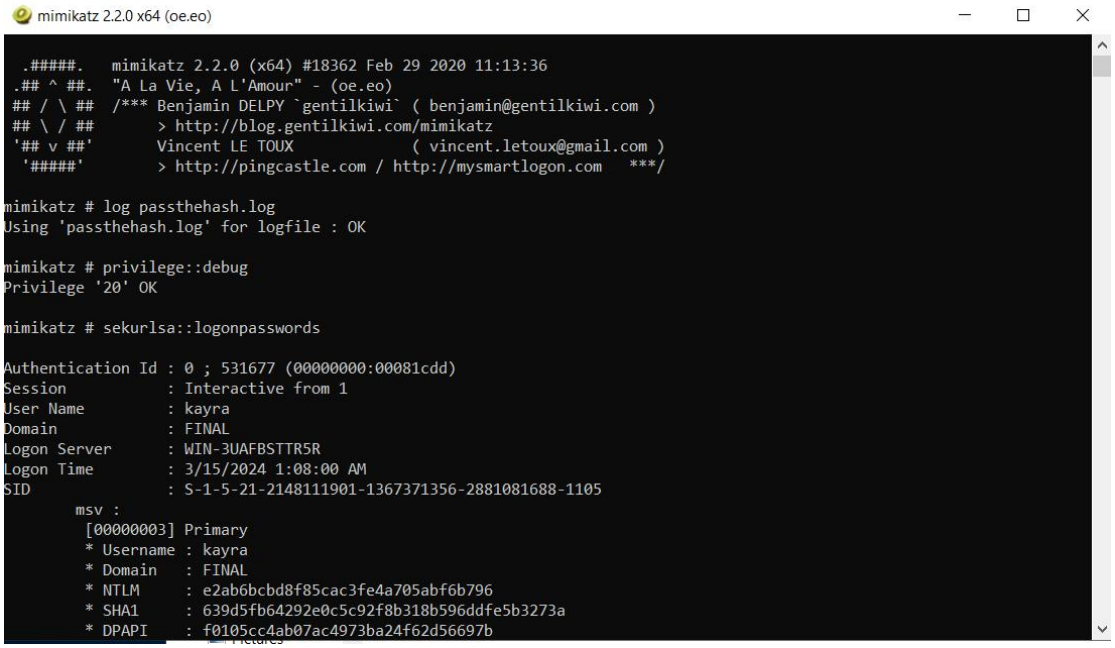
Pass the hash attack simulation

Threat actors use the pass-the-hash technique to take credentials and move laterally. Instead of using the account's plaintext password for authentication, this attack uses the NTLM authentication protocol to authenticate a user using a password hash that has been captured.

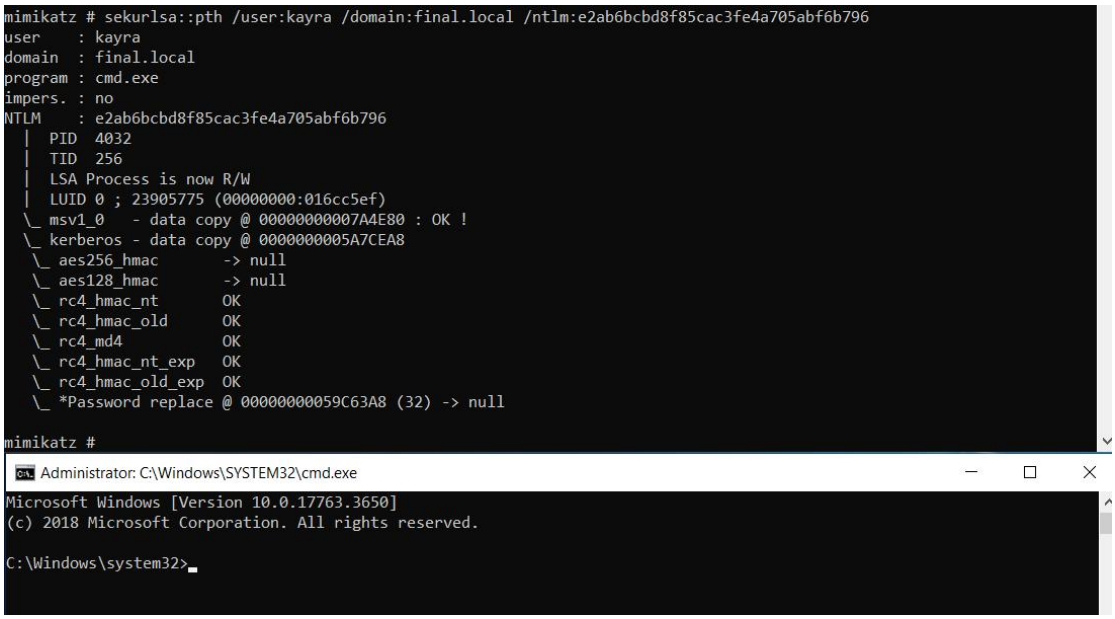
1. First of all, we need to install PsTools and then run PowerShell as administrator and change the current directory to the PsTools directory. Then run the .\PsExec.exe with Domain name and cmd command to connect to the domain controller and execute commands remotely.



2. Open mimikatz as an administrator, then run the log passthehash.log and privilege::debug commands. Then run sekurlsa::logonpasswords to extract password hashes from the LSASS.exe process memory, which stores the hashes for users with active sessions to the computer.



3. Then we need to authenticate as the compromised user.



4. Detection Result.

	Time	rule.description	rule.id
>	Mar 17, 2024 @ 20:20:30.549	Possible Pass the hash attack	110007
>	Mar 17, 2024 @ 20:19:46.159	Lsass process was accessed by C:\\Users\\kayra\\Desktop\\mimikatz-master\\mimikatz-master\\x64\\mimikatz.exe with read permissions, possible credential dump	92900

Ntds.dit password extraction simulation

The ntds.dit file located in C:\Windows\NTDS\ is the database that stores all the data in the Active Directory on every domain controller. Attackers can compromise users' credentials by extracting the password hash from the ntds.dit file. This attack can be achieved by using several techniques to copy the ntds.dit file from the DC to a local system to crack the password offline.

We need to access to the domain controller file system to extract ntds.dit file, hence this attack scenario will leverage the access obtained during pass the hash attack.

```
PS C:\Users\kayra\Desktop\PSTools> .\PsExec.exe \\WIN-3UAFBSTTR5R.final.local cmd

PsExec v2.43 - Execute processes remotely
Copyright (C) 2001-2023 Mark Russinovich
Sysinternals - www.sysinternals.com


Microsoft Windows [Version 10.0.17763.3650]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>NTDSUTIL "Activate Instance NTDS" "IFM" "Create Full C:\Files" "q" "q"
NTDSUTIL: Activate Instance NTDS
Active instance set to "NTDS".
NTDSUTIL: IFM
ifm: Create Full C:\Files
Creating snapshot...
Snapshot set {95bac5a9-f66a-4e68-b24e-cde247e0e274} generated successfully.
Snapshot {b67d67a6-4154-4091-b8a9-d4f6ec2af008} mounted as C:\$SNAP_202403170715_VOLUMEC$\
Snapshot {b67d67a6-4154-4091-b8a9-d4f6ec2af008} is already mounted.
Initiating DEFRAGMENTATION mode...
     Source Database: C:\$SNAP_202403170715_VOLUMEC$\Windows\NTDS\ntds.dit
     Target Database: C:\Files\Active Directory\ntds.dit

                Defragmentation  Status (omplete)

          0    10   20   30   40   50   60   70   80   90  100
          |----|----|----|----|----|----|----|----|----|----|
          ..................................................

Copying registry files...
Copying C:\Files\registry\SYSTEM
Copying C:\Files\registry\SECURITY
Snapshot {b67d67a6-4154-4091-b8a9-d4f6ec2af008} unmounted.
IFM media created successfully in C:\Files
ifm: q
NTDSUTIL: q

C:\Windows\system32>
```

Detection Result:

Mar 17, 2024 @ 18:15:27.432 Possible NTDS.dit file extraction using ntdsutil.exe

Expanded document

View surrounding documents View single document

Table JSON

t _index	wazuh-alerts-4.x-2024.03.17
t agent.id	008
t agent.ip	192.168.140.153
t agent.name	WIN-3UAFBSTTR5R
t data.win.eventdata.commandLine	NTDSUTIL \"Activate Instance NTDS\" \"IFM\" \"Create Full C:\\Files\" \"q\" \"q\"
t data.win.eventdata.company	Microsoft Corporation
t data.win.eventdata.currentDirectory	C:\\Windows\\system32\\
t data.win.eventdata.description	NT5DS
t data.win.eventdata.fileVersion	10.0.17763.2452 (WinBuild.160101.0800)
t data.win.eventdata.hashes	SHA1=7EA09B7B6CFE6EDA43C227976E7B7EE4B9FBFD59,MD5=A539B5B605582EA465D7306E0076D028,SHA256=08B052934E91C0CF0E4328FAD74BABE3A5CCD95D94C5124C205D4D1957F464A8,IMPHASH=21A2CEC3EDDE94A302888703A03E6B8D
t data.win.eventdata.image	C:\\Windows\\System32\\ntdsutil.exe
t data.win.eventdata.integrityLevel	High
t data.win.eventdata.logonGuid	{7da25987-01e0-65f4-dd1c-080000000000}
t data.win.eventdata.logonId	0x81cdd
t data.win.eventdata.originalFileName	ntdsutil.exe
t data.win.eventdata.parentCommandLine	\"cmd\"

Windows monitoring and detecting malicious actions with Wazuh

Detecting malware persistence technique

Wazuh monitors the startup folder automatically without requiring any user action. By default, the Wazuh configuration file at C:\Program Files (x86)\ossec-agent\ossec.conf uses the following setting to monitor the startup folder:

```
<!-- Detecting malware persistence technique -->
<directories realtime="yes">%PROGRAMDATA%\Microsoft\Windows\Start Menu\Programs\Startup</directories>
```

Use PowerShell to download an EICAR test file to the C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup directory. Delete the file from the Windows endpoint:

```
PS C:\Windows\system32> cd "C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup"
PS C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup> Invoke-WebRequest -Uri https://secure.eicar.org/eicar.com.txt -OutFile eicar.txt
PS C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup>
```

Navigate to Modules > Integrity monitoring on the Wazuh dashboard to view the alert generated when the FIM module detects changes in the Windows startup folder:

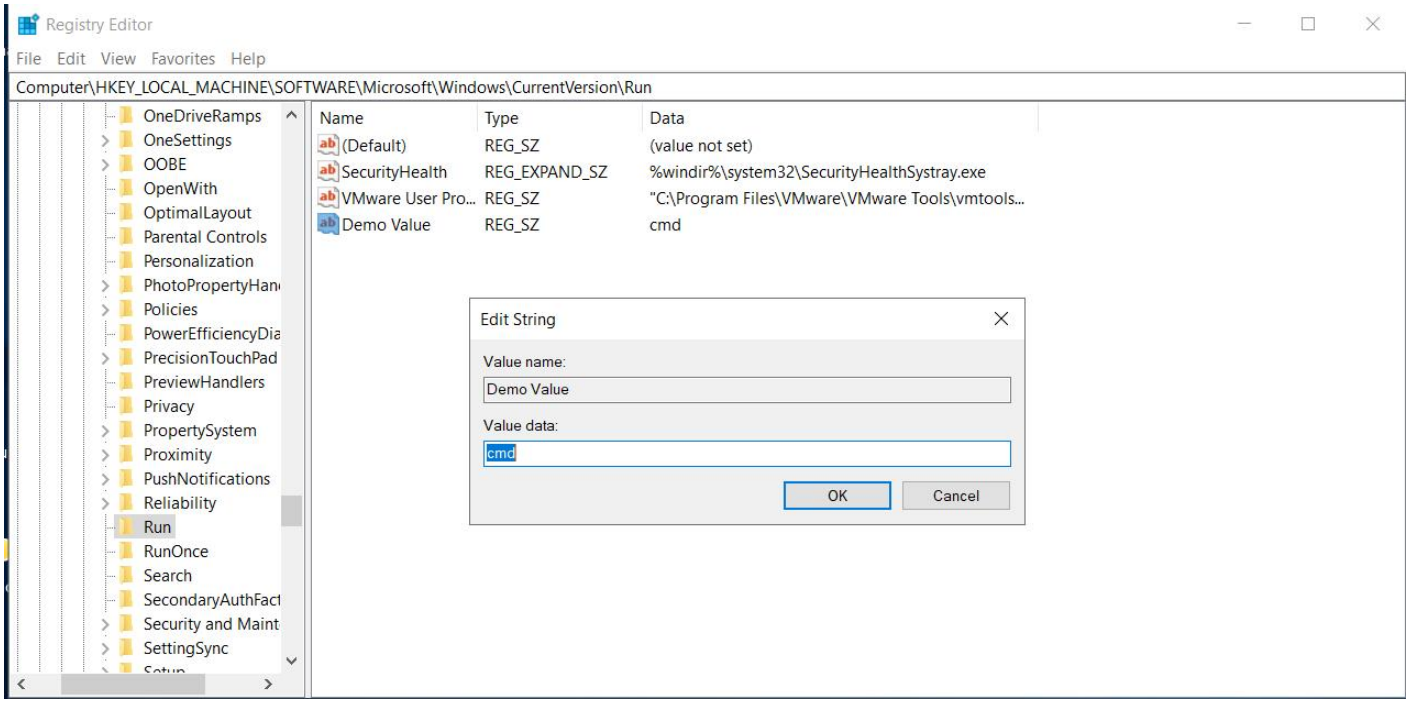
	Time	syscheck.path	syscheck.event	rule.description	rule.level	rule.id
›	Mar 18, 2024 @ 16:19:18.719	c:\programdata\microsoft\windows\start menu\programs\startup\eicar.txt	deleted	File deleted.	7	553
›	Mar 18, 2024 @ 16:16:30.927	c:\programdata\microsoft\windows\start menu\programs\startup\eicar.txt	modified	Integrity checksum changed.	7	550
›	Mar 18, 2024 @ 16:16:30.629	c:\programdata\microsoft\windows\start menu\programs\startup\eicar.txt	added	File added to the system.	5	554

Windows Registry monitoring

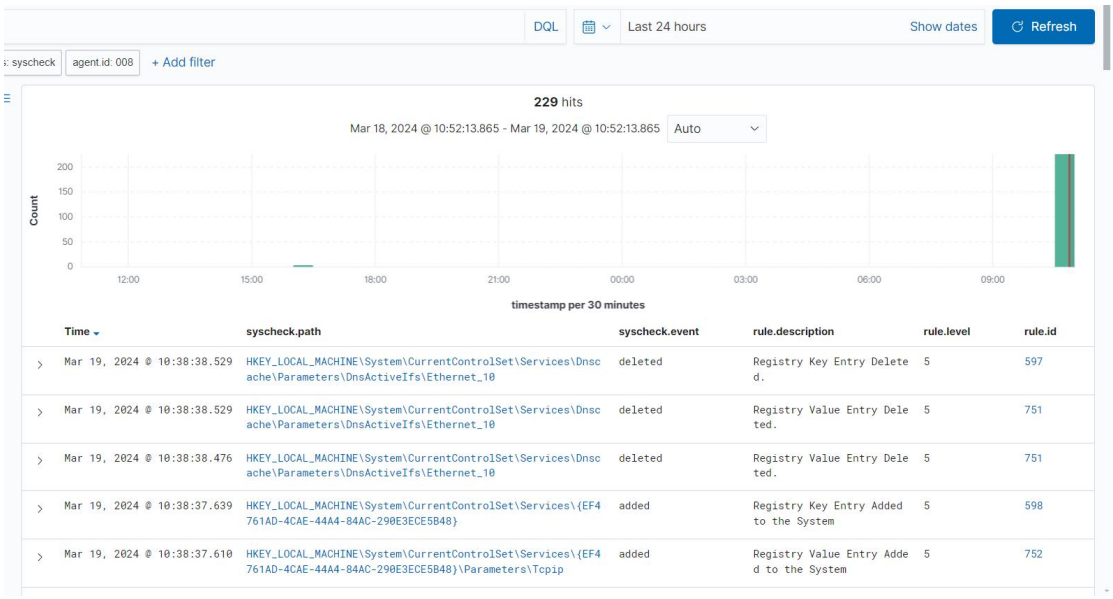
To configure the FIM module, it's necessary to specify the registry keys that FIM must monitor for creation, modification, and deletion. You can do this similarly to how you list directories and files, but using the label <windows_registry> instead. Also, defaultly you can see configurations in ossec.agent.

```
<windows_registry arch="both">HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run</windows_registry>
<windows_registry arch="both">HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunOnce</windows_registry>
```

Then we need open Registry Editor and add Demo value in HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run key:



Events:



Detecting and removing malware using VirusTotal integration

First of all, we searched for the <syscheck> block in the Wazuh agent C:\Program Files (x86)\ossec-agent\ossec.conf file. Make sure that <disabled> is set to no. Include an entry in the <syscheck> block to set up monitoring for a specific directory in almost real-time. In this scenario, I will configure Wazuh to monitor the C:\Users\<USER_NAME>\Downloads folder.

```
<!-- File integrity monitoring -->
<syscheck>

  <disabled>no</disabled>

  <!--Detecting and removing malware using VirusTotal integration--!>
  <directories realtime="yes">C:\Users\kayra\Downloads</directories>
```

Now we downloaded python and installed pyinstaller then, open file name as remove-theart.py with notepad and added script from https://documentation.wazuh.com/current/proof-of-concept-guide/detect-remove-malware-virustotal.html#windows-endpoint
We converted the active response Python script remove-threat.py to a Windows executable application and just restart wazuh agent. We added the following configuration to the /var/ossec/etc/ossec.conf file on the Wazuh server to enable the VirusTotal integration.

```
<integration>
  <name>virustotal</name>
  <api_key>ed7d543e1d9cca7c2a174437b21d2e74f6eef1f530fae1421a2ab86c9094c6f2</api_key>
  <group>syscheck</group>
  <alert_format>json</alert_format>
</integration>

<command>
  <name>remove-threat</name>
  <executable>remove-threat.exe</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>
```

```
</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)</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>
"/var/ossec/etc/rules/local_rules.xml" 111L, 4902C
```

Now we did attack emulation with download Eicar test file in C:\Users\kayra\Downloads:

Invoke-WebRequest -Uri https://secure.eicar.org/eicar.com.txt -OutFile eicar.txt

›	Mar 19, 2024 @ 12:52:56.091	WIN-3UAFBSTTR5R	File deleted.	7	553
›	Mar 19, 2024 @ 12:52:50.185	WIN-3UAFBSTTR5R	Windows logon success.	3	60106
›	Mar 19, 2024 @ 12:52:50.185	WIN-3UAFBSTTR5R	Windows User Logoff.	3	60137
›	Mar 19, 2024 @ 12:52:20.982	wazuh.manager	VirusTotal: Error: Public API request rate limit reached	3	87101
›	Mar 19, 2024 @ 12:52:17.411	WIN-3UAFBSTTR5R	VirusTotal: Alert - c:\users\kayra\downloads\eicar.txt - No positives found	3	87104
›	Mar 19, 2024 @ 12:52:13.581	WIN-3UAFBSTTR5R	VirusTotal: Alert - c:\users\kayra\downloads\eicar.txt - 67 engines detected this file	12	87105
›	Mar 19, 2024 @ 12:52:10.179	WIN-3UAFBSTTR5R	VirusTotal: Alert - c:\users\kayra\downloads\eicar.txt - 67 engines detected this file	12	87105
›	Mar 19, 2024 @ 12:52:06.587	WIN-3UAFBSTTR5R	VirusTotal: Alert - c:\users\kayra\downloads\eicar.txt - No positives found	3	87104
›	Mar 19, 2024 @ 12:52:05.673	WIN-3UAFBSTTR5R	Integrity checksum changed.	7	550
›	Mar 19, 2024 @ 12:52:05.504	WIN-3UAFBSTTR5R	File added to the system.	5	554

Detecting malware using file hashes in a CDB list

We created cdb list in malware-hashes file:

```
root@wazuh:/var/ossec/etc/lists# cat malware-hashes
e0ec2cd43f71c80d42cd7b0f17802c73:mirai
55142f1d393c5ba7405239f232a6c059:Xbash
```

Add a reference to the CDB list in the Wazuh manager configuration file /var/ossec/etc/ossec.conf by specifying the path to the list within the <ruleset> block:

```
  <ruleset>
    <!-- Default ruleset -->
    <decoder_dir>ruleset/decoders</decoder_dir>
    <rule_dir>ruleset/rules</rule_dir>
    <rule_exclude>0215-policy_rules.xml</rule_exclude>
    <list>etc/lists/audit-keys</list>
    <list>etc/lists/amazon/aws-eventnames</list>
    <list>etc/lists/security-eventchannel</list>
    <list>etc/lists/malware-hashes</list>
```

We created new rule in /var/ossec/etc/rules/local_rules.xml:

```
<group name="malware,">
    <rule id="110002" level="13">
            <if_sid>554, 550</if_sid>
            <list field="md5" lookup="match_key">etc/lists/malware-hashes</list>
            <description>File with known malware hash detected: $(file)</description>
            <mitre>
                    <id>T1204.002</id>
            </mitre>
    </rule>
</group>
"/var/ossec/etc/rules/local_rules.xml" 122L, 5218C
```

We configured directory monitoring by adding the <directories> block specifying the folders to be monitored in the agent configuration file:

```
  <!-- Detecting malware using file hashes in a CDB list --!>
  <directories check_all="yes" realtime="yes" whodata="yes">C:\Users\kayra\AppData\Local\Temp</directories>
```

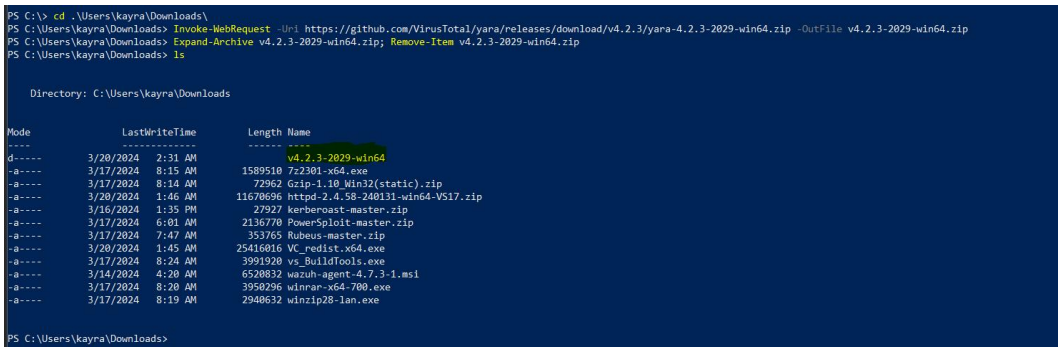
Testing:

```
PS C:\Users\kayra\AppData\Local\Temp> Invoke-WebRequest -Uri https://wazuh-demo.s3-us-west-1.amazonaws.com/mirai -OutFile mirai
PS C:\Users\kayra\AppData\Local\Temp> Invoke-WebRequest -Uri https://wazuh-demo.s3-us-west-1.amazonaws.com/xbash -OutFile Xbash
PS C:\Users\kayra\AppData\Local\Temp>
```

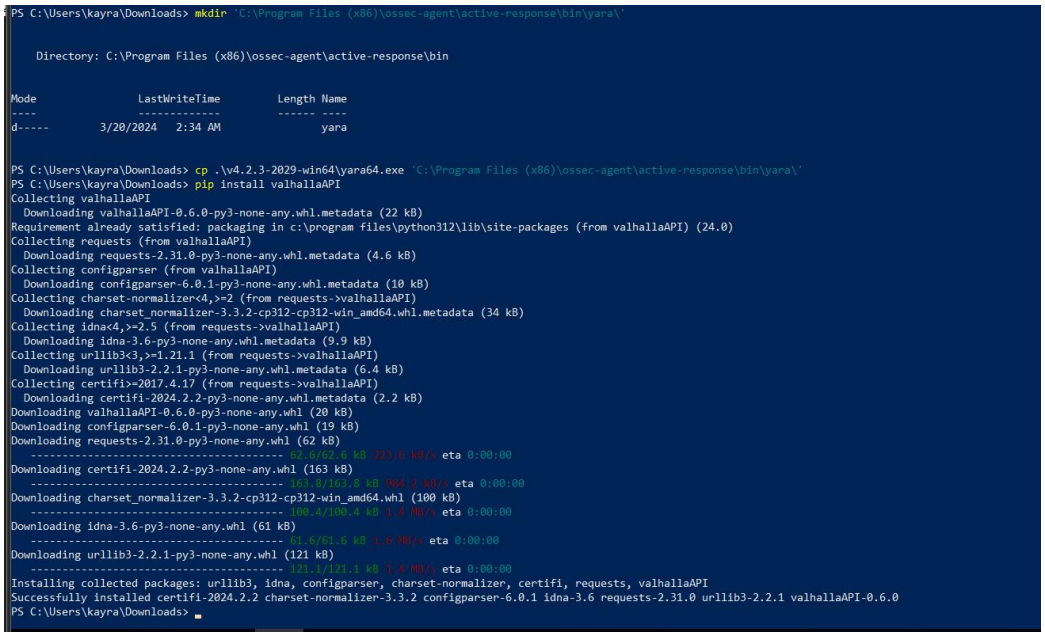
›	Mar 19, 2024 @ 23:58:27.950	WIN-3UAFBSTTR5R	VirusTotal: Alert - c:\users\kayra\appdata\local\temp\xbash - 39 engines detected this file	12	87105
›	Mar 19, 2024 @ 23:58:24.866	WIN-3UAFBSTTR5R	VirusTotal: Alert - c:\users\kayra\appdata\local\temp\mirai - 46 engines detected this file	12	87105

Detecting malware using Yara integration

Firstly, we downloaded and extracted Yara:



We created a directory called C:\Program Files (x86)\ossec-agent\active-response\bin\yara\ and copy the YARA executable into it and installed valhallaAPI:



We created python script for download yara rules and downloaded the rules and place them in the C:\Program Files (x86)\ossec-agent\active-response\bin\yara\rules\ directory

```
PS C:\Users\kayra\Downloads> python.exe download_yara_rules.py
PS C:\Users\kayra\Downloads> mkdir 'C:\Program Files (x86)\ossec-agent\active-response\bin\yara\rules\'


    Directory: C:\Program Files (x86)\ossec-agent\active-response\bin\yara


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        3/20/2024   2:39 AM                rules


PS C:\Users\kayra\Downloads> cp yara_rules.yar 'C:\Program Files (x86)\ossec-agent\active-response\bin\yara\rules\'
PS C:\Users\kayra\Downloads> ls


    Directory: C:\Users\kayra\Downloads


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        3/20/2024   2:31 AM                v4.2.3-2029-win64
-a----        3/17/2024   8:15 AM        1589510 7z2301-x64.exe
-a----        3/20/2024   2:39 AM            239 download_yara_rules.py
-a----        3/17/2024   8:14 AM          72962 Gzip-1.10_Win32(static).zip
-a----        3/20/2024   1:46 AM       11670696 httpd-2.4.58-240131-win64-VS17.zip
-a----        3/16/2024   1:35 PM          27927 kerberoast-master.zip
-a----        3/17/2024   6:01 AM        2136770 PowerSploit-master.zip
-a----        3/17/2024   7:47 AM         353765 Rubeus-master.zip
-a----        3/20/2024   1:45 AM       25416016 VC_redist.x64.exe
-a----        3/17/2024   8:24 AM        3991920 vs_BuildTools.exe
-a----        3/14/2024   4:20 AM        6520832 wazuh-agent-4.7.3-1.msi
-a----        3/17/2024   8:20 AM        3950296 winrar-x64-700.exe
-a----        3/17/2024   8:19 AM        2940632 winzip28-lan.exe
-a----        3/20/2024   2:39 AM        2989699 yara_rules.yar
```

We created the yara.bat script in the C:\Program Files (x86)\ossec-agent\active-response\bin\ directory. This is necessary for the Wazuh-YARA active response scans:

yara - Notepad

File Edit Format View Help

```
@echo off

setlocal enableDelayedExpansion

reg Query "HKLM\Hardware\Description\System\CentralProcessor\0" | find /i "x86" > NUL && SET OS=32BIT || SET OS=64BIT


if %OS%==32BIT (
    SET log_file_path="%programfiles%\ossec-agent\active-response\active-responses.log"
)

if %OS%==64BIT (
    SET log_file_path="%programfiles(x86)%\ossec-agent\active-response\active-responses.log"
)

set input=
for /f "delims=" %%a in ('PowerShell -command "$logInput = Read-Host; Write-Output $logInput"') do (
    set input=%%a
)


set json_file_path="C:\Program Files (x86)\ossec-agent\active-response\stdin.txt"
set syscheck_file_path=
echo %input% > %json_file_path%

for /F "tokens=* USEBACKQ" %%F in (`Powershell -Nop -C "(Get-Content 'C:\Program Files (x86)\ossec-agent\active-response\stdin.txt'|ConvertFrom-Json).parameters.alert.sy
set syscheck_file_path=%%F
)

del /f %json_file_path%
set yara_exe_path="C:\Program Files (x86)\ossec-agent\active-response\bin\yara\yara64.exe"
set yara_rules_path="C:\Program Files (x86)\ossec-agent\active-response\bin\yara\rules\yara_rules.yar"
echo %syscheck_file_path% >> %log_file_path%
for /f "delims=" %%a in ('powershell -command "& \"%yara_exe_path%\" \"%yara_rules_path%\" \"%syscheck_file_path%\""') do (
    echo wazuh-yara: INFO - Scan result: %%a >> %log_file_path%
```

```
PS C:\Program Files (x86)\ossec-agent\active-response\bin> notepad yara.bat
PS C:\Program Files (x86)\ossec-agent\active-response\bin> ls


    Directory: C:\Program Files (x86)\ossec-agent\active-response\bin


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         3/20/2024   2:34 AM                yara
-a----         2/29/2024   1:12 PM         191608 netsh.exe
-a----         3/19/2024   1:25 PM        7296642 remove-malware.exe
-a----         2/29/2024   1:12 PM         186488 restart-wazuh.exe
-a----         2/29/2024   1:12 PM         188536 route-null.exe
-a----         3/20/2024   2:45 AM           1375 yara.bat
```

We added the C:\Users\kayra\Downloads directory for monitoring within the <syscheck> block in the Wazuh agent configuration file C:\Program Files (x86)\ossec-agent\ossec.conf

```
<!-- Detecting and removing malware using VirusTotal/Yara integration --!>
<directories check_all="yes" realtime="yes" report_changes=="yes" whodata="yes">C:\Users\kayra\Downloads</directories>
```

We added the below decoders to the Wazuh server /var/ossec/etc/decoders/local_decoder.xml file. This allows extracting the information from YARA scan results:

```
<decoder name="yara_decoder">
        <prematch>wazuh-yara:</prematch>
</decoder>

<decoder name="yara_decoder1">
        <parent>yara_decoder</parent>
        <regex>wazuh-yara: (\S+) - Scan result: (\S+) (\S+)</regex>
        <order>log_type, yara_rule, yara_scanned_file</order>
</decoder>
```

Then we added the below rules to the Wazuh server /var/ossec/etc/rules/local_rules.xml file. The rules detect FIM events in the monitored directory. They also alert when malware is found by the YARA integration:

```
root@wazuh: /

<group name="syscheck,">
  <rule id="100303" level="7">
    <if_sid>550</if_sid>
    <field name="file">C:\\Users\\kayra\\Downloads</field>
    <description>File modified in C:\Users\kayra\Downloads directory.</description>
  </rule>

  <rule id="100304" level="7">
    <if_sid>554</if_sid>
    <field name="file">C:\\Users\\kayra\\Downloads</field>
    <description>File added to C:\Users\kayra\Downloads  directory.</description>
  </rule>
</group>

<group name="yara,">
  <rule id="108000" level="0">
    <decoded_as>yara_decoder</decoded_as>
    <description>Yara grouping rule</description>
  </rule>

  <rule id="108001" level="12">
    <if_sid>108000</if_sid>
    <match>wazuh-yara: INFO - Scan result: </match>
    <description>File "$(yara_scanned_file)" is a positive match. Yara rule: $(yara_rule)</description>
  </rule>

</group>
```

Added the below active-response configuration to the Wazuh server /var/ossec/etc/ossec.conf file:

```
root@wazuh: /
    <executable>netsh.exe</executable>
    <timeout_allowed>yes</timeout_allowed>
  </command>

  <command>
    <name>yara_windows</name>
    <executable>yara.bat</executable>
    <timeout_allowed>no</timeout_allowed>
  </command>

  <active-response>
    <command>yara_windows</command>
    <location>local</location>
    <rules_id>100303,100304</rules_id>
  </active-response>
```

Visual:

>	Mar 20, 2024 @ 22:24:34.003	WIN-3UAFBSTTR5R	File "c:\users\kayra\downloads\eicar.com" is a positive match. Yara rule: SUSP_Just_EICAR_RID2C24	12	108001
>	Mar 20, 2024 @ 22:24:32.580	WIN-3UAFBSTTR5R	C:\\Windows\\SysWOW64\\WindowsPowerShell\\v1.0\\powershell.exe created a new scripting file under Windows Temp or User data folder	9	92201
>	Mar 20, 2024 @ 22:24:3	WIN-3UAFBSTTR5R	C:\\Windows\\SysWOW64\\WindowsPowerShell\\v1.0\\powershell.exe created a new scripting file under Windows Temp or User data folder	9	92201
>	Mar 20, 2024 @ 22:24:30.744	WIN-3UAFBSTTR5R	VirusTotal: Alert - c:\users\kayra\downloads\eicar.com - 64 engines detected this file	12	87105

Log data collection

In this use case, Wazuh will detect when an application is installed on a Windows endpoint. In this use case, an application called Dr. Memory will be installed. First of all, we installed Dr.Memory as normal. By default, the Wazuh agent monitors the installation of applications using the configuration in the Wazuh agent configuration file C:\Program Files (x86)\ossec-agent\ossec.conf.

Wazuh has a built-in rule 60612 to detect when an application is installed on a Windows endpoint which can be viewed in the /var/ossec/ruleset/rules/0585-win-application_rules.xml rule file on the Wazuh server.

```
<rule id="60612" level="3">
  <if_sid>60609</if_sid>
  <field name="win.system.eventID">^11707$|^1033$</field>
  <options>no_full_log</options>
  <description>Application installed $(win.eventdata.data).</description>
</rule>
```

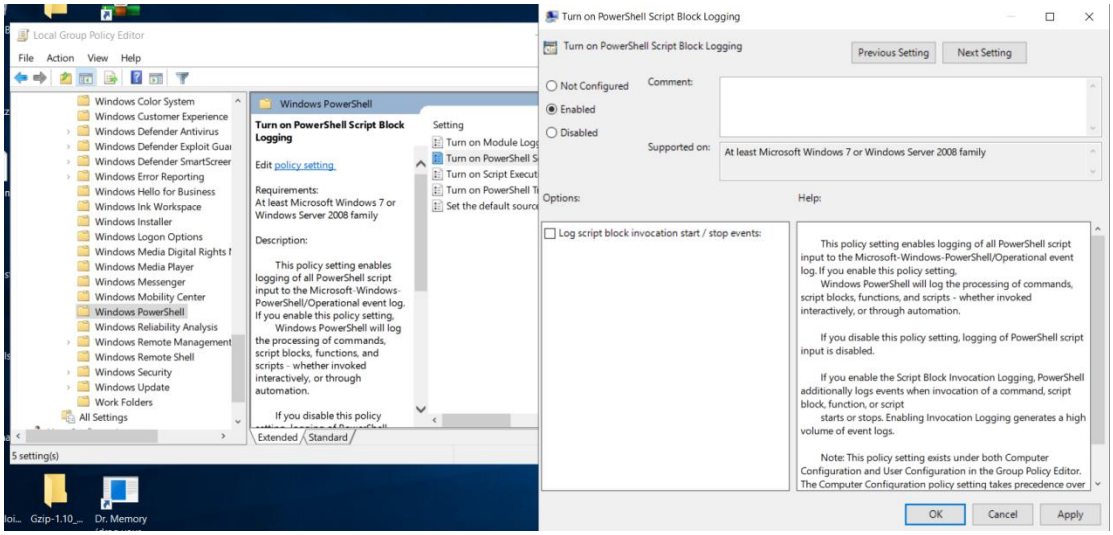
Visualize:

	Time	agent.name	rule.description	rule.level	rule.id
›	Mar 28, 2024 @ 16:50:45.113	WIN-3UAFBSTTR5R	Application installed Dr. Memory, 2.6.0, 1033, 0, Google.	3	60612
›	Mar 28, 2024 @ 16:50:44.981	WIN-3UAFBSTTR5R	Application installed Product: Dr. Memory -- Installation completed successfully..	3	60612

Monitoring PowerShell activity

Wazuh is set up to identify instances where a PowerShell script is executed.

Enter Local Group Policy Editor then navigate to Computer Configuration > Administrative Templates > Windows Components > Windows PowerShell > Turn on PowerShell Script Block Logging.

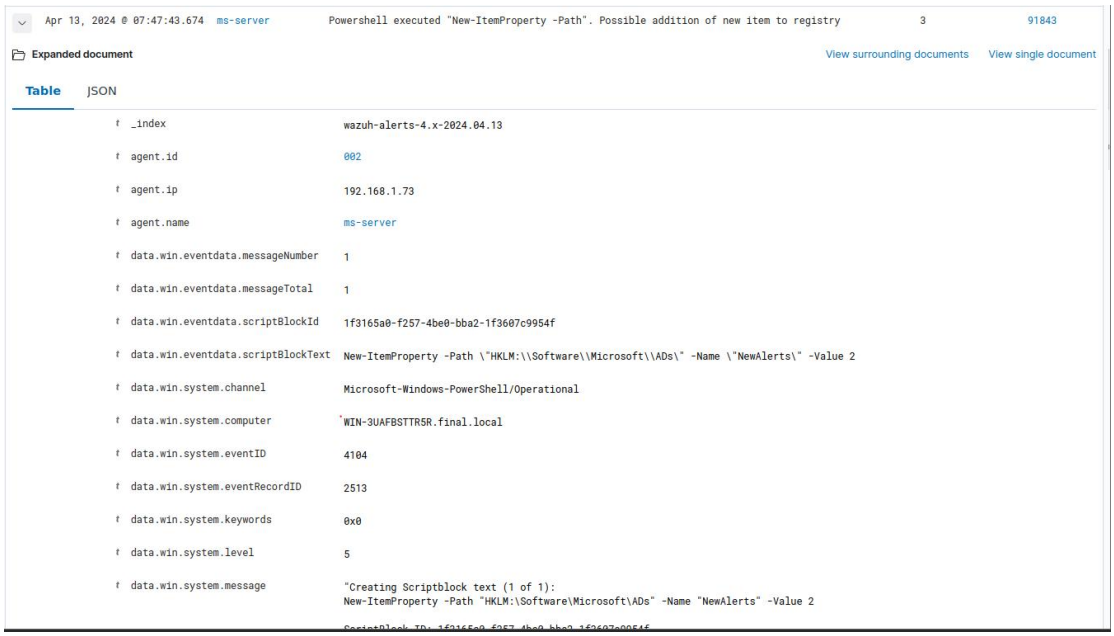


In Wazuh agent configuration file C:\Program Files (x86)\ossec-agent\ossec.conf to monitor PowerShell logs:

```
<localfile>
  <location>Microsoft-Windows-PowerShell/Operational</location>
  <log_format>eventchannel</log_format>
</localfile>
```

Testing and visualize:

We entered command via PowerShell with administrator privileges to add a registry entry NewAlerts to the HKLM\Software\Microsoft\ADs registry key, and set the value to 2:



Monitoring, detecting and response with Wazuh on Ubuntu endpoint

Blocking a known malicious actor

Before everything the step we followed to install the Apache web server and monitor its logs with the Wazuh agent and added the below to /var/ossec/etc/ossec.conf file to configure the Wazuh agent and monitor the Apache access logs:

```
<localfile>
  <log_format>syslog</log_format>
  <location>/var/log/apache2/access.log</location>
</localfile>
```

To added the IP address of Kali VM to a CDB list, and then configure rules and active response; these are the steps we took :

```
root@wazuh:/# wget https://raw.githubusercontent.com/firehol/blocklist-ipsets/master/alienvault_reputation.ipset -O /var/ossec/etc/lists/alienvault_reputation.ipset
--2024-04-16 12:32:40--  https://raw.githubusercontent.com/firehol/blocklist-ipsets/master/alienvault_reputation.ipset
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.110.133, 185.199.109.133, 185.199.111.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.110.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 9495 (9.3K) [text/plain]
Saving to: '/var/ossec/etc/lists/alienvault_reputation.ipset'

/var/ossec/etc/lists/alienvault_reputation.ip 100%[=========================================================================================>]   9.27K  --.-KB/s    in 0.005s

2024-04-16 12:32:41 (1.95 MB/s) - '/var/ossec/etc/lists/alienvault_reputation.ipset' saved [9495/9495]

root@wazuh:/# echo 192.168.140.130 >> /var/ossec/etc/lists/alienvault_reputation.ipset
root@wazuh:/# wget https://wazuh.com/resources/iplist-to-cdblist.py -O /tmp/iplist-to-cdblist.py
--2024-04-16 12:32:59--  https://wazuh.com/resources/iplist-to-cdblist.py
Resolving wazuh.com (wazuh.com)... 108.156.60.104, 108.156.60.80, 108.156.60.30, ...
Connecting to wazuh.com (wazuh.com)|108.156.60.104|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1570 (1.5K) [binary/octet-stream]
Saving to: '/tmp/iplist-to-cdblist.py'

/tmp/iplist-to-cdblist.py                   100%[=========================================================================================>]   1.53K  --.-KB/s    in 0s

2024-04-16 12:33:00 (495 MB/s) - '/tmp/iplist-to-cdblist.py' saved [1570/1570]

root@wazuh:/# /var/ossec/framework/python/bin/python3 /tmp/iplist-to-cdblist.py /var/ossec/etc/lists/alienvault_reputation.ipset /var/ossec/etc/lists/blacklist-alienvault
[/var/ossec/etc/lists/alienvault_reputation.ipset] -> [/var/ossec/etc/lists/blacklist-alienvault]
root@wazuh:/# rm -rf /var/ossec/etc/lists/alienvault_reputation.ipset
root@wazuh:/# rm -rf /tmp/iplist-to-cdblist.py
root@wazuh:/# chown wazuh:wazuh /var/ossec/etc/lists/blacklist-alienvault
root@wazuh:/# cd /var/ossec/etc/lists/
root@wazuh:/var/ossec/etc/lists# ls
amazon  audit-keys  audit-keys.cdb  blacklist-alienvault  security-eventchannel  security-eventchannel.cdb
root@wazuh:/var/ossec/etc/lists#
```

Added a custom rule in /var/ossec/etc/rules/local_rules.xml to trigger a Wazuh active response script:

```
<group name="attack,">
  <rule id="100100" level="10">
    <if_group>web|attack|attacks</if_group>
    <list field="srcip" lookup="address_match_key">etc/lists/blacklist-alienvault</list>
    <description>IP address found in AlienVault reputation database.</description>
  </rule>
</group>
```

```
 <ruleset>
   <!-- Default ruleset -->
   <decoder_dir>ruleset/decoders</decoder_dir>
   <rule_dir>ruleset/rules</rule_dir>
   <rule_exclude>0215-policy_rules.xml</rule_exclude>
   <list>etc/lists/audit-keys</list>
   <list>etc/lists/amazon/aws-eventnames</list>
   <list>etc/lists/security-eventchannel</list>
   <list>etc/lists/malware-hashes</list>
   <list>etc/lists/suspicious-programs</list>
   <list>etc/lists/blacklist-alienvault</list>
```

The firewall-drop command integrates with the Ubuntu local iptables firewall and drops incoming network connection from the attacker endpoint for 60 seconds:

```
<ossec_config>
  <active-response>
    <command>firewall-drop</command>
    <location>local</location>
    <rules_id>100100</rules_id>
    <timeout>60</timeout>
  </active-response>
</ossec_config>
"/var/ossec/etc/ossec.conf" 450L, 12173C
```

```
┌──(root㉿kali)-[/home/kali]
└─# curl http://192.168.140.130
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <!--
    Modified from the Debian original for Ubuntu
    Last updated: 2022-03-22
    See: https://launchpad.net/bugs/1966004
  -->
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>Apache2 Ubuntu Default Page: It works</title>
    <style type="text/css" media="screen">
  * {
    margin: 0px 0px 0px 0px;
    padding: 0px 0px 0px 0px;
  }

  body, html {
    padding: 3px 3px 3px 3px;
```

	Time	agent.name	rule.description	rule.level	rule.id
›	Mar 28, 2024 @ 15:11:40.903	Mergen-Ubuntu	Host Blocked by firewall-drop Active Response	3	651
›	Mar 28, 2024 @ 15:11:39.196	Mergen-Ubuntu	Host-based anomaly detection event (rootcheck).	7	510
›	Mar 28, 2024 @ 15:11:39.067	Mergen-Ubuntu	Host-based anomaly detection event (rootcheck).	7	510
›	Mar 28, 2024 @ 15:11:38.891	Mergen-Ubuntu	IP address found in AlienVault reputation database.	10	100100
›	Mar 28, 2024 @ 15:11:38.620	Mergen-Ubuntu	Host-based anomaly detection event (rootcheck).	7	510
›	Mar 28, 2024 @ 15:11:38.520	Mergen-Ubuntu	Host-based anomaly detection event (rootcheck).	7	510

Detecting account manipulation (FIM)

Here is another scenario where unauthorized account manipulation may allow attackers to gain access to a system. As a persistence technique, adversaries can modify the SSH authorized_keys file in the .ssh directory within a user home directory in a Linux system. For example, for a user named smith, you can find the authorized_keys file located at /home/smith/.ssh/authorized_keys. This file defines the public keys this user uses to login into some of their accounts.

We configured the FIM module to monitor SSH key modification:

<directories whodata="yes">/home/*/.ssh/authorized_keys</directories>

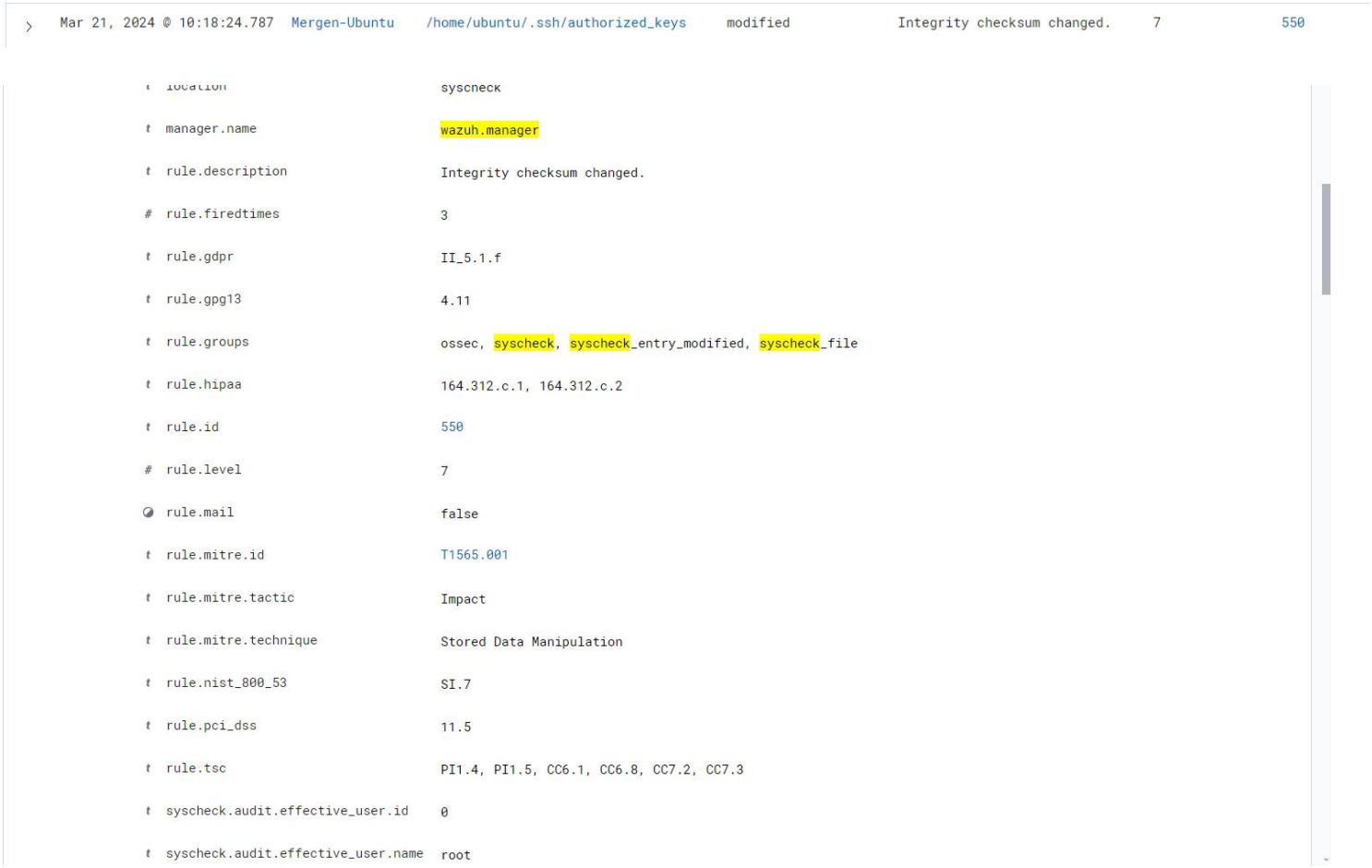
Testing:

Generated an SSH key-pair for user authentication and save it as .ssh/test_key using the following command:

We copied the content of the generated SSH public key test_key.pub and add it to the authorized_keys file in the target Ubuntu user .ssh directory:

We got errors executing the command so we used a different set of commands but functions with the same purpose.

Visualize the alert:



Reporting file changes (FIM)

This test is similar to the initial FIM test conducted that when changes such as creation, modification, or deletion of a file in the monitored file system is detected, it will trigger an alert. The only difference here is the addition of a configuration to exclude changes made to a path or file.

Edited the /var/ossec/etc/ossec.conf configuration file and added the configuration below.

```
<!-- Reporting file changes --!>
<directories check_all="yes" report_changes="yes" realtime="yes" whodata="yes">/appfolder</directories>
<nodiff>/appfolder/private-file.conf</nodiff>
```

Testing:

```
root@ubuntu-virtual-machine:/home/ubuntu# vim /var/ossec/etc/ossec.conf
root@ubuntu-virtual-machine:/home/ubuntu# systemctl restart wazuh-agent
root@ubuntu-virtual-machine:/home/ubuntu# mkdir /appfolder && touch /appfolder/appreport.conf && touch /appfolder/private-file.conf
root@ubuntu-virtual-machine:/home/ubuntu# echo “Bu metn test meqsedlidir!” | tee /appfolder/appreport.conf /appfolder/private-file.conf
bash: !”: event not found
root@ubuntu-virtual-machine:/home/ubuntu# echo “Bu metn test meqsedlidir” | tee /appfolder/appreport.conf /appfolder/private-file.conf
“Bu metn test meqsedlidir”
root@ubuntu-virtual-machine:/home/ubuntu#
```

Visualize:

	Time	agent.name	syscheck.path	syscheck.event	rule.description	rule.level	rule.id	syscheck.diff
›	Mar 21, 2024 @ 10:33:3	Mergen-Ubuntu	/appfolder/private-file.conf	modified	Integrity checksum changed.	7	550	<Diff truncated because nodiff option>
›	Mar 21, 2024 @ 10:33:35.097	Mergen-Ubuntu	/appfolder/appreport.conf	modified	Integrity checksum changed.	7	550	0a1 > “Bu metn test meqsedlidir”
›	Mar 21, 2024 @ 10:32:36.328	Mergen-Ubuntu	/appfolder/appreport.conf	added	File added to the system.	5	554	-
›	Mar 21, 2024 @ 10:32:36.200	Mergen-Ubuntu	/appfolder/private-file.conf	added	File added to the system.	5	554	-

Blocking SSH brute-force attack with active response

One such script is firewall-drop, which leverages iptables on the local endpoint to block malicious IP addresses. Configuring this firewall-drop script as an active response allows Wazuh to automatically execute it when malicious activity is detected, dropping the attacker's connection. This enables agent endpoints to protect themselves in real-time by blocking intruder IPs with the built-in active response capabilities.
First of all, we configured wazuh-manager's ossec.conf file:

```
<ossec_config>

  <command>
    <name>firewall-drop</name>
    <executable>firewall-drop</executable>
    <timeout_allowed>yes</timeout_allowed>
  </command>

  <active-response>
    <command>firewall-drop</command>
    <location>local</location>
    <rules_id>5763</rules_id>
    <timeout>180</timeout>
  </active-response>

</ossec_config>
"/var/ossec/etc/ossec.conf" 431L, 11640C
```

Testing:

```
┌──(kali㉿kali)-[~]
└─$ ping 192.168.1.3
PING 192.168.1.3 (192.168.1.3) 56(84) bytes of data.
64 bytes from 192.168.1.3: icmp_seq=1 ttl=64 time=0.757 ms
64 bytes from 192.168.1.3: icmp_seq=2 ttl=64 time=1.56 ms
64 bytes from 192.168.1.3: icmp_seq=3 ttl=64 time=0.532 ms
^C
— 192.168.1.3 ping statistics —
3 packets transmitted, 3 received, 0% packet loss, time 2022ms
rtt min/avg/max/mdev = 0.532/0.951/1.564/0.443 ms

┌──(kali㉿kali)-[~]
└─$ sudo hydra -l ubuntu -P /usr/share/wordlists/rockyou.txt  192.168.1.3 ssh
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-03-21 02:59:53
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a previous session found, to prevent overwriting, ./hydra.restore
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1/p:14344399), ~896525 tries per task
[DATA] attacking ssh://192.168.1.3:22/
^CThe session file ./hydra.restore was written. Type "hydra -R" to resume session.

┌──(kali㉿kali)-[~]
└─$ ping 192.168.1.3
PING 192.168.1.3 (192.168.1.3) 56(84) bytes of data.
^C
— 192.168.1.3 ping statistics —
4 packets transmitted, 0 received, 100% packet loss, time 3054ms
```

>	Mar 21, 2024 @ 11:00:13.824	Mergen-Ubuntu	Host Blocked by firewall-drop Active Response	3	651
>	Mar 21, 2024 @ 11:00:11.957	Mergen-Ubuntu	sshd: authentication failed.	5	5760
>	Mar 21, 2024 @ 11:00:11.957	Mergen-Ubuntu	sshd: authentication failed.	5	5760
>	Mar 21, 2024 @ 11:00:11.957	Mergen-Ubuntu	sshd: authentication failed.	5	5760
>	Mar 21, 2024 @ 11:00:11.957	Mergen-Ubuntu	sshd: authentication failed.	5	5760
>	Mar 21, 2024 @ 11:00:11.957	Mergen-Ubuntu	sshd: authentication failed.	5	5760
>	Mar 21, 2024 @ 11:00:11.957	Mergen-Ubuntu	sshd: authentication failed.	5	5760
>	Mar 21, 2024 @ 11:00:11.957	Mergen-Ubuntu	sshd: authentication failed.	5	5760
>	Mar 21, 2024 @ 11:00:11.865	Mergen-Ubuntu	sshd: authentication failed.	5	5760
>	Mar 21, 2024 @ 11:00:11.865	Mergen-Ubuntu	sshd: authentication failed.	5	5760

Detecting unauthorized processes

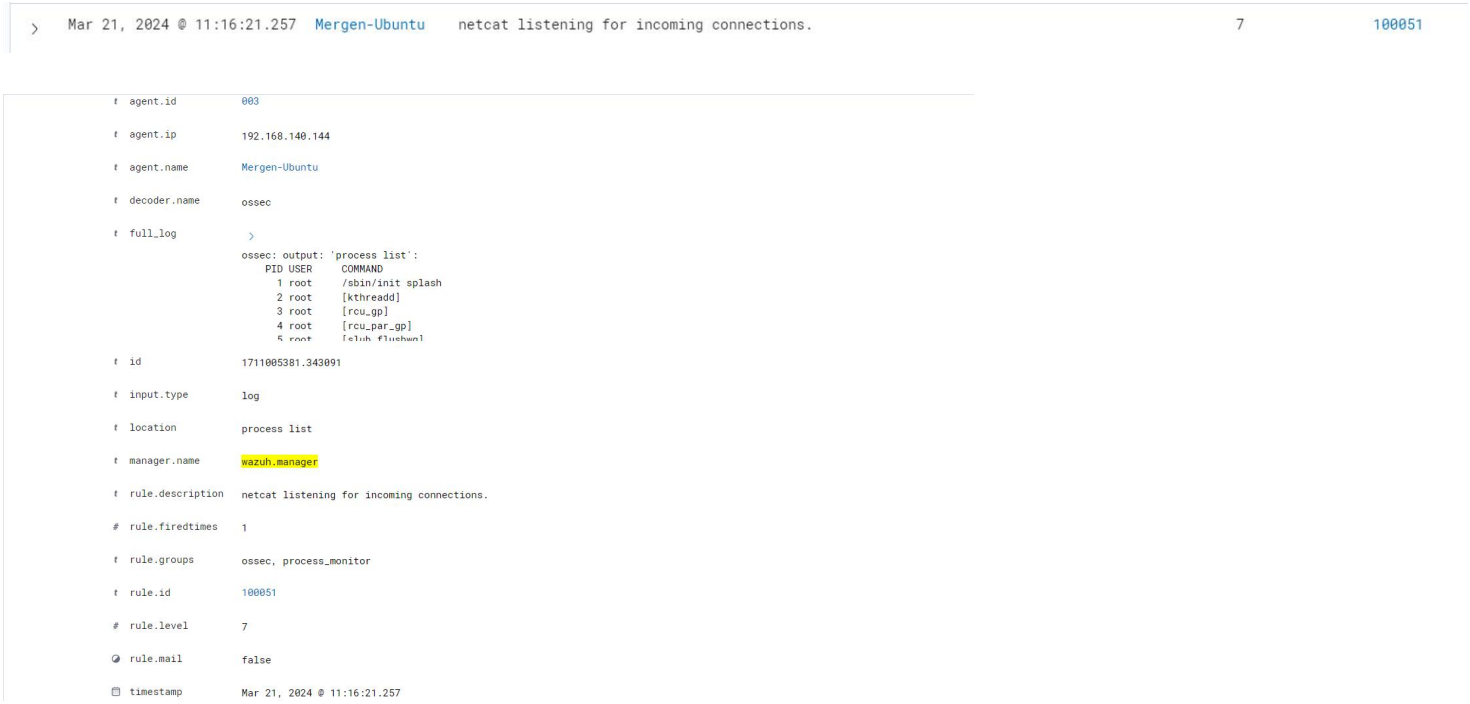
We added the below configuration block to the Wazuh agent /var/ossec/etc/ossec.conf file. This allows to periodically get a list of running processes:

```
<ossec_config>
  <localfile>
    <log_format>full_command</log_format>
    <alias>process list</alias>
    <command>ps -e -o pid,uname,command</command>
    <frequency>30</frequency>
  </localfile>
</ossec_config>
"/var/ossec/etc/ossec.conf" 236L, 6700B
```

We have to create a rule on the Wazuh server that triggers every time the Netcat program launches.

```
<group name="ossec,">
  <rule id="100050" level="0">
    <if_sid>530</if_sid>
    <match>^ossec: output: 'process list'</match>
    <description>List of running processes.</description>
    <group>process_monitor,</group>
  </rule>

  <rule id="100051" level="7" ignore="900">
    <if_sid>100050</if_sid>
    <match>nc -l</match>
    <description>netcat listening for incoming connections.</description>
    <group>process_monitor,</group>
  </rule>
</group>
```

Visualize:


Mar 21, 2024 @ 11:16:21.257 Mergen-Ubuntu netcat listening for incoming connections. 7 100051
t agent.id 003
t agent.ip 192.168.140.144
t agent.name Mergen-Ubuntu
t decoder.name ossec
t full_log
ossec: output: 'process list':
PID USER COMMAND
1 root /sbin/init splash
2 root [kthreadd]
3 root [rcu_gp]
4 root [rcu_par_gp]
t id 1711005381.343091
t input.type log
t location process list
t manager.name wazuh.manager
t rule.description netcat listening for incoming connections.
rule.firedtimes 1
t rule.groups ossec, process_monitor
t rule.id 100051
rule.level 7
rule.mail false
timestamp Mar 21, 2024 @ 11:16:21.257

Network IDS integration

We installed Suricata and rules. Then configured Suricata in suricata.yaml file. We added network interface name and ip. Then we added the below configuration to the /var/ossec/etc/ossec.conf file of the Wazuh agent. This allows the Wazuh agent to read the Suricata logs file:

```
<localfile>
  <log_format>json</log_format>
  <location>/var/log/suricata/eve.json</location>
</localfile>
```

Attack emulation:

We used nmap for testing suricata:

```
┌──(root㉿kali)-[/home/kali]
└─# nmap -sV 192.168.128.121
Starting Nmap 7.94 ( https://nmap.org ) at 2024-03-22 11:07 EDT
Nmap scan report for 192.168.128.121
Host is up (0.0023s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.9p1 Ubuntu 3ubuntu0.6 (Ubuntu Linux; protocol 2.0)
80/tcp open  http    Apache httpd 2.4.52 ((Ubuntu))
MAC Address: 00:0C:29:A7:95:08 (VMware)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 14.06 seconds
```

Visualize:

›	Mar 22, 2024 @ 19:07:53.023	Mergen-Ubuntu	Suricata: Alert - ET SCAN Possible Nmap User-Agent Observed	3	86601
›	Mar 22, 2024 @ 19:07:52.924	Mergen-Ubuntu	Suricata: Alert - ET SCAN Possible Nmap User-Agent Observed	3	86601
›	Mar 22, 2024 @ 19:07:52.870	Mergen-Ubuntu	Suricata: Alert - ET SCAN Possible Nmap User-Agent Observed	3	86601
›	Mar 22, 2024 @ 19:07:52.822	Mergen-Ubuntu	Suricata: Alert - ET SCAN Possible Nmap User-Agent Observed	3	86601

Detecting suspicious binaries

By default, the Wazuh rootcheck module is enabled in the Wazuh agent configuration file and the rootkit_trojans.txt contains signatures of files that rootkits have trojaned. So that's why we just added Reptile rootkit in rootkit_files.txt to specify known file paths to files used by rootkits.

```
#Reptile
reptile/reptile_cmd         ! Reptile rootkit ::
reptile/reptile_rc          ! Reptile rootkit ::
reptile/reptile_shell       ! Reptile rootkit ::
reptile/reptile_start       ! Reptile rootkit ::
lib/udev/reptile            ! Reptile rootkit ::
```

Attack emulation:

```
root@ubuntu-virtual-machine:/home/ubuntu# vim /var/ossec/etc/shared/rootkit_files.txt
root@ubuntu-virtual-machine:/home/ubuntu# vim /var/ossec/etc/shared/rootkit_trojans.txt
root@ubuntu-virtual-machine:/home/ubuntu# systemctl restart wazuh-agent
root@ubuntu-virtual-machine:/home/ubuntu# cp -p /usr/bin/w /usr/bin/w.copy
root@ubuntu-virtual-machine:/home/ubuntu# tee /usr/bin/w << EOF
> #!/bin/bash
> echo "`date` this is evil" > /tmp/trojan_created_file
> echo 'test for /usr/bin/w trojaned file' >> /tmp/trojan_created_file
> /usr/bin/w.copy
> EOF
#!/bin/bash
echo "cümə, 22 mart 2024 19:41:59 +04 this is evil" > /tmp/trojan_created_file
echo 'test for /usr/bin/w trojaned file' >> /tmp/trojan_created_file
/usr/bin/w.copy
```

	Time	agent.name	rule.description	rule.level	rule.id
›	Mar 22, 2024 @ 19:44:08.841	Mergen-Ubuntu	Host-based anomaly detection event (rootcheck).	7	510
›	Mar 22, 2024 @ 19:44:08.685	Mergen-Ubuntu	Host-based anomaly detection event (rootcheck).	7	510
›	Mar 22, 2024 @ 19:44:06.178	Mergen-Ubuntu	Host-based anomaly detection event (rootcheck).	7	510
›	Mar 22, 2024 @ 19:44:06.178	Mergen-Ubuntu	Host-based anomaly detection event (rootcheck).	7	510

t	_index	wazuh-alerts-4.x-2024.03.22
t	agent.id	003
t	agent.ip	192.168.140.144
t	agent.name	Mergen-Ubuntu
t	data.file	/usr/bin/w
t	data.title	Trojaned version of file detected.
t	decoder.name	rootcheck
t	full_log	Trojaned version of file '/usr/bin/w' detected. Signature used: 'uname -a|proc\.h|bash' (Generic).
t	id	1711122248.136865
t	input.type	log
t	location	rootcheck
t	manager.name	wazuh.manager
t	rule.description	Host-based anomaly detection event (rootcheck).
#	rule.firedtimes	34
t	rule.gdpr	IV_35.7.d
t	rule.groups	ossec, rootcheck

Monitoring execution of malicious commands

In this scenario, I set up Auditd on an Ubuntu device to track and log all commands executed by a specific user, even those performed with elevated privileges or as the root user.

We showed configuration the below and also we added audit.log localfile in ossec.conf

```
<localfile>
  <log_format>audit</log_format>
  <location>/var/log/audit/audit.log</location>
</localfile>
```

```
root@ubuntu-virtual-machine:/home/ubuntu# echo "-a exit,always -F auid=1000 -F egid!=994 -F auid!=-1 -F arch=b32 -S execve -k audit-wazuh-c" >> /etc/audit/audit.rules
root@ubuntu-virtual-machine:/home/ubuntu# echo "-a exit,always -F auid=1000 -F egid!=994 -F auid!=-1 -F arch=b64 -S execve -k audit-wazuh-c" >> /etc/audit/audit.rules
root@ubuntu-virtual-machine:/home/ubuntu# auditctl -R /etc/audit/audit.rules
No rules
enabled 1
failure 1
pid 574
rate_limit 0
backlog_limit 8192
lost 0
backlog 4
backlog_wait_time 60000
backlog_wait_time_actual 0
enabled 1
failure 1
pid 574
rate_limit 0
backlog_limit 8192
lost 0
backlog 4
backlog_wait_time 60000
backlog_wait_time_actual 0
enabled 1
failure 1
pid 574
rate_limit 0
backlog_limit 8192
lost 0
backlog 4
backlog_wait_time 60000
backlog_wait_time_actual 0
root@ubuntu-virtual-machine:/home/ubuntu# auditctl -l
-a always,exit -F arch=b32 -S execve -F auid=1000 -F egid!=994 -F auid!=-1 -F key=audit-wazuh-c
-a always,exit -F arch=b64 -S execve -F auid=1000 -F egid!=994 -F auid!=-1 -F key=audit-wazuh-c
-w /appfolder -p wa -k wazuh_fim
-w /home/ubuntu/.ssh/authorized_keys -p wa -k wazuh_fim
-w /tmp -p wa -k wazuh_fim
```

We created new file name is suspicious-programs and added some CDB lists:

```
root@wazuh:/# cat /var/ossec/etc/lists/suspicious-programs
ncat:yellow
nc:red
tcpdump:orange
nmap:purple
root@wazuh:/#
```

And added the list to the <ruleset> section of the Wazuh server /var/ossec/etc/ossec.conf file:

```
<ruleset>
  <!-- Default ruleset -->
  <decoder_dir>ruleset/decoders</decoder_dir>
  <rule_dir>ruleset/rules</rule_dir>
  <rule_exclude>0215-policy_rules.xml</rule_exclude>
  <list>etc/lists/audit-keys</list>
  <list>etc/lists/amazon/aws-eventnames</list>
  <list>etc/lists/security-eventchannel</list>
  <list>etc/lists/malware-hashes</list>
  <list>etc/lists/suspicious-programs</list>
```

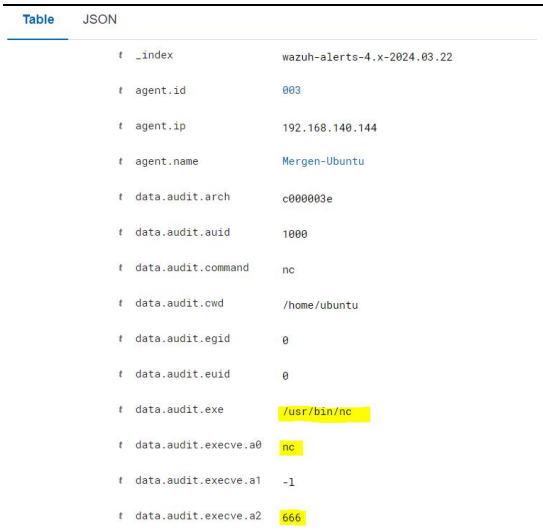
Lastly, we need to trigger rule for it:

```
<group name="audit">
  <rule id="100210" level="12">
    <if_sid>80792</if_sid>
    <list field="audit.command" lookup="match_key_value" check_value="red">etc/lists/suspicious-programs</list>
    <description>Audit: Highly Suspicious Command executed: $(audit.exe)</description>
    <group>audit_command,</group>
  </rule>
</group>
"/var/ossec/etc/rules/local_rules.xml" 182L, 7074C
```

Attack emulation:

We started nc -l 666 so in rule we added if it is red show it

Time	agent.name	rule.description	rule.level	rule.id
Mar 22, 2024 @ 20:14:4	Mergen-Ubuntu	Audit: Highly Suspicious Command executed: /usr/bin/nc	12	100210



Furthermore, we created tcpdump as orange when we started tcpdump Wazuh visualize like that:

You can see tcpdump not red so it shows it with rule.level:3

Monitoring file and directory access

We configured audit.rules in home directory

```
root@ubuntu-virtual-machine:/home/ubuntu# echo "-w /home -p w -k audit-wazuh-w" >> /etc/audit/audit.rules
root@ubuntu-virtual-machine:/home/ubuntu# echo "-w /home -p a -k audit-wazuh-a" >> /etc/audit/audit.rules
root@ubuntu-virtual-machine:/home/ubuntu# echo "-w /home -p r -k audit-wazuh-r" >> /etc/audit/audit.rules
root@ubuntu-virtual-machine:/home/ubuntu# echo "-w /home -p x -k audit-wazuh-x" >> /etc/audit/audit.rules
root@ubuntu-virtual-machine:/home/ubuntu# auditctl -R /etc/audit/audit.rules
No rules
enabled 1
failure 1
pid 574
rate_limit 0
backlog_limit 8192
lost 0
backlog 4
backlog_wait_time 60000
backlog_wait_time_actual 0
enabled 1
failure 1
pid 574
rate_limit 0
backlog_limit 8192
lost 0
backlog 4
backlog_wait_time 60000
backlog_wait_time_actual 0
enabled 1
failure 1
pid 574
rate_limit 0
backlog_limit 8192
lost 0
backlog 4
backlog_wait_time 60000
backlog_wait_time_actual 0
root@ubuntu-virtual-machine:/home/ubuntu# auditctl -l
-a always,exit -F arch=b32 -S execve -F auid=1000 -F egid!=994 -F auid!=-1 -F key=audit-wazuh-c
-a always,exit -F arch=b64 -S execve -F auid=1000 -F egid!=994 -F auid!=-1 -F key=audit-wazuh-c
-w /home -p w -k audit-wazuh-w
-w /home -p a -k audit-wazuh-a
-w /home -p r -k audit-wazuh-r
-w /home -p x -k audit-wazuh-x
-w /appfolder -p wa -k wazuh_fim
-w /home/ubuntu/.ssh/authorized_keys -p wa -k wazuh_fim
-w /tmp -p wa -k wazuh_fim
```

Testing:

We created just malware.py with sudo

```
root@ubuntu-virtual-machine:/home/ubuntu# exit
exit
ubuntu@ubuntu-virtual-machine:~$ sudo touch /home/malware.py
[sudo] password for ubuntu:
ubuntu@ubuntu-virtual-machine:~$
```

Visualize:

›	Mar 22, 2024 @ 21:44:47.765	Mergen-Ubuntu	Audit: Created: /home/malware.py.	⊕ ⊖	3	80790
›	Mar 22, 2024 @ 21:44:47.765	Mergen-Ubuntu	Audit: Command: /usr/bin/touch.		3	80792

Monitoring commands run as root

We added the rules below in the /etc/audit/audit.rules audit rule file:

```
root@ubuntu-virtual-machine:/home/ubuntu# echo "-w /home -p w -k audit-wazuh-w" >> /etc/audit/audit.rules
root@ubuntu-virtual-machine:/home/ubuntu# echo "-w /home -p a -k audit-wazuh-a" >> /etc/audit/audit.rules
root@ubuntu-virtual-machine:/home/ubuntu# echo "-w /home -p r -k audit-wazuh-r" >> /etc/audit/audit.rules
root@ubuntu-virtual-machine:/home/ubuntu# echo "-w /home -p x -k audit-wazuh-x" >> /etc/audit/audit.rules
root@ubuntu-virtual-machine:/home/ubuntu# auditctl -R /etc/audit/audit.rules
No rules
enabled 1
failure 1
pid 574
rate_limit 0
backlog_limit 8192
lost 0
backlog 4
backlog_wait_time 60000
backlog_wait_time_actual 0
enabled 1
failure 1
pid 574
rate_limit 0
backlog_limit 8192
lost 0
backlog 4
backlog_wait_time 60000
backlog_wait_time_actual 0
enabled 1
failure 1
pid 574
rate_limit 0
backlog_limit 8192
lost 0
backlog 4
backlog_wait_time 60000
backlog_wait_time_actual 0
root@ubuntu-virtual-machine:/home/ubuntu# auditctl -l
-a always,exit -F arch=b32 -S execve -F auid=1000 -F egid!=994 -F auid!=-1 -F key=audit-wazuh-c
-a always,exit -F arch=b64 -S execve -F auid=1000 -F egid!=994 -F auid!=-1 -F key=audit-wazuh-c
-w /home -p w -k audit-wazuh-w
-w /home -p a -k audit-wazuh-a
-w /home -p r -k audit-wazuh-r
-w /home -p x -k audit-wazuh-x
-w /appfolder -p wa -k wazuh_fim
-w /home/ubuntu/.ssh/authorized_keys -p wa -k wazuh_fim
-w /tmp -p wa -k wazuh_fim
```

Testing:

```
root@ubuntu-virtual-machine:/home/ubuntu# exit
exit
ubuntu@ubuntu-virtual-machine:~$ sudo touch /home/malware.py
[sudo] password for ubuntu:
ubuntu@ubuntu-virtual-machine:~$
```

Visualize:

›	Mar 22, 2024 @ 21:44:47.765	Mergen-Ubuntu	Audit: Created: /home/malware.py.	⊕ ⊖	3	80790
›	Mar 22, 2024 @ 21:44:47.765	Mergen-Ubuntu	Audit: Command: /usr/bin/touch.		3	80792

Privilege abuse

We created the two users, ulgen and erlik, on the Ubuntu endpoint:

```
root@ubuntu-virtual-machine:/home/ubuntu# useradd ulgen
root@ubuntu-virtual-machine:/home/ubuntu# useradd erlik
```

```
root@ubuntu-virtual-machine:/home/ubuntu# mkdir /home/ulgen
root@ubuntu-virtual-machine:/home/ubuntu# chown ulgen:ulgen /home/ulgen
root@ubuntu-virtual-machine:/home/ubuntu#
```

We edited the audit rule file /etc/audit/audit.rules and add the following configuration:

```
root@ubuntu-virtual-machine:/home/ubuntu# echo "-a always,exit -S open -S openat -F dir=/home/jane/ -F perm=rwa -F auid>=1000 -F auid!=-1 -F euid!=<EUID_OF_JANE> -F uid!=0 -C auid!=
obj_uid -F key=power_abuse">>/etc/audit/audit.rules
root@ubuntu-virtual-machine:/home/ubuntu# id -u ulgen
1001
root@ubuntu-virtual-machine:/home/ubuntu# auditctl -R /etc/audit/audit.rules
No rules
enabled 1
failure 1
pid 574
rate_limit 0
backlog_limit 8192
lost 0
backlog 4
backlog_wait_time 60000
backlog_wait_time_actual 0
enabled 1
failure 1
pid 574
rate_limit 0
backlog_limit 8192
lost 0
backlog 4
backlog_wait_time 60000
backlog_wait_time_actual 0
enabled 1
failure 1
pid 574
rate_limit 0
backlog_limit 8192
lost 0
backlog 4
backlog_wait_time 60000
backlog_wait_time_actual 0
WARNING - 32/64 bit syscall mismatch in line 15, you should specify an arch
Unknown user: <EUID_OF_JANE>
There was an error in line 15 of /etc/audit/audit.rules
root@ubuntu-virtual-machine:/home/ubuntu# auditctl -l
-a always,exit -F arch=b32 -S execve -F auid=1000 -F egid!=994 -F auid!=-1 -F key=audit-wazuh-c
-a always,exit -F arch=b64 -S execve -F auid=1000 -F egid!=994 -F auid!=-1 -F key=audit-wazuh-c
-w /home -p w -k audit-wazuh-w
-w /home -p a -k audit-wazuh-a
-w /home -p r -k audit-wazuh-r
-w /home -p x -k audit-wazuh-x
-a always,exit -F arch=b64 -S execve -F euid=0 -F key=audit-wazuh-c
-a always,exit -F arch=b32 -S execve -F euid=0 -F key=audit-wazuh-c
```

Updated the /var/ossec/etc/lists/audit-keys CDB list with the custom audit key:

```
root@wazuh:/# echo "power_abuse:abuse" >> /var/ossec/etc/lists/audit-keys
root@wazuh:/#
```

Added the following rule to the custom:

```
<group name="audit">
  <rule id="100210" level="8">
    <if_sid>80700</if_sid>
    <list field="audit.key" lookup="match_key_value" check_value="abuse">etc/lists/audit-keys</list>
    <description>Audit: User with uid $(audit.uid) trying to access $(audit.directory.name) files.</description>
    <group>audit_command,</group>
  </rule>
</group>
"/var/ossec/etc/rules/local_rules.xml" 191L, 7421C
```

Testing:

```
root@ubuntu-virtual-machine:/home/ubuntu# su erlik
$ ls /home/ulgen
$
```

Visualize:

Time	agent.name	rule.description	rule.level	rule.id
Mar 22, 2024 @ 22:26:05.190	Mergen-Ubuntu	Audit: Watch - Read access: /home/ulgen.	3	80785

t	data.audit.tty	pts3
t	data.audit.type	SYSCALL
t	data.audit.uid	1002
t	decoder.name	auditd
t	decoder.parent	auditd
t	full_log	type=SYSCALL msg=audit(1711131964.768:21713): arch=c000003e syscall=257 success=yes exit=3 a0=ffffff9c a1=5feae33e7a60 a2=90800 a3=0 items=1 ppid=147020 pid=147030 auid=1000 uid=1002 gid=1002 euid=1002 suid=1002 fsuid=1002 egid=1002 sgid=1002 fsgid=1002 tty=pts3 ses=3 comm="ls" exe="/usr/bin/ls" subj=unconfined key="audit-wazuh-r" ARCH=x86_64 SYSCALL=openat AUID="ubuntu" UID="erlik" GID="erlik" EUID="erlik" SUID="erlik" FSUID="erlik" EGID="erlik" SGID="erlik" FSGID="erlik" type=CWD msg=audit(1711131964.768:21713): cwd="/home/ubuntu" type=PATH msg=audit(1711131964.768:21713): item=0 name="/home/ulgen" inode=817997 dev=08:03 mode=040755 ouid=1001 ogid=1001 rdev=00:00 nametype=NORMAL cap_fp=0 cap_fi=0 cap_fe=0 cap_fver=0 cap_frootid=0 OUID="ulgen" OGID="ulgen" type=PROCTITLE msg=audit(1711131964.768:21713): proctitle=6C73002F686F6
t	id	1711131965.37721352
t	input.type	log
t	location	/var/log/audit/audit.log
t	manager.name	wazuh.manager
t	rule.description	Audit: Watch - Read access: /home/ulgen.
#	rule.firedtimes	19
t	rule.gdpr	IV_30.1.g
t	rule.groups	audit, audit_watch_read
t	rule.id	80785
#	rule.level	3

Detecting keyword in a file

We created the test file and add some text to it, including the phrase password_enabled: yes:

```
root@ubuntu-virtual-machine:/home/ubuntu# echo -e "config_file\nsecond line of configuration\npassword_enabled: yes" > /usr/share/testfile.txt
root@ubuntu-virtual-machine:/home/ubuntu# cat /usr/share/testfile.txt
config_file
second line of configuration
password_enabled: yes
```

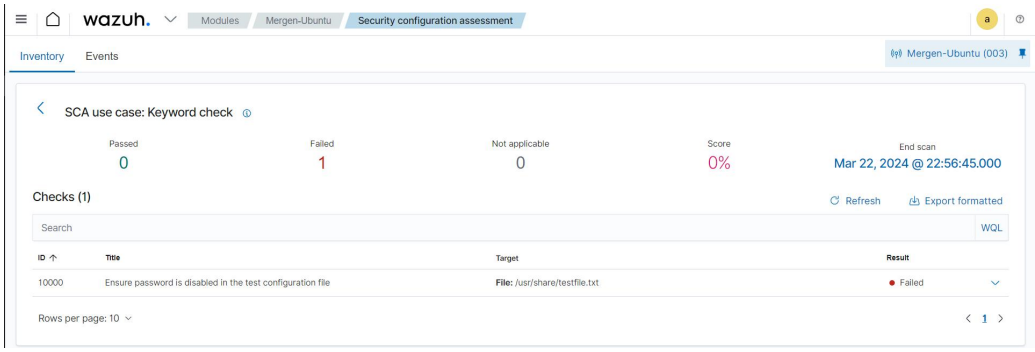
Created a new directory to a new SCA policy file /var/ossec/etc/custom-sca-files/keywordcheck.yml and add the following content to it:

```
root@ubuntu-virtual-machine:/var/ossec/etc/custom-sca-files# vim keywordcheck.yml
root@ubuntu-virtual-machine:/var/ossec/etc/custom-sca-files# chown wazuh:wazuh /var/ossec/etc/custom-sca-files/keywordcheck.yml
root@ubuntu-virtual-machine:/var/ossec/etc/custom-sca-files# vim /var/ossec/etc/ossec.conf
root@ubuntu-virtual-machine:/var/ossec/etc/custom-sca-files# systemctl restart wazuh-agent
root@ubuntu-virtual-machine:/var/ossec/etc/custom-sca-files# cat keywordcheck.yml
policy:
  id: "keyword_check"
  file: "keywordcheck.yml"
  name: "SCA use case: Keyword check"
  description: "Guidance for checking for a keyword or phrase in files on Ubuntu endpoints."
  references:
    - https://documentation.wazuh.com/current/user-manual/capabilities/sec-config-assessment/index.html
    - https://documentation.wazuh.com/current/user-manual/capabilities/sec-config-assessment/creating-custom-policies.html

requirements:
  title: "Check that the desired file exists on the monitored endpoints"
  description: "Requirements for running the SCA scans against endpoints with testfile.txt on them."
  condition: any
  rules:
    - 'f:/usr/share/testfile.txt'

checks:
  - id: 10000
    title: "Ensure password is disabled in the test configuration file"
    description: "Password is enabled in the test configuration file."
    rationale: "Password is considered weak for the custom test application. Threat actors can brute-force your password."
    remediation: "Disable password by setting the value of the password_enabled option to no."
    condition: none
    rules:
      - 'f:/usr/share/testfile.txt -> r:^password_enabled: yes$'
root@ubuntu-virtual-machine:/var/ossec/etc/custom-sca-files#
```

In the ossec.conf file we added sca policy:

```
<sca>
  <policies>
    <policy enabled="yes">/var/ossec/etc/custom-sca-files/keywordcheck.yml</policy>
  </policies>
</sca>
```

Visualize:



Detecting a running process

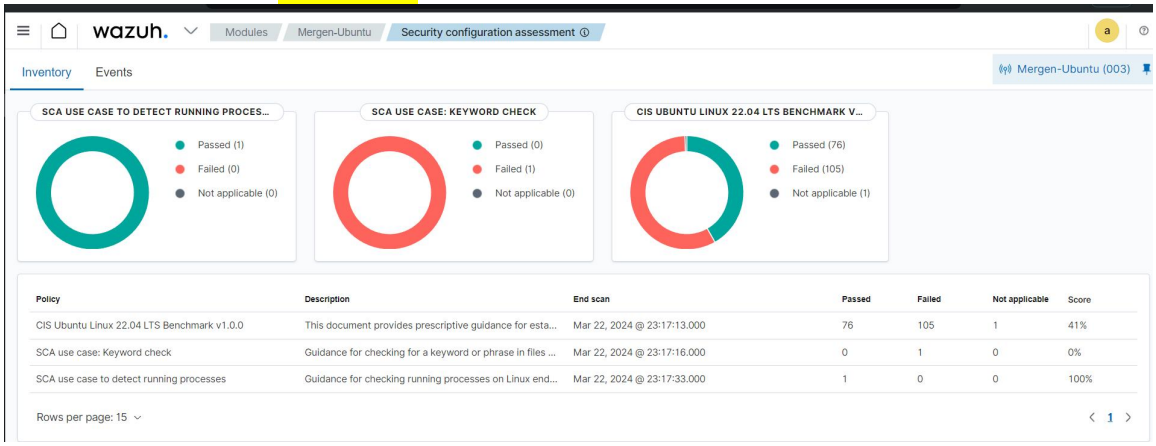
Created a new SCA policy file /var/ossec/etc/custom-sca-files/processcheck.yml and add the following content to it:

```
root@ubuntu-virtual-machine:/var/ossec/etc/custom-sca-files# vim processcheck.yml
root@ubuntu-virtual-machine:/var/ossec/etc/custom-sca-files# chown wazuh:wazuh /var/ossec/etc/custom-sca-files/processcheck.yml
root@ubuntu-virtual-machine:/var/ossec/etc/custom-sca-files# cat processcheck.yml
policy:
  id: "process_check"
  file: "processcheck.yml"
  name: "SCA use case to detect running processes"
  description: "Guidance for checking running processes on Linux endpoints."
  references:
    - https://documentation.wazuh.com/current/user-manual/capabilities/sec-config-assessment/index.html
    - https://documentation.wazuh.com/current/user-manual/capabilities/sec-config-assessment/creating-custom-policies.html

requirements:
  title: "Check that the SSH service and password-related files are present on the system"
  description: "Requirements for running the SCA scan against the Unix based systems policy."
  condition: any
  rules:
    - "f:$sshd_file"
    - "f:/etc/passwd"
    - "f:/etc/shadow"

variables:
  $sshd_file: /etc/ssh/sshd_config

checks:
  - id: 10003
    title: "Ensure that netcat is not running on your endpoint"
    description: "Netcat is running on your endpoint."
    rationale: "Threat actors can use netcat to open ports on your endpoints or to connect to remote servers."
    remediation: "Kill the netcat process if confirmed to be malicious after further investigation."
    condition: none
    rules:
      - 'p:nc'
      - 'p:netcat'
root@ubuntu-virtual-machine:/var/ossec/etc/custom-sca-files#
```

We added this policy in ossec.conf file in wazuh-agent:

```
<sca>
  <policies>
    <policy enabled="yes">/var/ossec/etc/custom-sca-files/processcheck.yml</policy>
  </policies>
</sca>
```

Visualize:
When we started nc -l 666 it shows:



Disabling a Linux user account with active response

Added the rule below to the Wazuh server /var/ossec/etc/rules/local_rules.xml file:

```
<group name="pam,syslog,">
  <rule id="100101" level="10" frequency="3" timeframe="120">
    <if_matched_sid>5503</if_matched_sid>
    <description>Possible password guess on $(dstuser): 3 failed logins in a short period of time</description>
    <mitre>
      <id>T1110</id>
    </mitre>
  </rule>
</group>
"/var/ossec/etc/rules/local_rules.xml" 210L, 8028C
```

Ossec file:

```
<ossec_config>

<command>
  <name>disable-account</name>
    <executable>disable-account</executable>
    <timeout_allowed>yes</timeout_allowed>
  </command>

  <active-response>
    <command>disable-account</command>
    <location>local</location>
    <rules_id>100101</rules_id>
    <timeout>300</timeout>
  </active-response>
</ossec_config>
"/var/ossec/etc/ossec.conf" 466L, 12519C
```

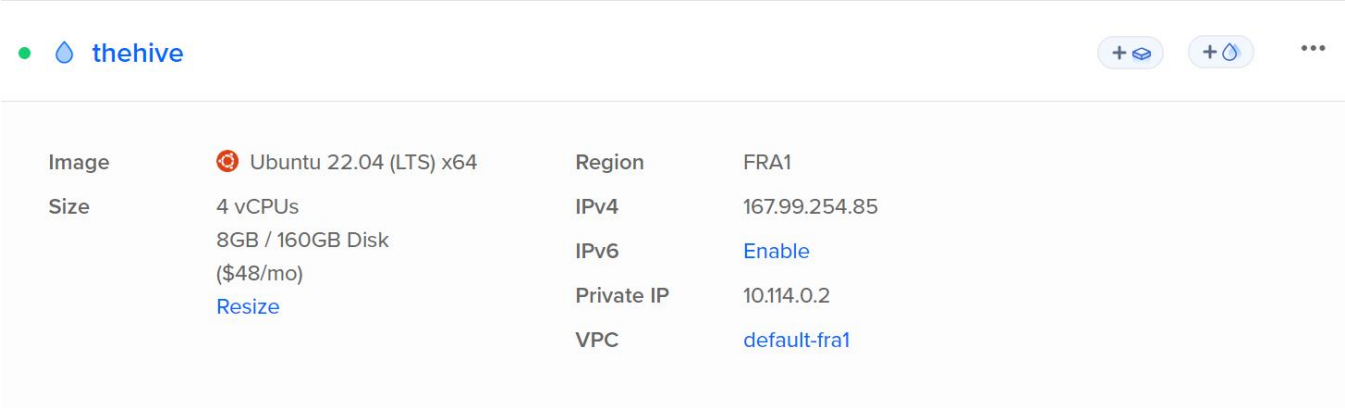
Testing:

```
user2@ubuntu-virtual-machine:/home/ubuntu$ su user2
Password:
su: Authentication failure
user2@ubuntu-virtual-machine:/home/ubuntu$ su user2
Password:
su: Authentication failure
user2@ubuntu-virtual-machine:/home/ubuntu$ su user2
Password:
su: Authentication failure
user2@ubuntu-virtual-machine:/home/ubuntu$ su ubuntu
Password:
ubuntu@ubuntu-virtual-machine:~$ sudo passwd --status user2
user2 L 03/28/2024 0 99999 7 -1
```

›	Mar 28, 2024 @ 15:53:06.184	Mergen-Ubuntu	User missed the password to change UID (user id).	5	5301
›	Mar 28, 2024 @ 15:53:06.172	Mergen-Ubuntu	Active response: active-response/bin/disable-account - add	3	657
›	Mar 28, 2024 @ 15:53:0⊕ ⊖	Mergen-Ubuntu	Possible password guess on user2: 3 failed logins in a short period of time	10	100101
›	Mar 28, 2024 @ 15:52:52.144	Mergen-Ubuntu	User missed the password to change UID (user id).	5	5301
›	Mar 28, 2024 @ 15:52:50.240	Mergen-Ubuntu	PAM: User login failed.	5	5503
›	Mar 28, 2024 @ 15:52:46.236	Mergen-Ubuntu	User missed the password to change UID (user id).	5	5301
›	Mar 28, 2024 @ 15:52:44.276	Mergen-Ubuntu	PAM: User login failed.	5	5503

The hive 5.2 deployment

We raised up Ubuntu 22.04 on Digital Ocean platform for the deployment of The hive. You can accesses machine via SSH by using ip address and password.
Password:!123Salam
The hive admin panel: http://167.99.254.85:9000
Analyst:
Username: alert@code.edu.az Password: secret
Organization Admin:
Username: admin@code.edu.az Password:secret
Local Admin:
Username: admin@thehive.local Password:secret



Let's get started. We deployed The hive with docker.

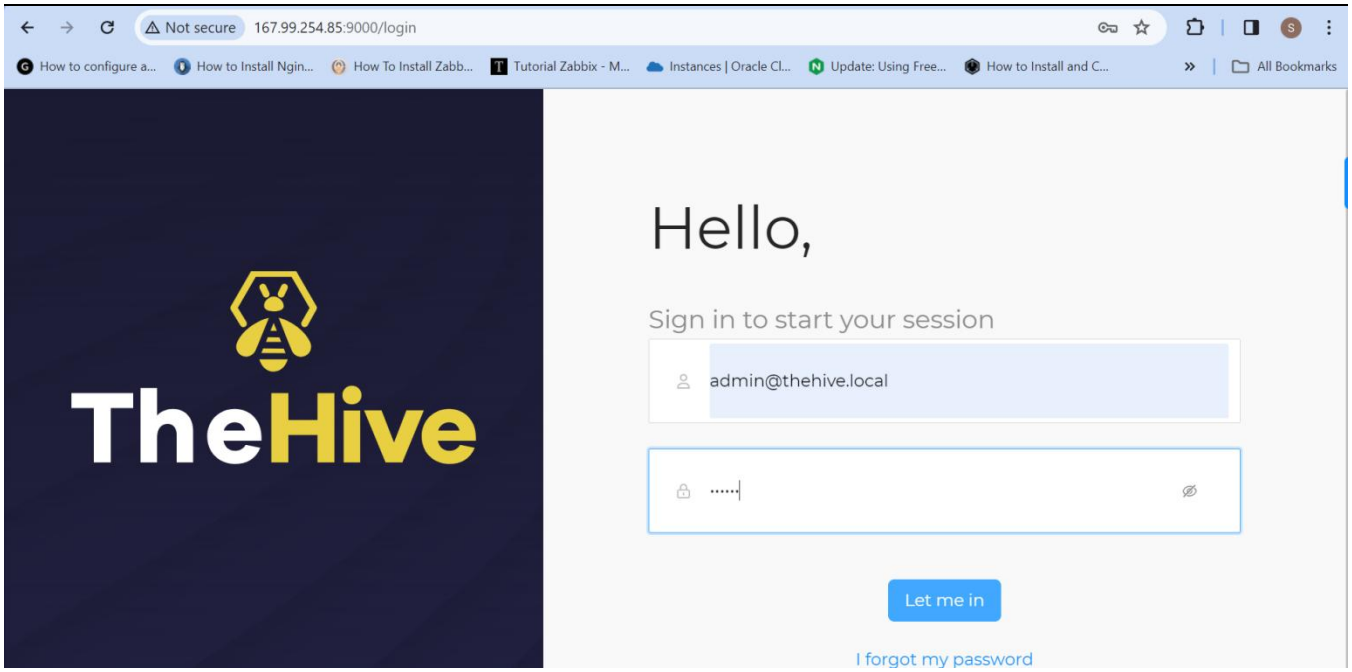
docker pull strangebee/thehive:5.2

docker run –d –p 9000:9000 image id

```
root@thehive:~# docker image ls
REPOSITORY           TAG       IMAGE ID       CREATED         SIZE
strangebee/thehive   5.2       e8afa23c4b0e   2 weeks ago     937MB
strangebee/thehive   5.0       b8c853e335af   13 months ago   755MB
root@thehive:~# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED      STATUS      PORTS                                                  NAMES
a6f10e093f48   e8afa23c4b0e   "/opt/thehive/entryp…"   4 days ago   Up 4 days   0.0.0.0:9000->9000/tcp, :::9000->9000/tcp   keen_mcca
rthy
```

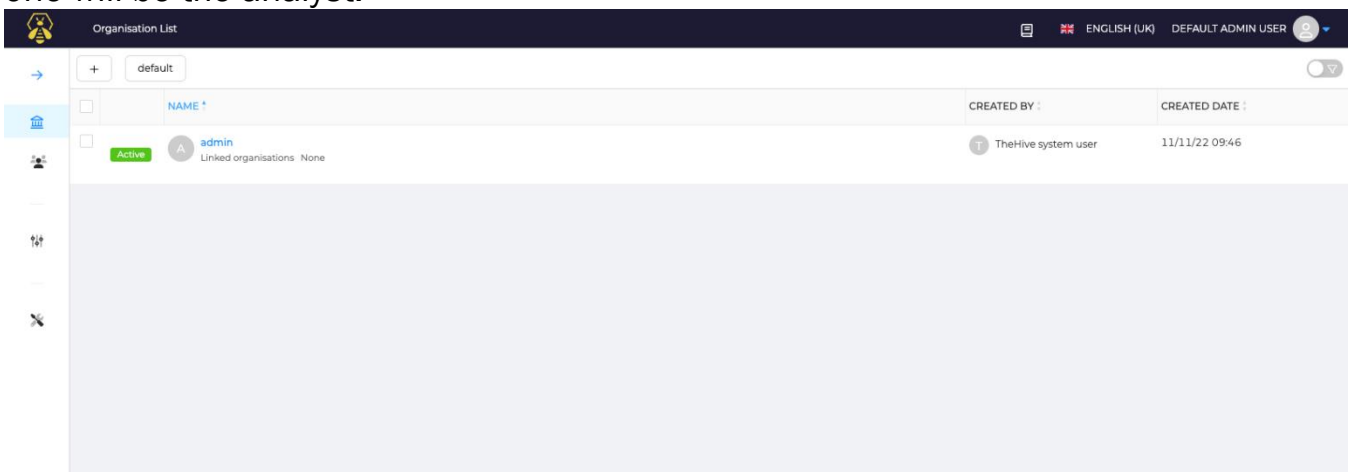
As you can see now we can enter the admin panel.

Let's get in.

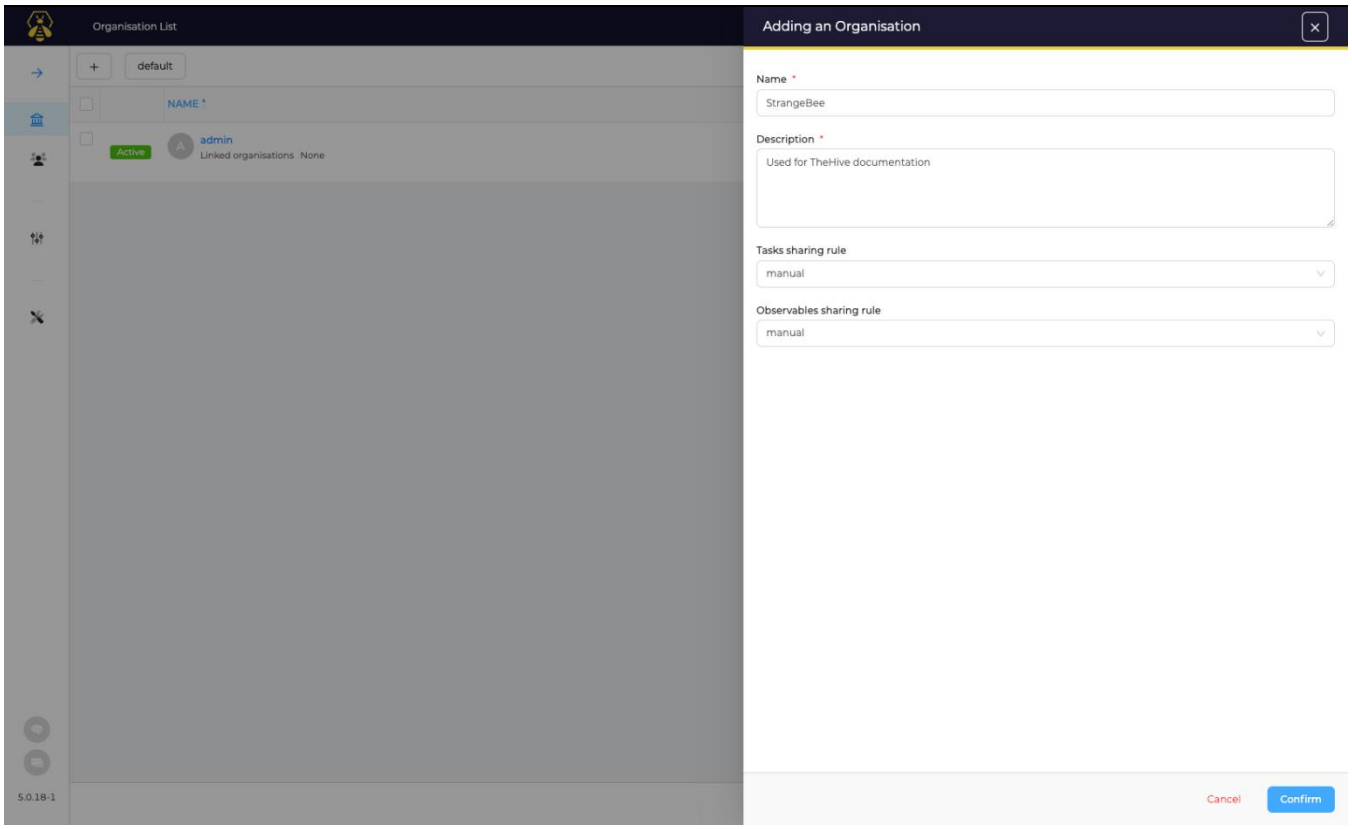


I am going to enter the hive interface as a local admin. Because we didn't create any organization and user yet.

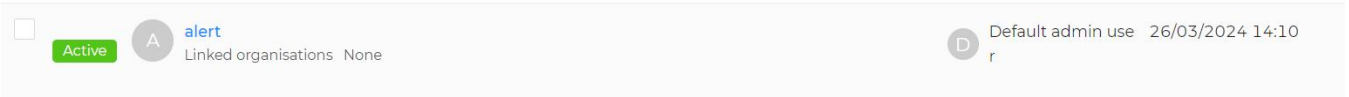
There is one default organization in the interface and we can only create one more organization because of free trial. I am going to create new organization called alert and I will add 2 users. One of them is going to be organization admin and another one will be the analyst.



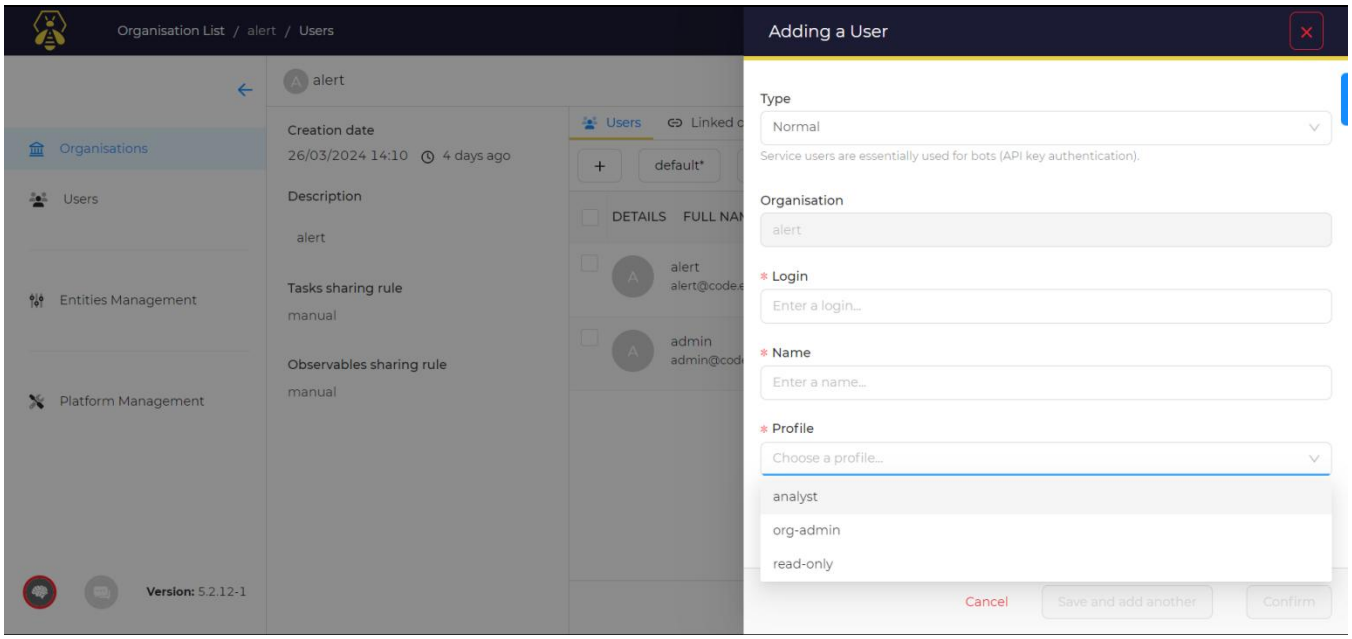
There is plus button at the left top , You need to click that button in order to add organization. Then the windows pops up and you need to feel the blanks with your data and click confirm.



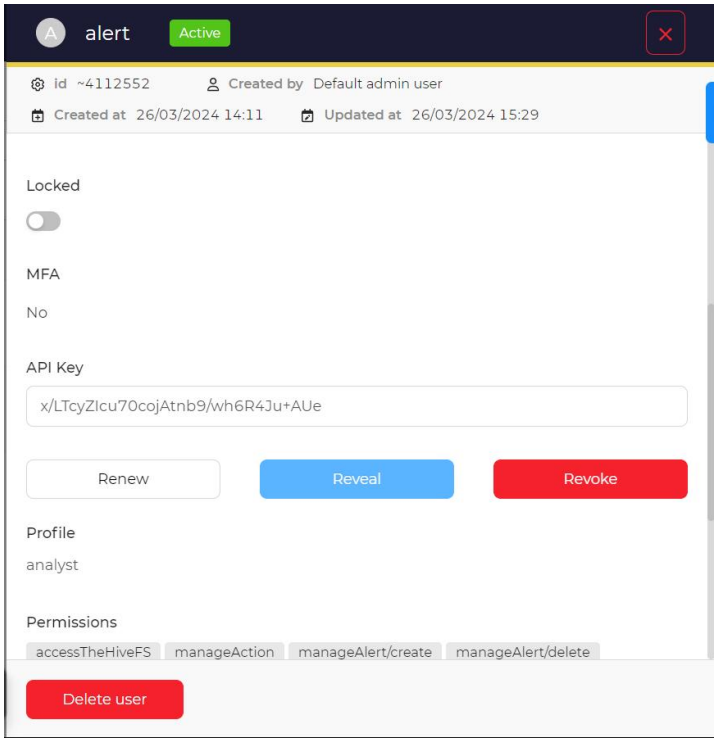
I have created organization called Alert.



Let’s create users. It’s as easy as creating new organization. You should fill the blanks with your data and choose the role of your user. Unfortunately in free trial we are able to create only one organization and 2 users.

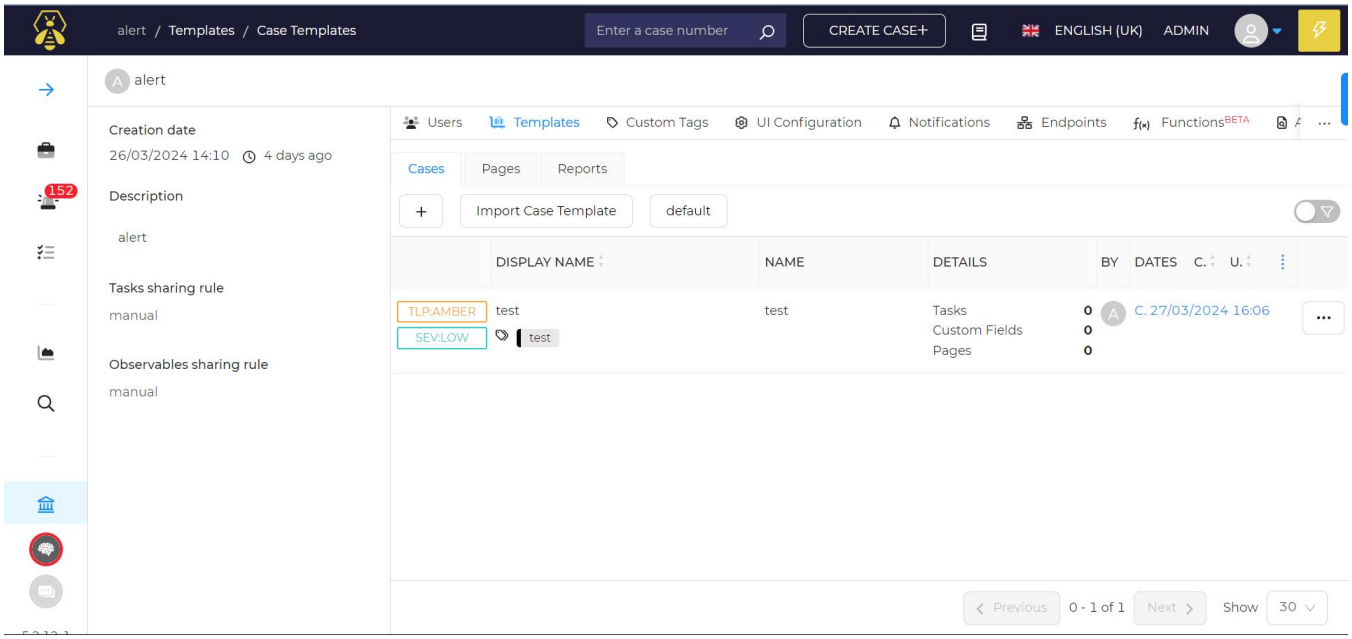


Then you should create API key for the analyst user. And note it down because you will be in need of that a little bit later.



One more thing left. We should create case template so let's switch to organization admin user. We will need that while creating workflow in shuffle.

You should click organization button at the left bottom side. Then Switch it templates from user and click plus sign. That is all we need to do. As you can see I have already created template.



That is all for The Hive.

Cortex deployment and adding scanners and responders.

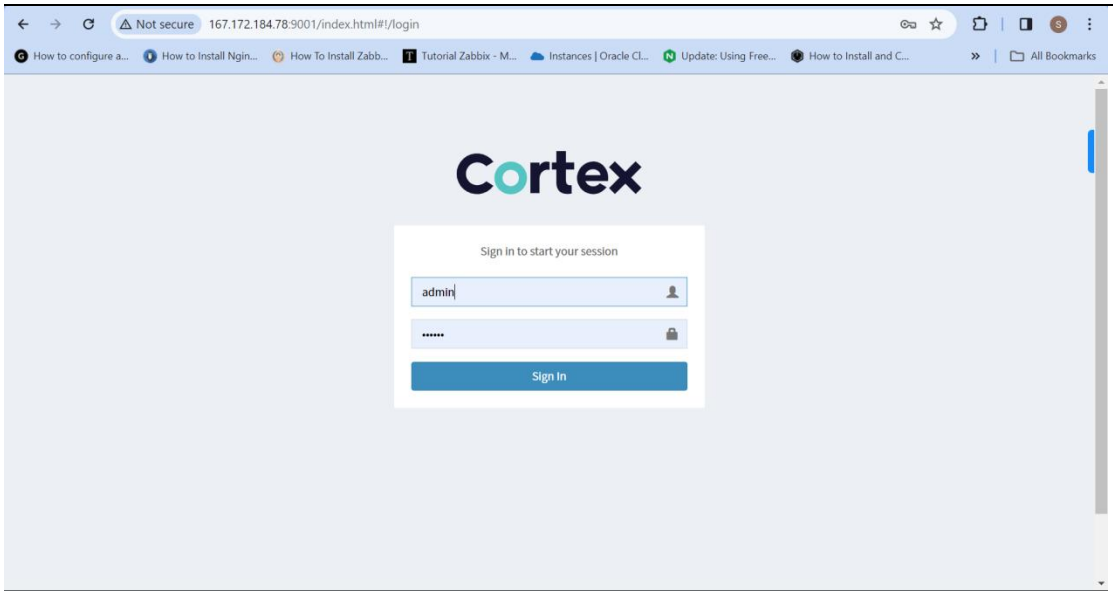
Cortex is the brain of the Hive, you can do CTI by Cortex. Unfortunately we are using free trial of the hive and it's not compatible with Cortex but it's not a big problem because I will make it to work separately from The Hive and it will be 90% the same result. You will see how I am going to do it in the next pages now worries.

So We deployed Cortex on Digital Ocean Cloud platform like the other ones. You can access the machine via ssh by using ip and password. Password: !123Salam
url: http://167.172.184.78:9001
Local Admin
Username: admin Password: secret
Organization Admin
Username: admin@cortex.com Password: secret
Analyst
Username: alert@cortex.com Password: secret

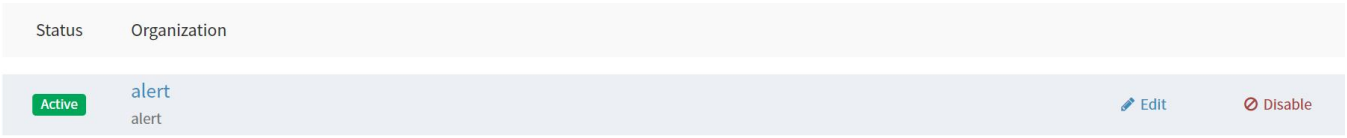
All you need to use this installation script and choose one of the options. I choose 3rd and didn't face any problem.

`wget -q -O /tmp/install.sh https://archives.strangebee.com/scripts/install.sh ; sudo -v ; bash /tmp/install.sh`

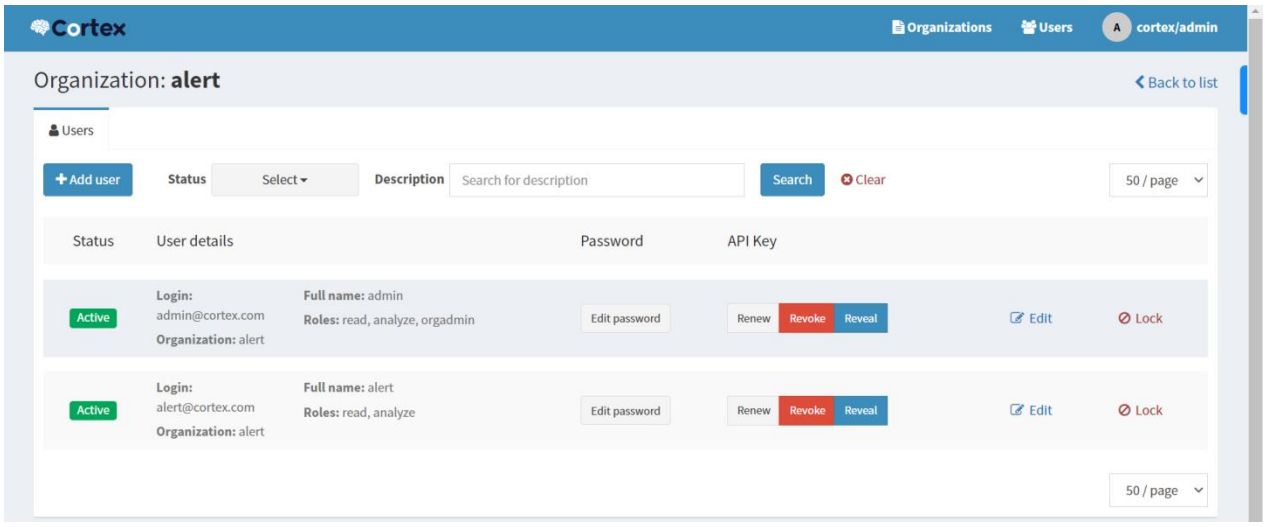
```
Installation script for Linux operating systems with DEB or RPM packages

Following install options are available:
  - Configure proxy settings
  - Install TheHive 5.x
  - Install Cortex (running Analyzers and Responders with Docker)
  - Install Cortex (running Analyzers and Responders on the host -- Not recommended, supported on Ubuntu and Debian ONLY)


This script has sucessfully been tested on freshly installed Operating Systems:
  - Fedora 35
  - RHEL 8.5
  - Ubuntu 20.04
  - Debian 11

Requirements:
  - 4vCPU
  - 16 GB of RAM

Usage:

   $ wget -q -O /tmp/install.sh https://archives.strangebee.com/scripts/install.sh ; sudo -v ; bash /tmp/install.sh

Maintained by: ©StrangeBee - https://www.strangebee.com

---


1) Setup proxy settings
2) Install TheHive
3) Install Cortex (run Neurons with docker)
4) Install Cortex (run Neurons locally)
5) Quit
Select an option: _
```



Let’s get in with local admin and create new organization and new users. One organization admin and one analyst user.
It's completely the the same process like how we did in The Hive. So I don’t want to show that again. So I created organization named alert.



And I created 2 users. And as you can see created API keys as well.



Let’s add scanners and responders.
To run docker images of Analyzers & Responders, Cortex should have permissions to use docker.

sudo usermod -G docker cortex

Some system packages are required to run Analyzers&Responders programs successfully:

sudo apt install -y --no-install-recommends python3-pip python3-dev ssdeep libfuzzy-dev libfuzzy2 libimage-exiftool-perl libmagic1 build-essential git libssl-dev

You may need to install Python's setuptools and update pip/pip3:

sudo pip3 install -U pip setuptools

Once finished, clone the Cortex-analyzers repository in the directory of your choosing:

cd /opt
git clone https://github.com/TheHive-Project/Cortex-Analyzers
chown -R cortex:cortex /opt/Cortex-Analyzers

```
root@cortex:/opt/Cortex-Analyzers# ls
AUTHORS       COMPONENTS  README.md    analyzers           docs    responders          utils
CHANGELOG.md  LICENSE     SECURITY.md  code_of_conduct.md  images  thehive-templates
```

Each analyzer comes with its own, pip compatible requirements.txt file. You can install all requirements with the following commands:

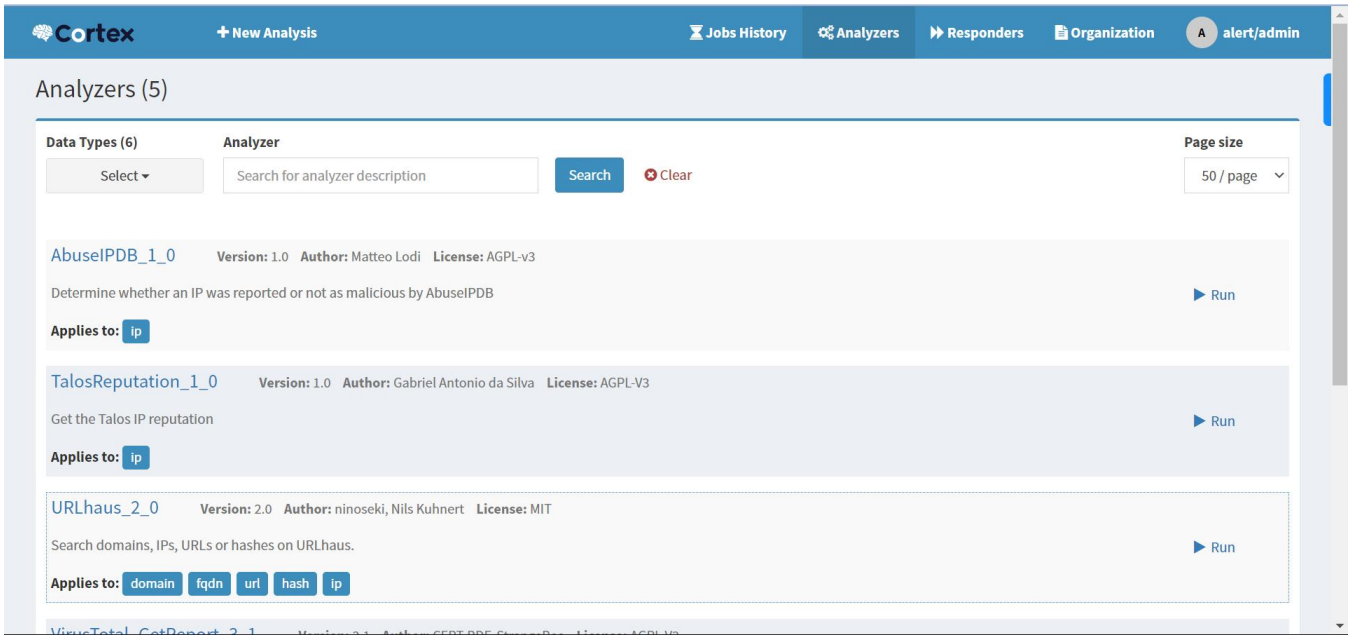
cd /opt
for I in $(find Cortex-Analyzers -name 'requirements.txt'); do sudo -H pip3 install -r $I || true; done

Installing dependencies of analyzers will take same time. You can go and have some coffe ☺.
Next, you'll need to tell Cortex where to find the analyzers. Analyzers may be in different directories as shown in this dummy example of the Cortex configuration file (application.conf):

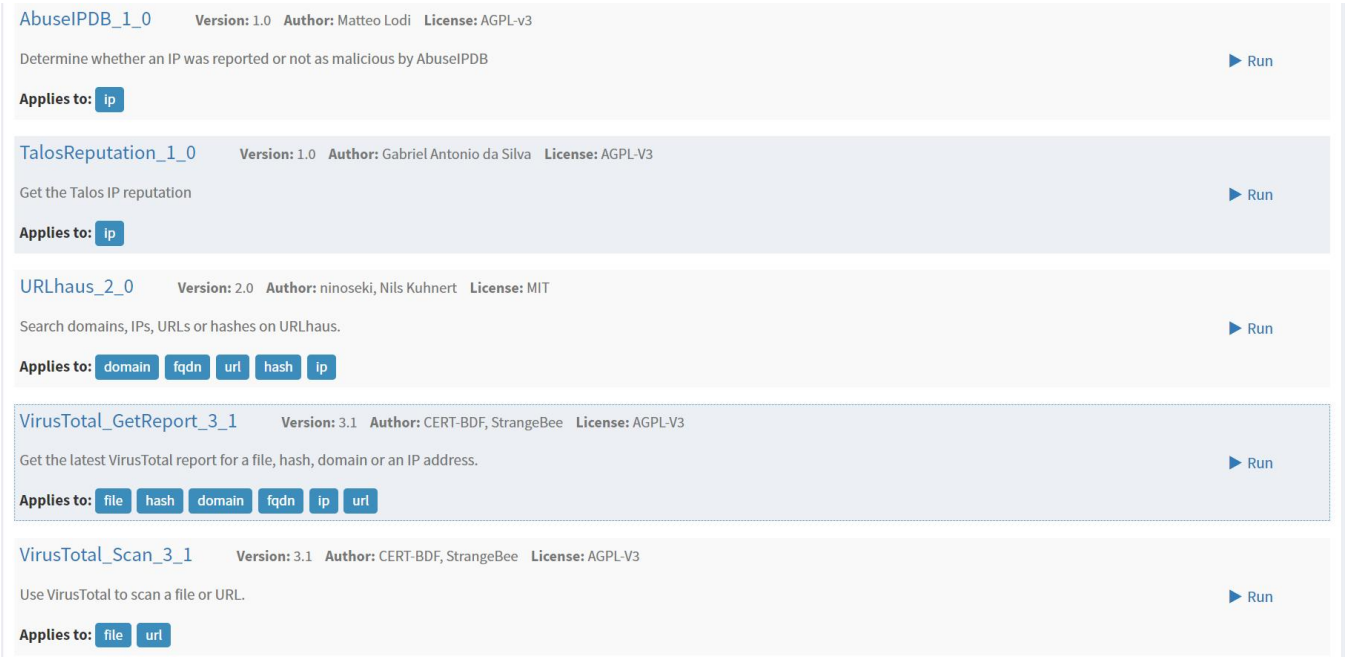
vim /etc/cortex/application.conf

```
[..]
analyzer {
  # Directory that holds analyzers
  urls = [
    "/opt/Cortex-Analyzers/analyzers",
  ]

  fork-join-executor {
    # Min number of threads available for analyze
    parallelism-min = 2
    # Parallelism (threads) ... ceil(available processors * factor)
    parallelism-factor = 2.0
    # Max number of threads available for analyze
    parallelism-max = 4
  }
}

responder {
  # Directory that holds responders
  urls = [
    "/opt/Cortex-Analyzers/responders"
  ]

  fork-join-executor {
    # Min number of threads available for analyze
    parallelism-min = 2
    # Parallelism (threads) ... ceil(available processors * factor)
    parallelism-factor = 2.0
    # Max number of threads available for analyze
    parallelism-max = 4
  }
}
[..]
```

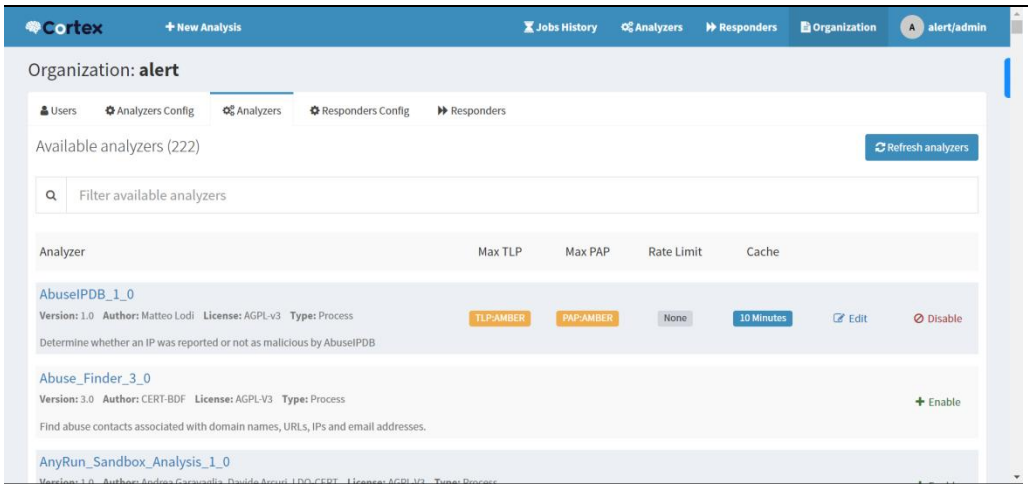
We come back to the Web interface and log in with organization admin. Then we click analyzers button at the top and you will be able to see all the scanners which is enabled.



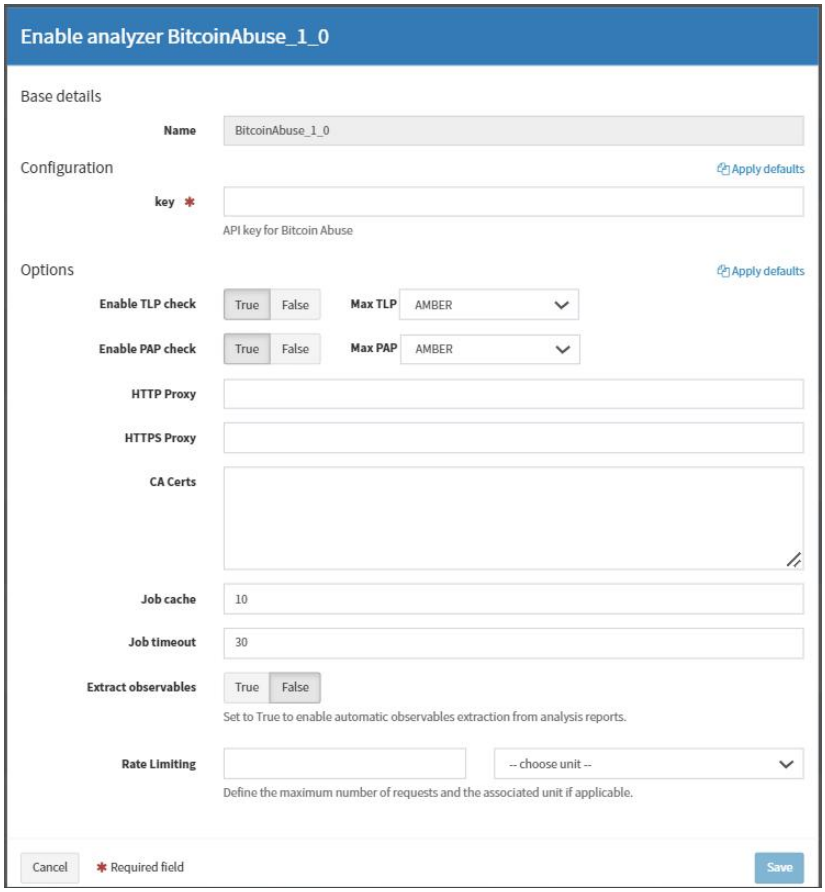
The 4 down below are the ones I have enabled and use. You only need to have a API key for some of them. But fortunately some of them doesn't require to have API key.



Let's enable some more. We need to click organization button and then click analyzers. So we can enable what ever we want by simply clicking enable and save.



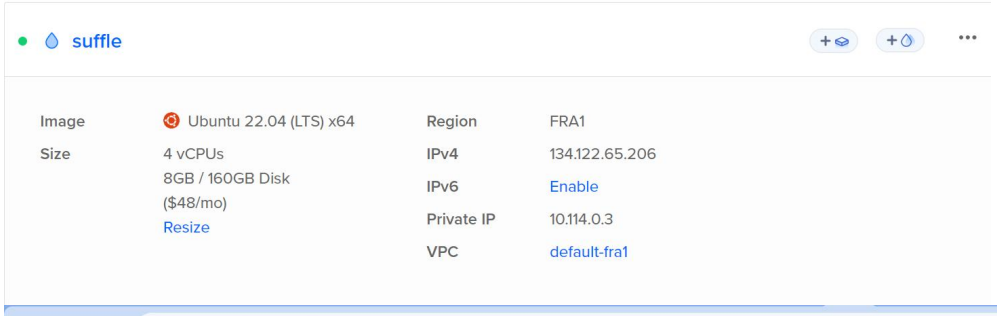
For example this is the one which doesn't require any API key. You need only click save button. The ones which requires API keys differs from this one only with its API key section.

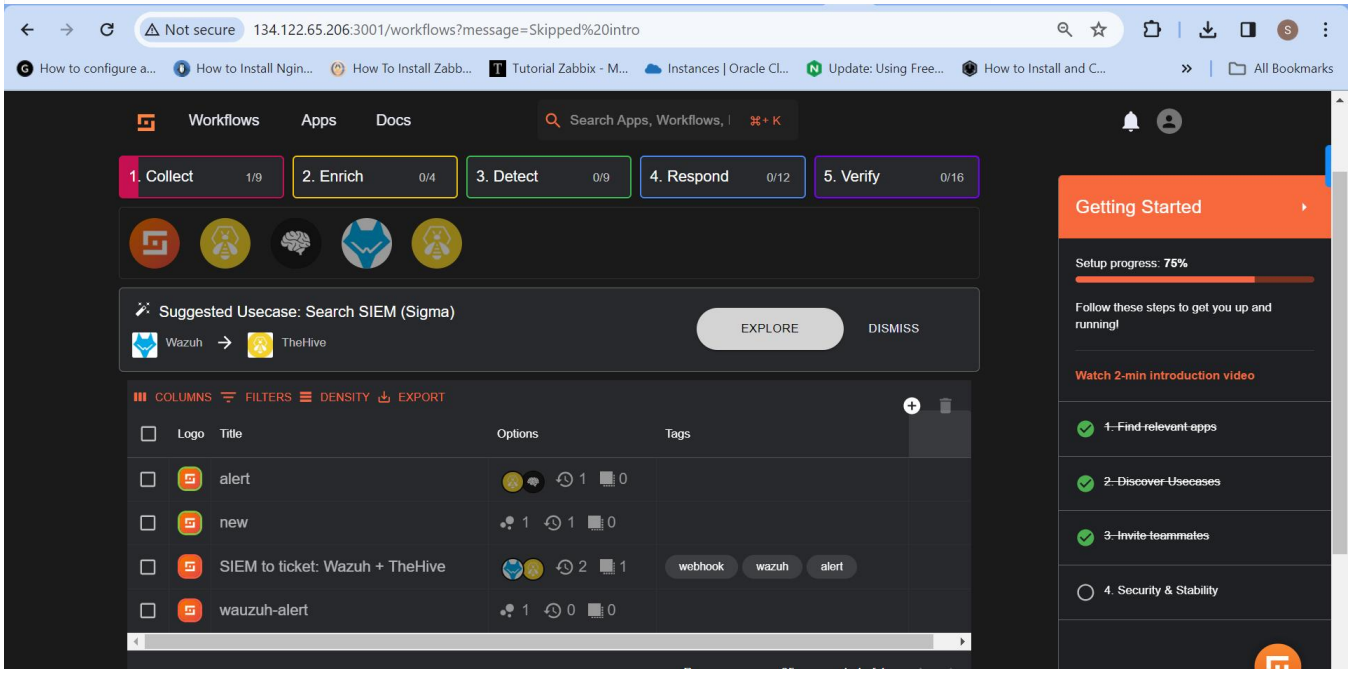


That is all we need to do so far with cortex.

Shuffle deployment and creating custom workflow

We raised up Ubuntu 22.04 on Digital Ocean platform for the deployment of Shuffle automation platform. You can accesses machine via SSH by using ip address and password. Password: !123Salam
Url: http://134.122.65.206:3001
Username: admin@code.edu.az Password: Salam123!

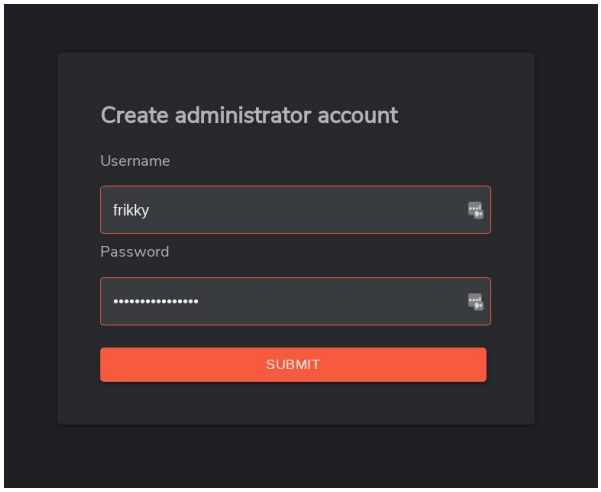




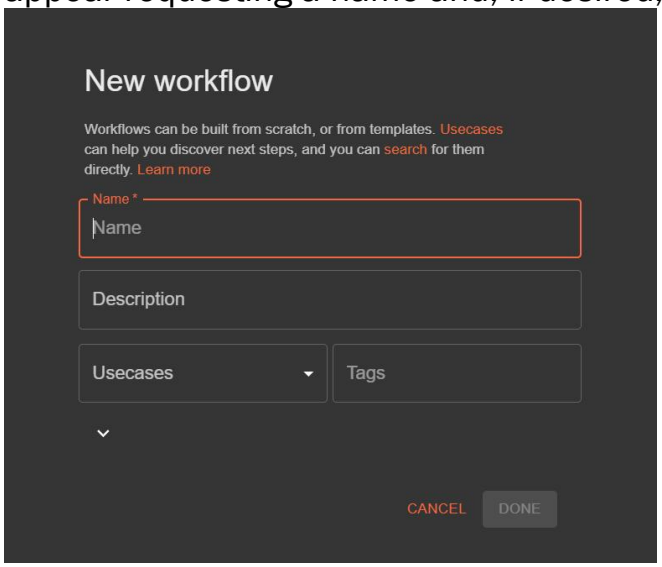
With everything Shuffle-related being on Github and running in Docker, Docker, Docker-compose and git to get started. With that out of the way, let's delve into the setup itself. The following commands will clone the repository, enter it, before running the docker containers specified in docker-compose.yml.

```
git clone https://github.com/frikky/Shuffle
cd Shuffle
docker-compose up –d
```

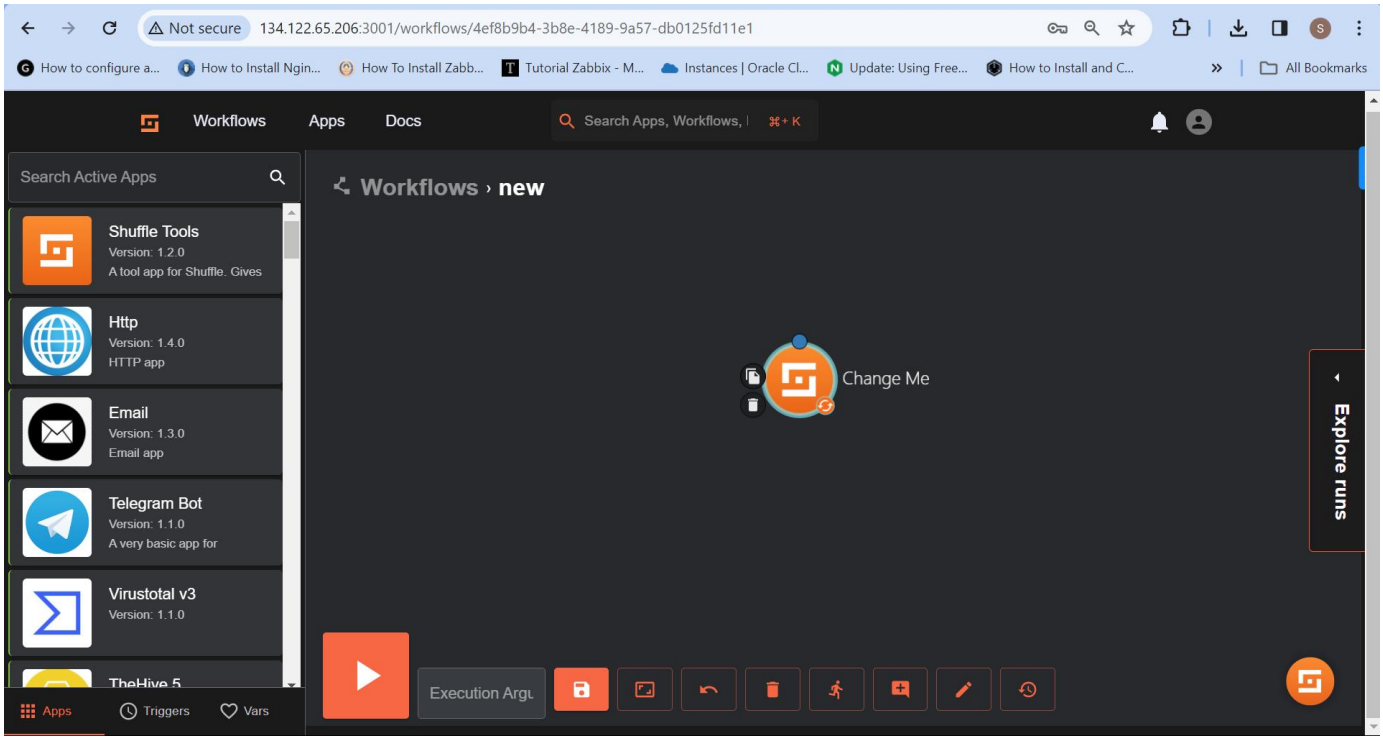
We are now on the first and only step of the admin setup: creating the first admin user.



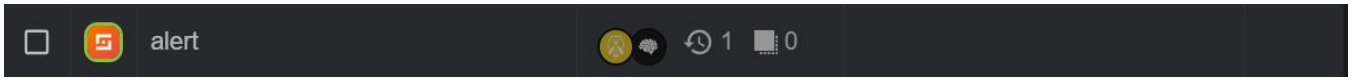
There’s no way to register users from outside the platform itself other than here. If you forget before your first login, delete the database (/etc/shuffle) and restart the backend.
After registering an admin account, we are redirected to the login screen, expecting the same credentials.
After logging in, we’re presented with a welcome screen. It has links to everything necessary to get cooking.
Workflows are where everything comes together in Shuffle, as was mentioned in the first installment of the series. Click the "New workflow" button to begin. A window will appear requesting a name and, if desired, a description.



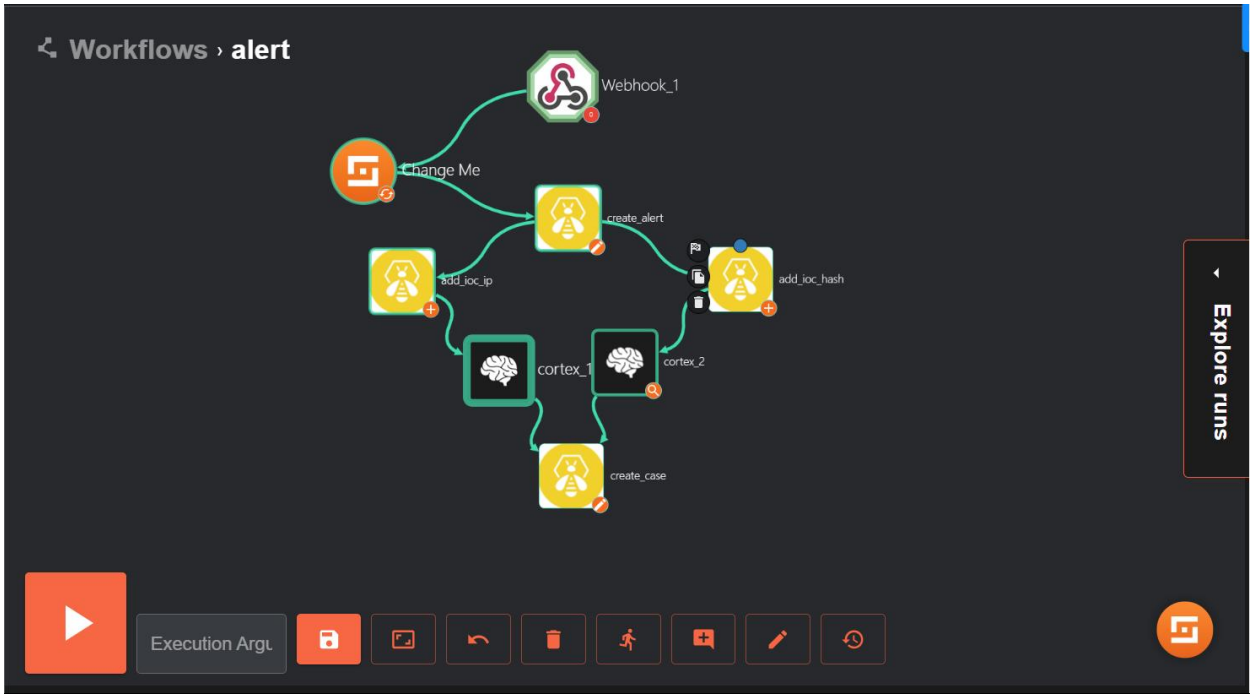
We are now on the workflow view after submitting. There are numerous apps on the left side of this empty display. Wait a few minutes and try refreshing the window if there are no apps listed. When Shuffle is run for the first time, several programs need to be loaded and developed, which could take some time.



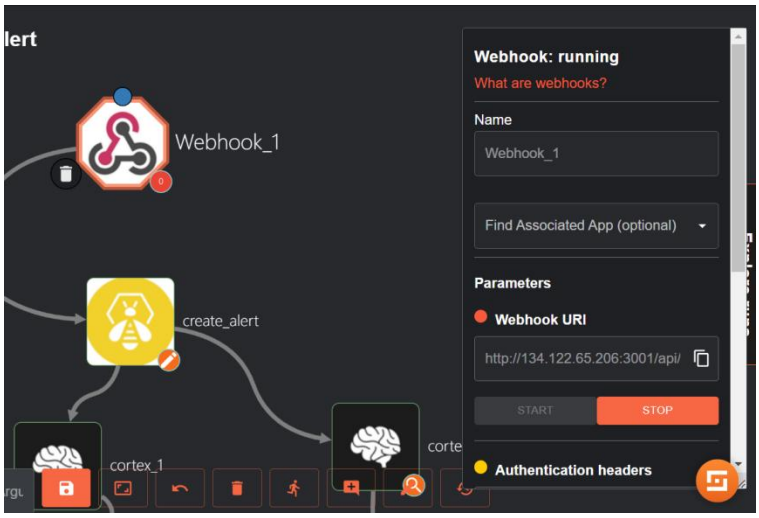
So from here I will show you already made workflow by us. And I will break it down for you. The name of the workflow is Alert. It consist of workflow among Wazuh, The Hive and Cortex.



This is the workflow of automated creating case in The Hive platform by using generated alerts from Wazuh and checking their IOCs in Cortex.
Let's delve into whole process.

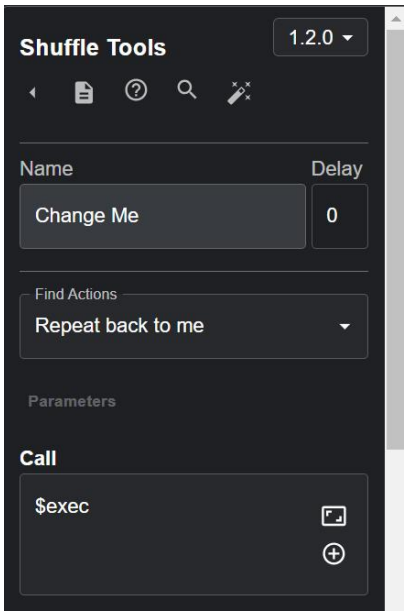


At first we need to send our Wazuh alerts to the shuffle. We come to the triggers and add Webhook. Then we take the API uri and we add the integration code block to the osse.conf file of the Wazuh.

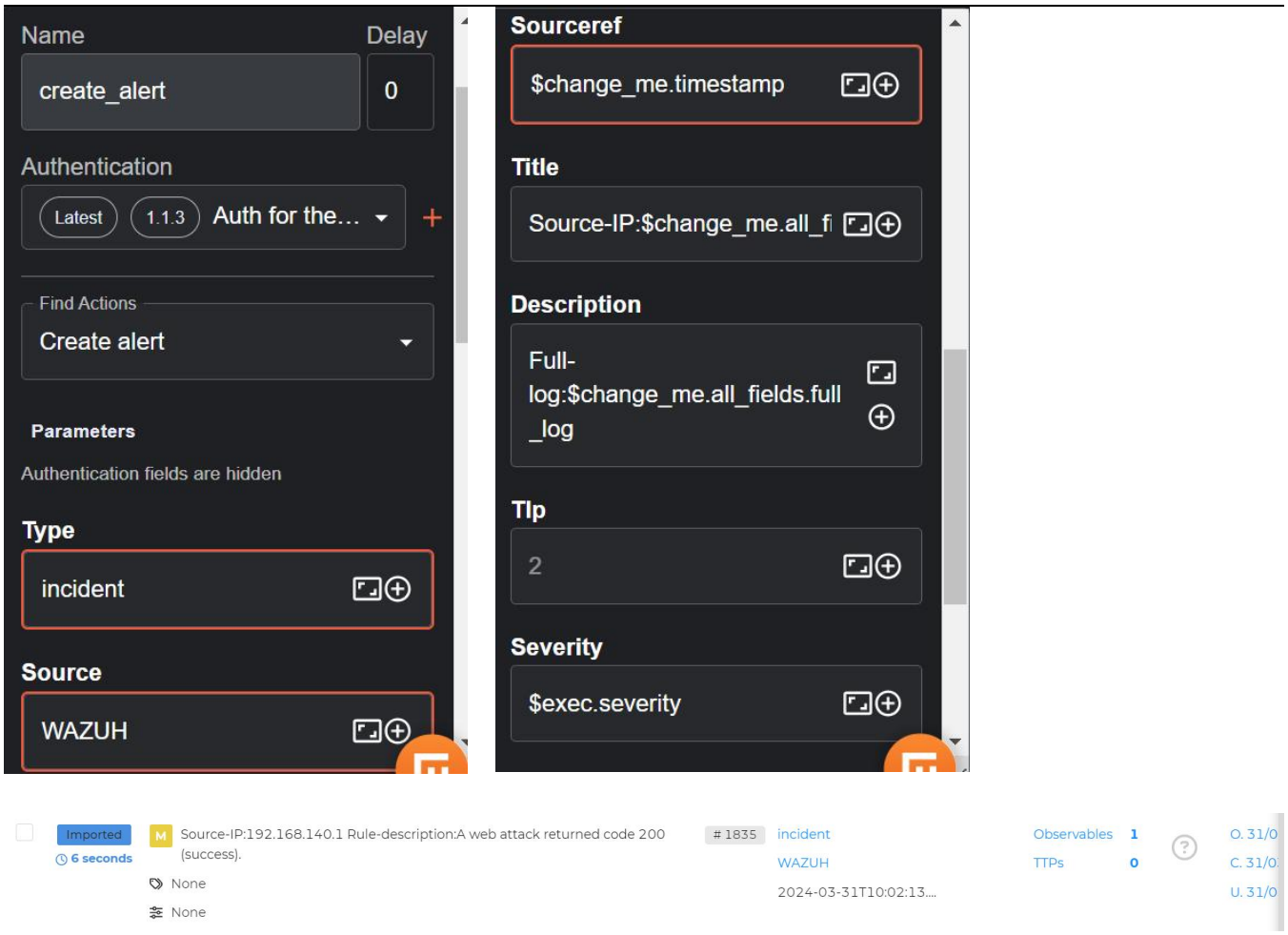


```
<integration>
  <name>shuffle</name>
  <hook_url>http://134.122.65.206:3001/api/v1/hooks/webhook_82089c67-0c5c-4c18-b602-9e6969f00d3b</hook_url>
  <level>6</level>
  <alert_format>json</alert_format>
</integration>
```

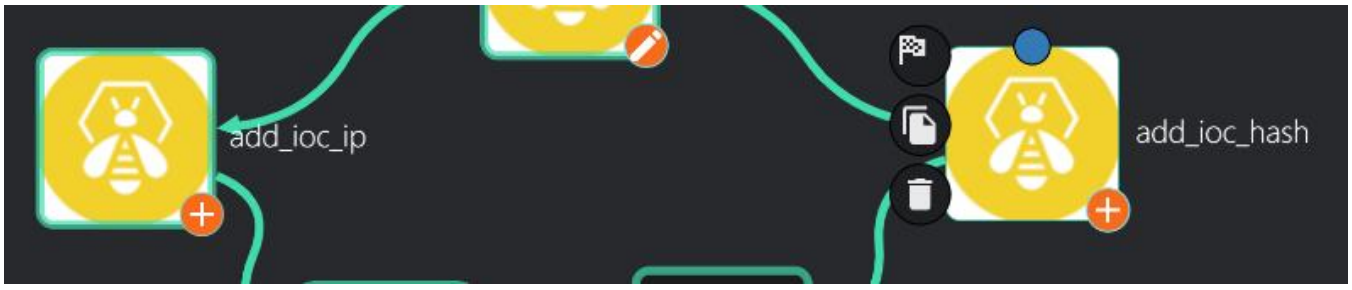
That is it. Let's change the action of the "change me" node. We need to change the action to "repeat back to me" in order to see the alerts in real time in the run window.



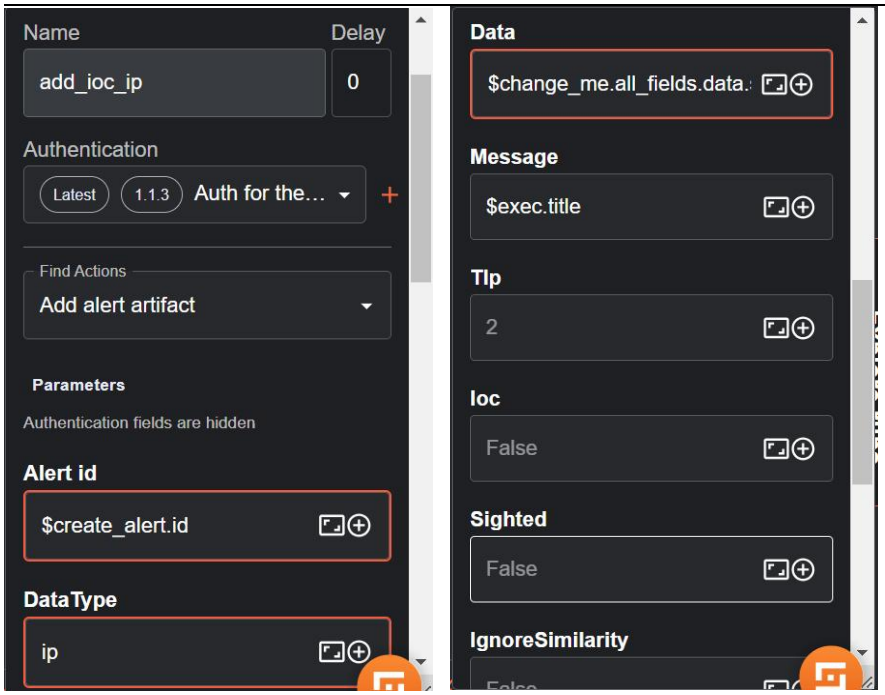
It's time to create Alerts in The Hive. It's pretty easy you should fist add authentication, it requires API key and URL. After that we change action option to the "create alert" and add our parametrs. We must add "Type, Source, Sourceref". If you don't add these ones it won't run. But the rest of them "Title, Description, Severity, Tip" are optional. However I recommend you add title and Description. You can add necessary data to the Title and Description like I did. You will see I have added source ip address and the rule description to the title and full log to the description.



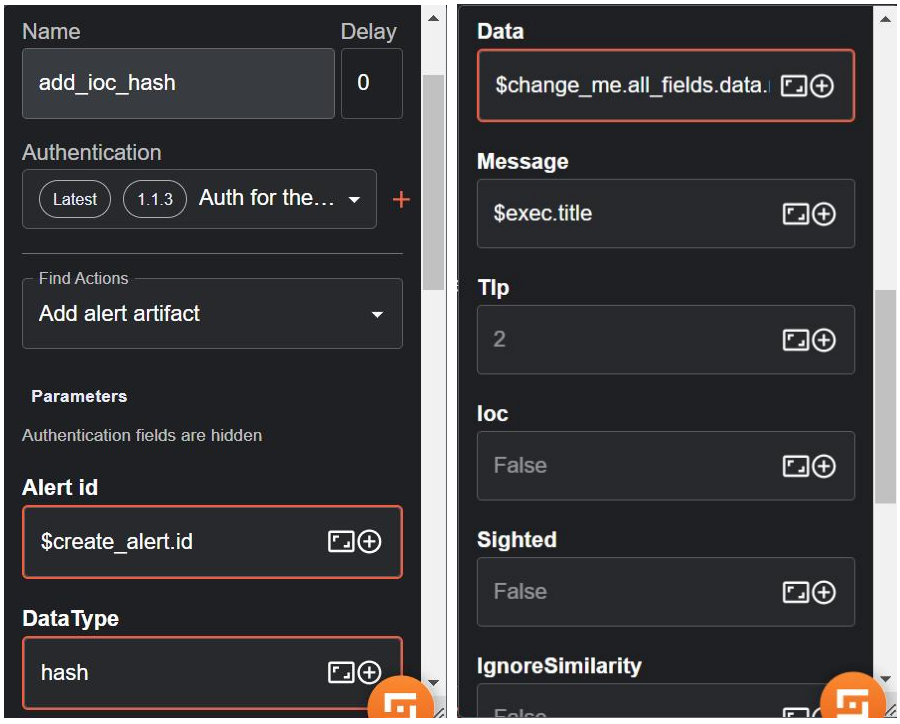
Then let's add 2 node which will add observables to our alerts. One of them will be IP and another one will be hash. We will be using of The Hive node itself for this.



We double click one of them and add necessary parameters. First we need to change action to "add artifacts". Then We have to add alert id. It have to be the id from create_alert node. Then we have to add Data_type and Data.

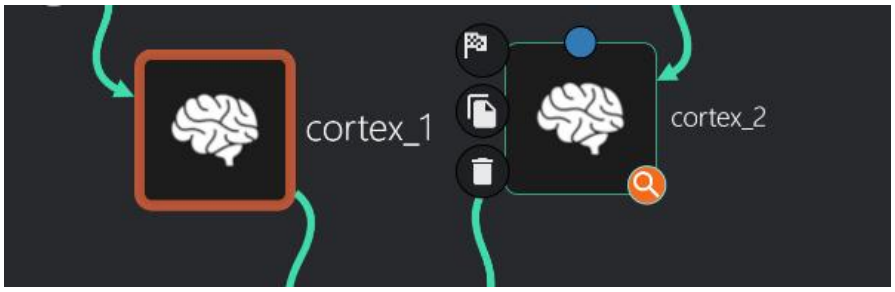


The pictures in the above are the content of the add_ioc_ip node.

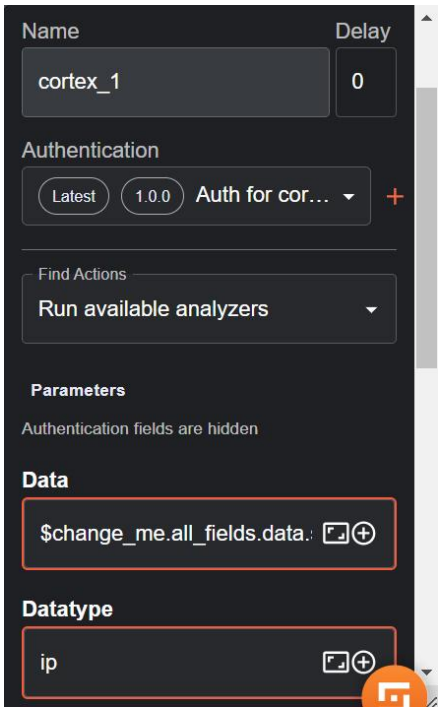


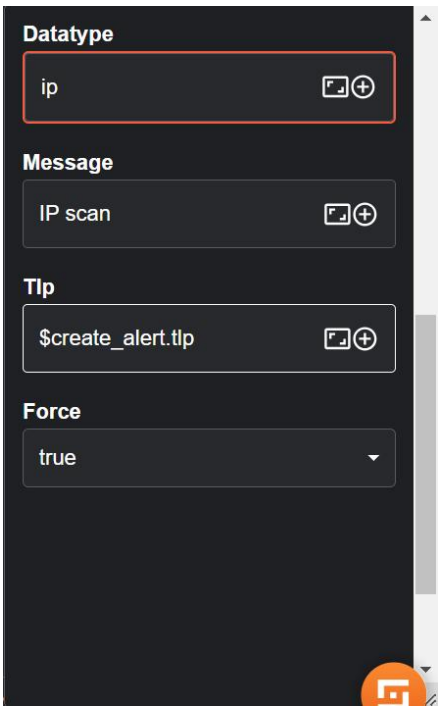
And these ones are the content of the add_ios_hash node. Only Data differs from first one.

It's time to add Cortex nodes. I have added 2 Cortex nodes. Cortex_1 is for checking IP addresses and Cortex_2 is for checking hashes.



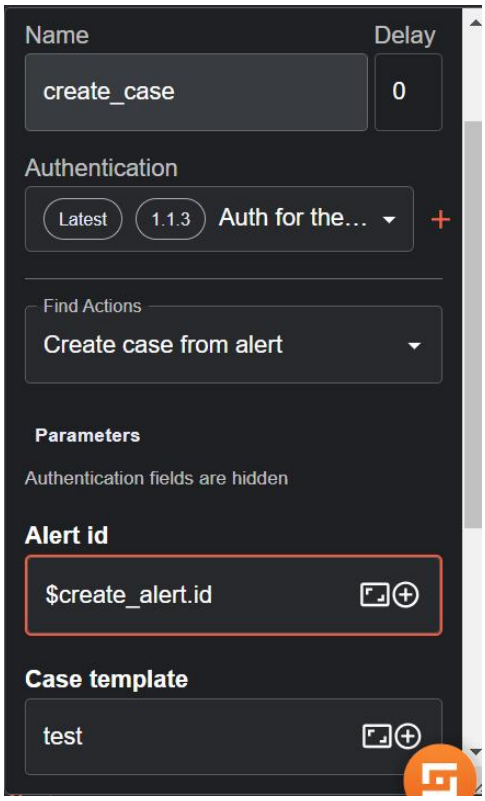
First of all we have to add authentication parameter. It only requires URL and API key. Then we change action to the "run available analyzers". After that we should add other necessary parameters such as Data, Datatype, Message and Tip.



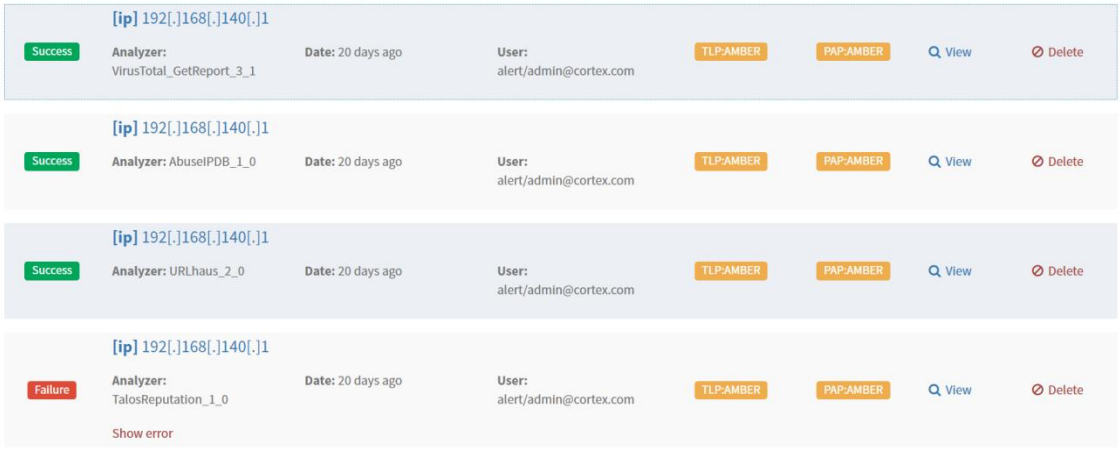


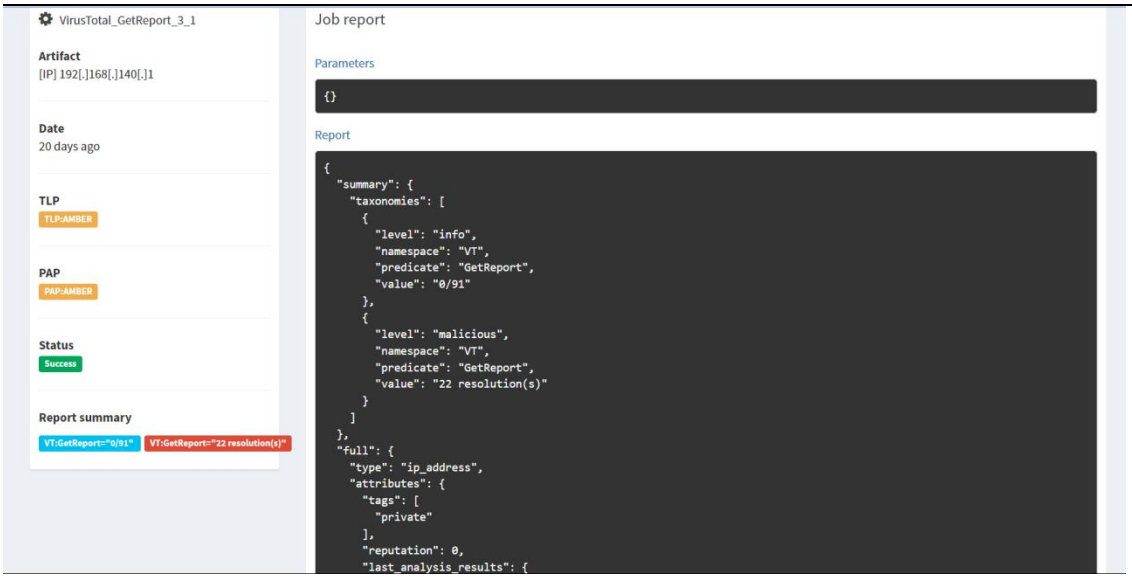
The another cortex node is for checking hashes. Parameters completely the same with this one only Data and Datatype differs from this one.

The last node is for creating case in The Hive. We should add new The Hive node and change action to the “create case from alert”. Then we have to add “alert id” and “case tamplate” name which we already created before.

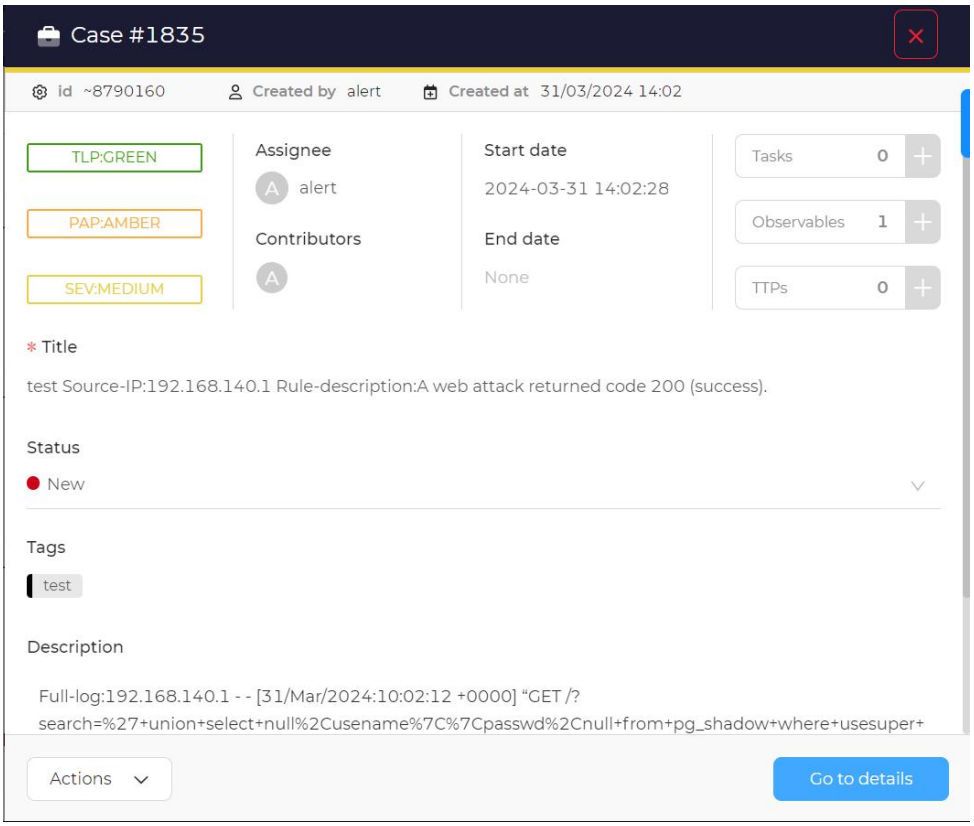


It’s time to see the result.

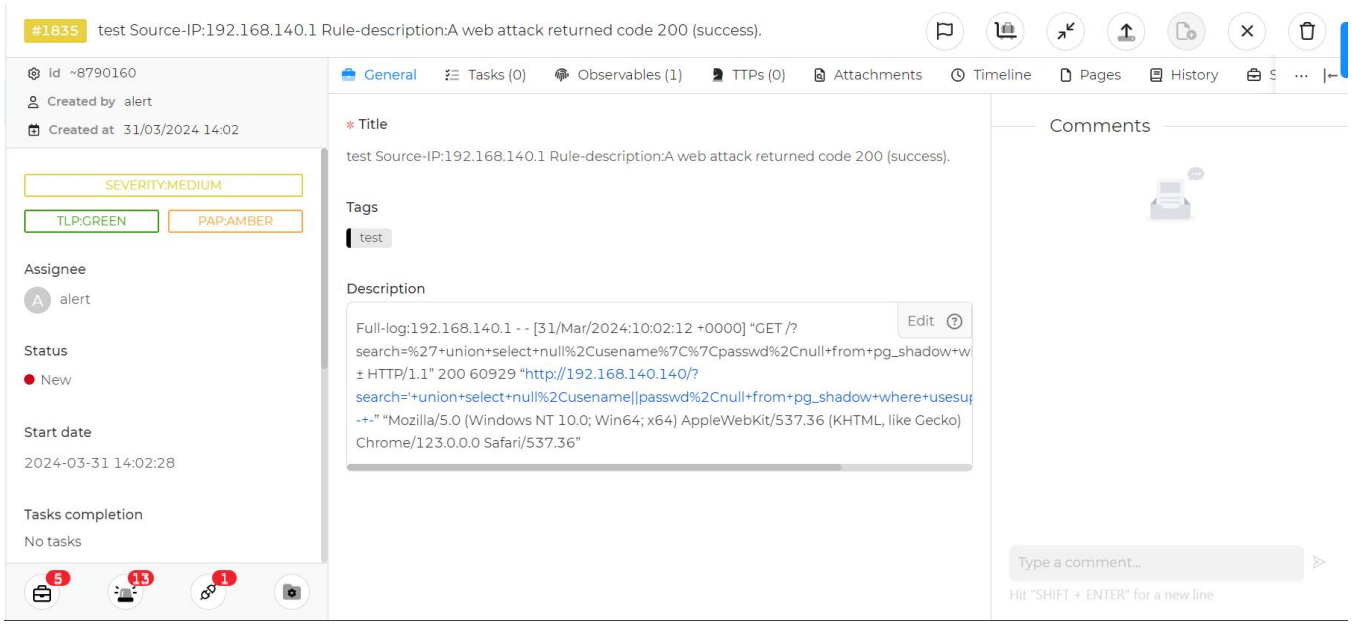



VirusTotal_GetReport_3_1
Artifact
[IP] 192[.]168[.]140[.]1
Date
20 days ago
TLP
TLP:AMBER
PAP
PAP:AMBER
Status
Success
Report summary
VT:GetReport="0/91"
VT:GetReport="22 resolution(s)"
Job report
Parameters
{}
Report
{
"summary": {
"taxonomies": [
{
"level": "info",
"namespace": "VT",
"predicate": "GetReport",
"value": "0/91"
},
{
"level": "malicious",
"namespace": "VT",
"predicate": "GetReport",
"value": "22 resolution(s)"
}
]
},
"full": {
"type": "ip_address",
"attributes": {
"tags": [
"private"
],
"reputation": 0,
"last_analysis_results": {


STATUS
SEVERITY
#NUMBER
TITLE
DETAILS
ASSIGNEE
DATES
S.
C.
U.
152
New
9 hours
#1835 - test Source-IP:192.168.140.1 Rule-description:A web attack returned code 200 (success).
test
None
Tasks 0
Observables 1
TTPs 0
Linked Alerts 1
S. 31/03/2024 14:02
C. 31/03/2024 14:02

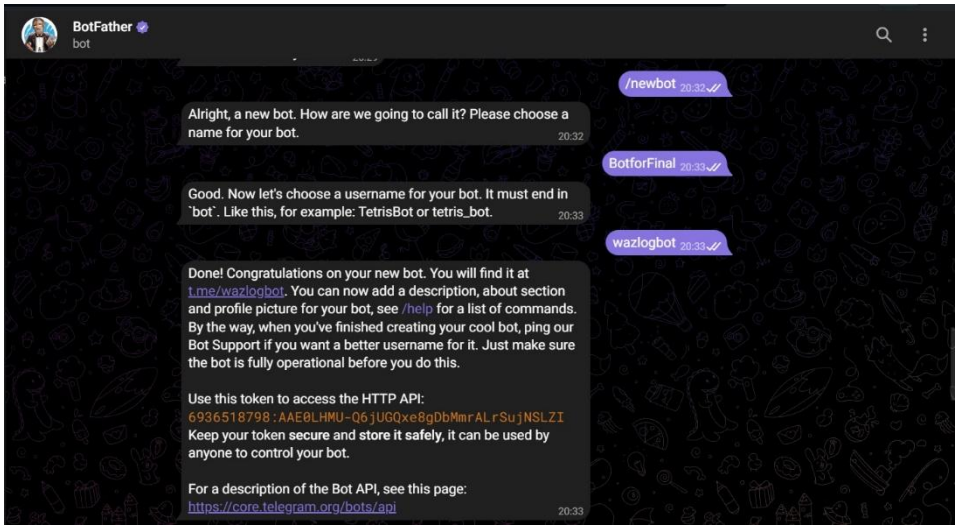

Case #1835
id ~8790160
Created by alert
Created at 31/03/2024 14:02
TLP:GREEN
PAP:AMBER
SEV:MEDIUM
Assignee
alert
Contributors
Start date
2024-03-31 14:02:28
End date
None
Tasks 0
Observables 1
TTPs 0
* Title
test Source-IP:192.168.140.1 Rule-description:A web attack returned code 200 (success).
Status
New
Tags
test
Description
Full-log:192.168.140.1 - - [31/Mar/2024:10:02:12 +0000] "GET /?
search=%27+union+select+null%2Cusename%7C%7Cpasswd%2Cnull+from+pg_shadow+where+usesuper+
Actions
Go to details

Detailed:

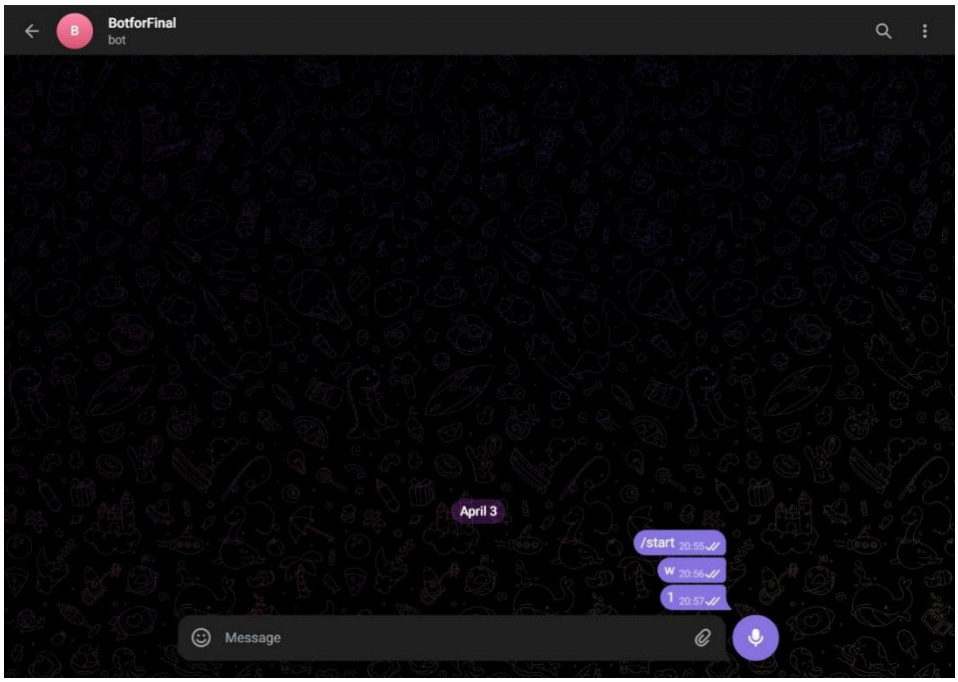

#1835
test Source-IP:192.168.140.1 Rule-description:A web attack returned code 200 (success).
Id ~8790160
Created by alert
Created at 31/03/2024 14:02
SEVERITY:MEDIUM
TLP:GREEN
PAP:AMBER
Assignee
alert
Status
New
Start date
2024-03-31 14:02:28
Tasks completion
No tasks
General
Tasks (0)
Observables (1)
TTPs (0)
Attachments
Timeline
Pages
History
Title
test Source-IP:192.168.140.1 Rule-description:A web attack returned code 200 (success).
Tags
test
Description
Edit
Full-log:192.168.140.1 - - [31/Mar/2024:10:02:12 +0000] "GET /?
search=%27+union+select+null%2Cusename%7C%7Cpasswd%2Cnull+from+pg_shadow+w
± HTTP/1.1" 200 60929 "http://192.168.140.140/?
search='+union+select+null%2Cusename||passwd%2Cnull+from+pg_shadow+where+usesu
-+-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/123.0.0.0 Safari/537.36"
Comments
Type a comment...
Hit "SHIFT + ENTER" for a new line

Wazuh Telegram Integration

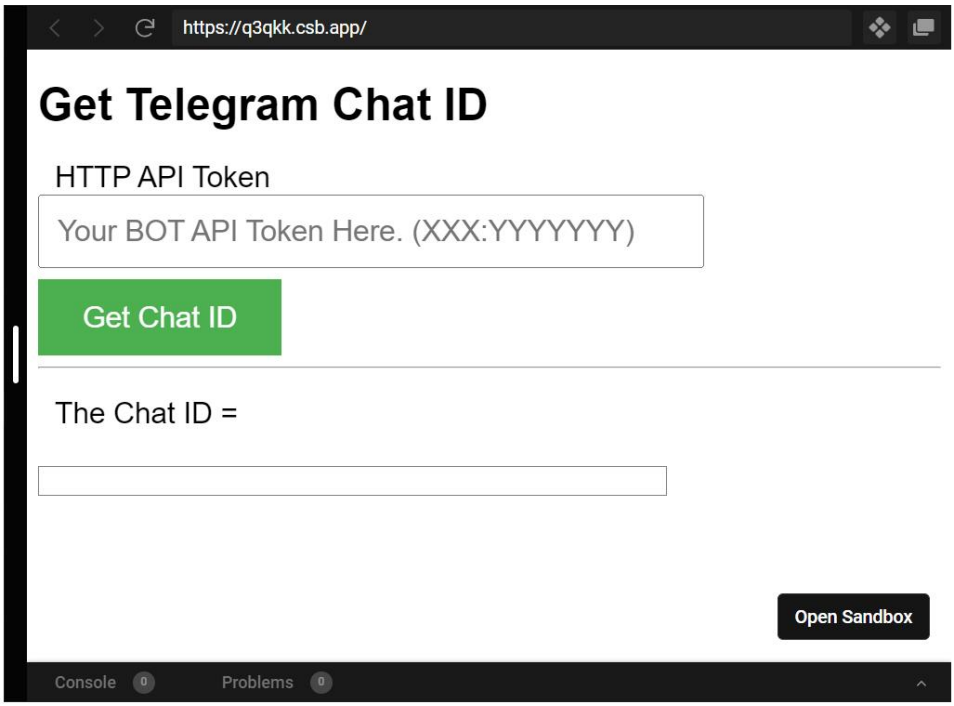
To send Wazuh alerts to a Telegram chat, we need to create a bot first. To do this we have to send a couple of messages to @BotFather. After starting the bot with the /start command, we have to send the /newbot command to start creating the bot, and we will choose the name of the bot, BotForFinal in this case :))



Clicked the link which given to me at the top of the message. I made the new group and added my bot to that group as well. When wrote simple word.



Let’s get a chat id. It is pretty simple. I will give the link to you (https://sean-bradley.medium.com/get-telegram-chat-id-80b575520659). We can copy our token then paste to the place which you going to see in the link that I gave to you.



Take the chat id from this website. We will add it to the config file later. Let’s start to configuration.

We used demos repisotery for integrating Wazuh with telegram bot. There are 3 sections in the repo. First one one is curstom-telegram script it used for trigering 2nd one (custum-telegram.py). But we used our script instead of custom-telegram.py . The 3rd one is a integration code block which we need to add into the ossec.conf file.

Let’s see the real content of these files.

vim /var/ossec/integrations/custom-telegram

```
#!/bin/sh
WPYTHON_BIN="framework/python/bin/python3"
SCRIPT_PATH_NAME="$0"
DIR_NAME="$(cd $(dirname ${SCRIPT_PATH_NAME}); pwd -P)"
SCRIPT_NAME="$(basename ${SCRIPT_PATH_NAME})"
case ${DIR_NAME} in
   */active-response/bin | */wodles*)
      if [ -z "${WAZUH_PATH}" ]; then
```

```
        WAZUH_PATH="$(cd ${DIR_NAME}/../..; pwd)"
      fi
      PYTHON_SCRIPT="${DIR_NAME}/${SCRIPT_NAME}.py"
   ;;
   */bin)
      if [ -z "${WAZUH_PATH}" ]; then
         WAZUH_PATH="$(cd ${DIR_NAME}/..; pwd)"
      fi
      PYTHON_SCRIPT="${WAZUH_PATH}/framework/scripts/${SCRIPT_NAME}.py"
   ;;
    */integrations)
      if [ -z "${WAZUH_PATH}" ]; then
         WAZUH_PATH="$(cd ${DIR_NAME}/..; pwd)"
      fi
      PYTHON_SCRIPT="${DIR_NAME}/${SCRIPT_NAME}.py"
   ;;
esac
${WAZUH_PATH}/${WPYTHON_BIN} ${PYTHON_SCRIPT} "$@"
```

vim /var/ossec/integrations/custom-telegram.py

```
#!/usr/bin/env python3
import sys
import json
import requests
from requests.auth import HTTPBasicAuth
CHAT_ID = "-4060206950"
alert_file = open(sys.argv[1])
hook_url = sys.argv[3]
alert_json = json.loads(alert_file.read())
alert_file.close()

# Extract data fields

date = alert_json['timestamp'] if 'timestamp' in alert_json else "N/A"

agent = alert_json['agent']['name'] if 'name' in alert_json['agent'] else "N/A"

srcip=alert_json['agent']['ip'] if 'ip' in alert_json['agent'] else "N/A"

payload = alert_json['data']['url'] if 'url' in alert_json['data'] else "N/A"

alert_level = alert_json['rule']['level'] if 'level' in alert_json['rule'] else "N/A"

description = alert_json['rule']['description'] if 'description' in alert_json['rule'] else "N/A"
```

```
# Generate request

msg_data = {}

msg_data['chat_id'] = CHAT_ID

msg_data['text'] = f"Date: {date}\nAgent Name: {agent}\nSource IP: {srcip}\nPayload: {payload}\nAlert Level: {alert_level}\nDescription: {description}"

headers = {'content-type': 'application/json', 'Accept-Charset': 'UTF-8'}

# Send the request
requests.post(hook_url, headers=headers, data=json.dumps(msg_data))
sys.exit(0)
```

```
chown root:wazuh /var/ossec/integrations/custom-telegram*

chmod 750 /var/ossec/integrations/custom-telegram*
```

vim /var/ossec/etc/ossec.conf

```
<integration>
      <name>custom-telegram</name>
      <level>3</level>
      <hook_url>https://api.telegram.org/bot*YOUR API KEY*/sendMessage</hook_url>
      <alert_format>json</alert_format>
   </integration>
```

At last we restart our manager

Here is the result.