



Wazuh installation

Student: Yagub Hajiyev



Table of content

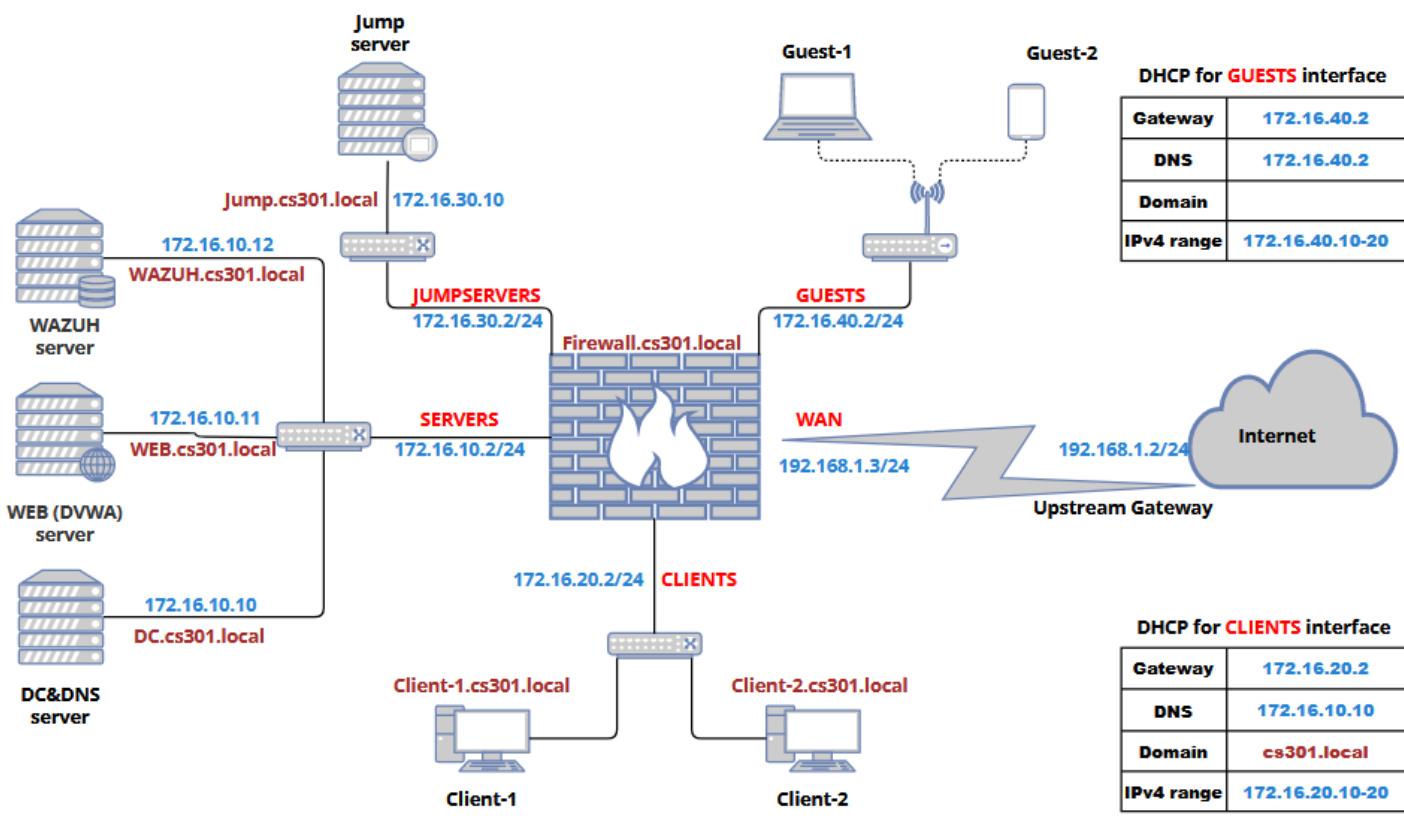
Summary	3
1. Wazuh installation	4
2. Joining agents	7
2.1 Windows agent joining (DC)	7
2.2 Firewall joining (OPNsense)	9
2.3 Linux agent joining (WEB)	11
3. Testing agents	14
3.1 Testing DC	14
3.2 Testing OPNsense	15
4. Wazuh Proof of Concept	17
4.1 Blocking a known malicious actor	17
4.2 Responding to network attacks with Suricata and Wazuh XDR	20
4.3 Detecting malware using YARA integration	24
4.4 Detect threats on Windows by monitoring Sysmon events	30



Summary

Our infrastructure consists of several networks and they are protected by network security solution, firewall. All endpoint devices generate security logs in their local environment and these logs have to be monitored to detect malicious activities. At this point, **SIEM** (Security Information Events Management) solutions help us to collect all of these logs from those devices and analyse them at the centralized location.

We will implement **Wazuh** open source **SIEM** solution to control all system logs from connected devices. You can find much more detailed information about this tool on its official webpage, but as a short explanation **Wazuh** has a central service called **Wazuh Manager** and log collector service which is called **Wazuh Agent**. In our infrastructure, we need to add new server for **Wazuh** and assign it a static IP from **SERVERS** network. In following capture, we will deploy **Wazuh** in this machine and later install agent services into our other devices one by one for joining them to central **Wazuh** server. After the setup, we will conduct several tasks on our infrastructure to learn different security methods and discover **Wazuh** capabilities.



Adapter name	Interface name	Subnet IP	Host IP	Firewall IP
NAT	WAN	192.168.1.0/24	192.168.1.1	192.168.1.3
VMNET1	SERVERS	172.16.10.0/24	172.16.10.1	172.16.10.2
VMNET2	CLIENTS	172.16.20.0/24	172.16.20.1	172.16.20.2
VMNET3	JUMPSERVERS	172.16.30.0/24	172.16.30.1	172.16.30.2
VMNET4	GUESTS	172.16.40.0/24	172.16.40.1	172.16.40.2



Wazuh installation

In order to install **Wazuh** all we need to download [installation script](#) and run it with **root** privileges.

```
[root@wazuh ~]# curl -s0 https://packages.wazuh.com/4.7/wazuh-install.sh
[root@wazuh ~]# ll
total 172
-rw----- 1 root root 1034 Mar 31 08:30 anaconda-ks.cfg
-rw-r--r-- 1 root root 1261 Mar 31 08:49 initial-setup-ks.cfg
-rw-r--r-- 1 root root 165733 Apr 23 02:00 wazuh-install.sh
[root@wazuh ~]#
```

I installed **Wazuh** into **Oracle Linux** machine with 3Gb RAM and 2 core CPUs, but as default **Wazuh** doesn't let to setup in such a weak system and gives errors. We can overcome it with the help of "**-i**" (ignore) option as displayed in the figure. Once we run the script it install all configuration files and services automatically. At the end of the installation **Wazuh** create a default user **admin** and its random password for us. This password is difficult to memorize, so we can replace it with our custom defined password.

```
[root@wazuh ~]# bash wazuh-install.sh -a -i
23/04/2024 02:01:26 INFO: Starting Wazuh installation assistant. Wazuh version: 4.7.3
23/04/2024 02:01:26 INFO: Verbose logging redirected to /var/log/wazuh-install.log
23/04/2024 02:01:41 WARNING: Hardware and system checks ignored.
23/04/2024 02:01:41 INFO: Wazuh web interface port will be 443.
23/04/2024 02:01:44 WARNING: The system has Firewalld enabled. Please ensure that tra...
23/04/2024 02:01:48 INFO: Wazuh repository added.
23/04/2024 02:01:48 INFO: --- Configuration files ---
23/04/2024 02:01:48 INFO: Generating configuration files.
23/04/2024 02:01:50 INFO: Created wazuh-install-files.tar. It contains the Wazuh clus...
23/04/2024 02:01:50 INFO: --- Wazuh indexer ---
23/04/2024 02:01:50 INFO: Starting Wazuh indexer installation.
23/04/2024 02:07:32 INFO: Wazuh indexer installation finished.
23/04/2024 02:07:33 INFO: Wazuh indexer post-install configuration finished.
23/04/2024 02:07:33 INFO: Starting service wazuh-indexer.
23/04/2024 02:08:49 INFO: wazuh-indexer service started.
23/04/2024 02:08:49 INFO: Initializing Wazuh indexer cluster security settings.
23/04/2024 02:09:15 INFO: Wazuh indexer cluster initialized.
23/04/2024 02:09:15 INFO: --- Wazuh server ---
23/04/2024 02:09:15 INFO: Starting the Wazuh manager installation.
23/04/2024 02:12:13 INFO: Wazuh manager installation finished.
23/04/2024 02:12:13 INFO: Starting service wazuh-manager.
23/04/2024 02:12:31 INFO: wazuh-manager service started.
23/04/2024 02:12:31 INFO: Starting Filebeat installation.
23/04/2024 02:12:56 INFO: Filebeat installation finished.
23/04/2024 02:13:02 INFO: Filebeat post-install configuration finished.
23/04/2024 02:13:02 INFO: Starting service filebeat.
23/04/2024 02:13:03 INFO: filebeat service started.
23/04/2024 02:13:03 INFO: --- Wazuh dashboard ---
23/04/2024 02:13:03 INFO: Starting Wazuh dashboard installation.
23/04/2024 02:16:29 INFO: Wazuh dashboard installation finished.
23/04/2024 02:16:29 INFO: Wazuh dashboard post-install configuration finished.
23/04/2024 02:16:29 INFO: Starting service wazuh-dashboard.
23/04/2024 02:16:29 INFO: wazuh-dashboard service started.
23/04/2024 02:17:44 INFO: Initializing Wazuh dashboard web application.
23/04/2024 02:17:45 INFO: Wazuh dashboard web application initialized.
23/04/2024 02:17:45 INFO: --- Summary ---
23/04/2024 02:17:45 INFO: You can access the web interface https://<wazuh-dashboard-ip>:443
    User: admin
    Password: O??aJ*K.P?+q9RKDyMKsnkEhJG3Xr4co
23/04/2024 02:17:45 INFO: Installation finished.
[root@wazuh ~]#
```

For changing the predefined password, we have to use another ready [script](#).

```
[root@wazuh ~]# curl -s0 wazuh-passwords-tool.sh https://packages.wazuh.com/4.7/wazuh-passwords-tool.sh
[root@wazuh ~]# ll wazuh-passwords-tool.sh
-rw-r--r-- 1 root root 39641 Apr 23 02:23 wazuh-passwords-tool.sh
[root@wazuh ~]#
```



After downloading the script into your **Wazuh** server, execute it with “**-u <username>**” option to define the user (**admin** in this situation) whose password you want to change and “**-p <new_password>**” for defining new password.

```
[root@wazuh ~]# bash wazuh-passwords-tool.sh -u admin -p Salam123*
23/04/2024 02:25:16 INFO: Generating password hash
23/04/2024 02:26:03 WARNING: Password changed. Remember to update the password in the
Wazuh dashboard and Filebeat nodes if necessary, and restart the services.
[root@wazuh ~]#
```

In the next step we must add **Wazuh** ports (**1514** for agent connection, **1515** for agent enrollment and **443** for **Wazuh** web user interface) into our firewall.

```
[root@wazuh ~]# firewall-cmd --list-ports
[root@wazuh ~]# firewall-cmd --add-port={1514/tcp,1515/tcp,443/tcp} --permanent
success
[root@wazuh ~]# firewall-cmd --reload
success
[root@wazuh ~]# firewall-cmd --list-ports
443/tcp 1514/tcp 1515/tcp
[root@wazuh ~]#
```

Now open your browser and go to the <https://<Wazuh server IP>> address.

The screenshot shows a Firefox browser window with the following details:

- Address Bar:** Not Secure https://172.16.10.12 90%
- Warning Message:** **Warning: Potential Security Risk Ahead**
- Description:** Firefox detected a potential security threat and did not continue to **172.16.10.12**. If you visit this site, attackers could try to steal information like your passwords, emails, or credit card details.
- What can you do about it?**
 - The issue is most likely with the website, and there is nothing you can do to resolve it.
 - If you are on a corporate network or using antivirus software, you can reach out to the support teams for assistance. You can also notify the website's administrator about the problem.
- Buttons:** Learn more... Go Back (Recommended) Advanced...
- Details Panel:**
 - Someone could be trying to impersonate the site and you should not continue.
 - Websites prove their identity via certificates. Firefox does not trust 172.16.10.12 because its certificate issuer is unknown, the certificate is self-signed, or the server is not sending the correct intermediate certificates.
 - Error code: [SEC_ERROR_UNKNOWN_ISSUER](#)
 - [View Certificate](#)
- Bottom Buttons:** Go Back (Recommended) Accept the Risk and Continue



Insert your credentials (username: **admin**, password: **Salam123*** for this lab) and enter to the **GUI** interface.



NOTE: if you face with the error displayed below, just be patient and restart all **Wazuh** services.

```
[root@wazuh ~]# systemctl restart wazuh-manager.service wazuh-indexer wazuh-dashboard.service
```

Once you open the **GUI**, the home page will meet you. This page contains some general information about our agents, shorcuts to monitoring and events windows etc. As you we doesn't have any connected agent yet, although the **Wazuh** server itself also has default agent in it to send machine logs into our **SIEM**. In the next capture we will add all machines into our **SIEM** as agents.



Joining agents

2.1 Windows agent joining (DC)

As you know we can integrate all systems into **Wazuh** as agents in our infrastructure. Lets begin with **DC** machine. In the home page press “**Add agent**” button, and it will carry you into agent deploy window. In the first step select **Windows** platform, next type you **Wazuh** server’s **IP** address, in the third step leave it empty (then **Wazuh** will use machine’s hostname as agent’s name or you can define your custom name) and choose **default** group for initial configuration (all agents has to belong at least one group). In the fourth and fifth steps **Wazuh** will give us the installation script, we must run these scripts in our target (**DC** in this situation) to download agent service into this machine.

Deploy new agent

The screenshot shows the "Deploy new agent" window with the following steps:

- Select the package to download and install on your system:**
 - LINUX**: Options: RPM amd64, RPM aarch64, DEB amd64, DEB aarch64
 - WINDOWS**: Option: MSI 32/64 bits
 - macOS**: Options: Intel, Apple silicon

For additional systems and architectures, please check our documentation.
- Server address:**

This is the address the agent uses to communicate with the server. Enter an IP address or a fully qualified domain name (FDQN).

Assign a server address:
- Optional settings:**

By default, the deployment uses the hostname as the agent name. Optionally, you can use a different agent name in the field below.

Assign an agent name:
The agent name must be unique. It can't be changed once the agent has been enrolled.

Select one or more existing groups:
- Run the following commands to download and install the agent:**

```
Invoke-WebRequest -Uri https://packages.wazuh.com/4.x/windows/wazuh-agent-4.7.3-1.msi -OutFile $env:tmp\wazuh-agent; msiexec.exe /i $env:tmp\wazuh-agent /q WAZUH_MANAGER='172.16.10.12' WAZUH_AGENT_GROUP='default' WAZUH_REGISTRATION_SERVER='172.16.10.12'
```
- Start the agent:**

NET START WazuhSvc

Close



Go to the DC and run **Power Shell** as an administrator, then copy and paste the script which is given in the fourth step of agent deployment process. Once the agent service downloaded, run the second script to start **Wazuh** agent service.

```

Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Windows\system32> Invoke-WebRequest -Uri https://packages.wazuh.com/4.x/windows/wazuh-agent-4.7.3-1.msi -OutFile ${env:tmp}\wazuh-agent; msieexec.exe /i ${env:tmp}\wazuh-agent /q WAZUH_MANAGER='172.16.10.12' WAZUH_AGENT_GROUP='default' WAZUH_REGISTRATION_SERVER='172.16.10.12'
PS C:\Windows\system32> NET START WazuhSvc
The Wazuh service is starting.
The Wazuh service was started successfully.

PS C:\Windows\system32>

```

After successfully installing the agent service, we can control it with **GUI**. You can find its configuration and **.exe** files under “**C:\Program files (x86)\ossec-agent**” path.

Name	Date modified	Type	Size
client.keys.save	4/21/2024 8:56 PM	SAVE File	1 KB
dbsync.dll	2/29/2024 2:13 PM	Application exten...	473 KB
help	2/29/2024 1:06 PM	Text Document	2 KB
internal_options.conf	2/29/2024 1:06 PM	CONF File	15 KB
libfimdb.dll	2/29/2024 2:13 PM	Application exten...	559 KB
libgcc_s_dw2-1.dll	2/29/2024 2:13 PM	Application exten...	145 KB
libstdc++-6.dll	2/29/2024 2:13 PM	Application exten...	2,303 KB
libwazuhext.dll	2/29/2024 2:13 PM	Application exten...	7,158 KB
libwazuhshared.dll	2/29/2024 2:13 PM	Application exten...	1,149 KB
libwinpthread-1.dll	2/29/2024 2:13 PM	Application exten...	518 KB
LICENSE	2/29/2024 1:06 PM	Text Document	25 KB
local_internal_options.conf	2/29/2024 1:06 PM	CONF File	1 KB
local_internal_options.conf.save	2/29/2024 1:06 PM	SAVE File	1 KB
manage_agents	2/29/2024 2:12 PM	Application	1,362 KB
ossec.conf	4/23/2024 10:53 AM	CONF File	10 KB
ossec.conf.save	4/21/2024 8:27 PM	SAVE File	10 KB
ossec	4/23/2024 10:56 AM	Text Document	13 KB
REVISION	2/29/2024 1:06 PM	File	1 KB
rsync.dll	2/29/2024 2:13 PM	Application exten...	383 KB
syscollector.dll	2/29/2024 2:13 PM	Application exten...	558 KB
sysinfo.dll	2/29/2024 2:13 PM	Application exten...	612 KB
VERSION	2/29/2024 1:06 PM	File	1 KB
vista_sec	2/28/2024 9:25 PM	Text Document	92 KB
W wazuh-agent	2/29/2024 2:12 PM	Application	2,358 KB
wazuh-agent.state	4/23/2024 10:56 AM	STATE File	1 KB
W win32ui	2/29/2024 2:12 PM	Application	1,290 KB
win32ui.exe.manifest	2/28/2024 9:25 PM	MANIFEST File	1 KB
wpk_root.pem	2/28/2024 9:25 PM	PEM File	2 KB

44 items 1 item selected 1.25 MB

Manager IP:	172.16.10.12
Authentication key:	<insert_auth_key_here>
<input type="button" value="Save"/> <input type="button" value="Refresh"/>	

https://wazuh.com Revision 40714



Go back to the **Wazuh** interface in your browser and observe the changes. You can see that our first agent added successfully into our **SIEM** and its status is active. Just click on the agent count (which is 1 here) to open the agents window for detailed information.

The screenshot shows the Wazuh Modules interface. At the top, there are five status indicators: Total agents (1), Active agents (1), Disconnected agents (0), Pending agents (0), and Never connected agents (0). Below these are four main sections: SECURITY INFORMATION MANAGEMENT, AUDITING AND POLICY MONITORING, THREAT DETECTION AND RESPONSE, and REGULATORY COMPLIANCE. Each section contains several sub-components with icons and brief descriptions.

In this window, we can get much more information about agents and their status.

The screenshot shows the Wazuh Agents interface. It displays a summary of agent status: Active (1), Disconnected (0), Pending (0), and Never connected (0). It also shows the last registered agent (DC) and the most active agent (DC). The Agents table lists one entry: ID 001, Name DC, IP address 172.16.10.10, Group(s) default, Operating system Microsoft Windows Server 2022 Standard Evaluation 10.0.20348.587, Cluster node node01, Version v4.7.3, Status active. There are buttons for Deploy new agent, Refresh, Export formatted, and WQL.

2.2 Firewall joining (OPNsense)

One of the most essential security tools is a firewall. We must join it into our **SIEM**, because firewall logs are so precious for us, it stays in front of the whole infrastructure and control all network traffic. So most of security events happen in the firewall environment and these events have to be monitored from the centralized location.

We had deployed an opensource firewall, **OPNsense**, in our infrastructure and this firewall uses plugins to manage the services in it. For simplicity, we will collect only **IDS** logs from our firewall. This also will minimize the load on our system, because gathering logs and managing them consume additional resources such as **RAM**, **CPU**, memory etc.

Before we install and configure the **Wazuh** plugin in it, open your firewall **GUI** and go to the “**Intrusion Detection**” service and make sure it is running in the **IDS** mode.



Later got to the **System=>Firmware=>Plugins** and search for wazuh, then install **os-wazuh-agent** plugin.

NOTE: make sure you checked updates, otherwise it will not find the plugin.

After installing service, go to the **Services=>Wazuh Agent=>Settings** section, enable the service, inserts the **Wazuh** server machine's IP address, select **suricata** in **Applications** option and enable “**Intrusion detection events**” option.

Once the service successfully started, go back to the Wazuh GUI in your browser and observe the changes. As you see in the figure, our second agent successfully joined into our Wazuh server and the total number of agent is 2.



The dashboard displays the following statistics:

- Total agents: 2
- Active agents: 2
- Disconnected agents: 0
- Pending agents: 0
- Never connected agents: 0

SECURITY INFORMATION MANAGEMENT

- Security events: Browse through your security alerts, identifying issues and threats in your environment.
- Integrity monitoring: Alerts related to file changes, including permissions, content, ownership and attributes.

AUDITING AND POLICY MONITORING

- Policy monitoring: Verify that your systems are configured according to your security policies baseline.
- System auditing: Audit users behavior, monitoring command execution and alerting on access to critical files.
- Security configuration assessment: Scan your assets as part of a configuration assessment audit.

THREAT DETECTION AND RESPONSE

- Vulnerabilities: Discover what applications in your environment are affected by well-known vulnerabilities.
- MITRE ATT&CK: Security events from the knowledge base of adversary tactics and techniques based on real-world observations.

REGULATORY COMPLIANCE

- PCI DSS: Global security standard for entities that process, store or transmit payment cardholder data.
- NIST 800-53: National Institute of Standards and Technology Special Publication 800-53 (NIST 800-53) sets guidelines for federal information systems.
- TSC: Trust Services Criteria for Security, Availability, Processing Integrity, Confidentiality, and Privacy.
- GDPR: General Data Protection Regulation (GDPR) sets guidelines for processing of personal data.

STATUS

	Active (2)	Disconnected (0)	Pending (0)	Never connected (0)
Agents	2	0	0	0

DETAILS

Active	Disconnected	Pending	Never connected	Agents coverage
2	0	0	0	100.00%

Last registered agent: Firewall.cs301.local

Most active agent: DC

EVOLUTION

Last 24 hours

Count

15:00 18:00 21:00 00:00 03:00 06:00 09:00 timestamp per 10 minutes

Agents (2)

ID	Name	IP Address	Group(s)	Operating system	Cluster node	Version	Status	Actions
001	DC	172.16.10.10	default	Microsoft Windows Server 2022 Standard Evaluation 10.0.20348.587	node01	v4.7.3	active	
003	Firewall.cs301.local	172.16.10.2	default	BSD 13.2	node01	v4.7.3	active	

Rows per page: 10

2.3 Linux agent joining (WEB)

In this section, we will add our web server as the next agent into our **SIEM** environment. As you know we had deployed our web server in the **Oracle Linux** machine, so this time we must install **Wazuh** agent for **Linux** platform. The installation process is similar with the **Windows**, select platform type, enter the **Wazuh** server address and select the **default** group.

Select the package to download and install on your system:

LINUX

RPM amd64 RPM aarch64
 DEB amd64 DEB aarch64

WINDOWS

MSI 32/64 bits

macOS

Intel Apple silicon

For additional systems and architectures, please check our documentation [here](#).

Server address:

This is the address the agent uses to communicate with the server. Enter an IP address or a fully qualified domain name (FQDN).

Assign a server address: [here](#)

172.16.10.11



Optional settings:

By default, the deployment uses the hostname as the agent name. Optionally, you can use a different agent name in the field below.

Assign an agent name:

The agent name must be unique. It can't be changed once the agent has been enrolled.

Select one or more existing groups:

4 Run the following commands to download and install the agent:

```
curl -o wazuh-agent-4.7.3-1.x86_64.rpm https://packages.wazuh.com/4.x/yum/wazuh-agent-4.7.3-1.x86_64.rpm && sudo WAZUH_MANAGER='172.16.10.11' WAZUH_AGENT_GROUP='default' rpm -ihv wazuh-agent-4.7.3-1.x86_64.rpm
```

5 Start the agent:

```
sudo systemctl daemon-reload
sudo systemctl enable wazuh-agent
sudo systemctl start wazuh-agent
```

Close

After generating installation script, go to the web server and open your terminal window and run the the script which is given in the fourth step with **root** privileges.

```
[root@web ~]# curl -o wazuh-agent-4.7.3-1.x86_64.rpm -k https://packages.wazuh.com/4.x/yum/wazuh-agent-4.7.3-1.x86_64.rpm && sudo WAZUH_MANAGER='172.16.10.11' WAZUH_AGENT_GROUP='default' rpm -ihv wazuh-agent-4.7.3-1.x86_64.rpm
% Total    % Received % Xferd  Average Speed   Time   Time     Time Current
          Dload  Upload Total   Spent    Spent    Left  Speed
100 9247k  100 9247k    0      0  2816k      0  0:00:03  0:00:03  --:--:-- 2816k
warning: wazuh-agent-4.7.3-1.x86_64.rpm: Header V3 RSA/SHA256 Signature, key ID 29111145: NOKEY
Verifying...                               #####[100%]
Preparing...                                #####[100%]
      package wazuh-agent-4.7.3-1.x86_64 is already installed
[root@web ~]# ll wazuh-agent-4.7.3-1.x86_64.rpm
-rw-r--r-- 1 root root 9468952 Apr 23 11:56 wazuh-agent-4.7.3-1.x86_64.rpm
[root@web ~]#
```

After installing agent package, enable and start the **wazuh-agent** service.

```
[root@web ~]# sudo systemctl enable wazuh-agent
[root@web ~]# sudo systemctl start wazuh-agent
[root@web ~]# systemctl status wazuh-agent.service
● wazuh-agent.service - Wazuh agent
   Loaded: loaded (/usr/lib/systemd/system/wazuh-agent.service; enabled; vendor
r>
   Active: active (running) since Tue 2024-04-23 11:58:33 +04; 34s ago
     Process: 8736 ExecStart=/usr/bin/env /var/ossec/bin/wazuh-control start (cod
```



Now go back to the **Wazuh GUI** in your browser and you will see that the third agent was joined successfully and the total number of agents is **3**.

Follow the procedures and join rest of the machines into **SIEM** environment in our infrastructure.

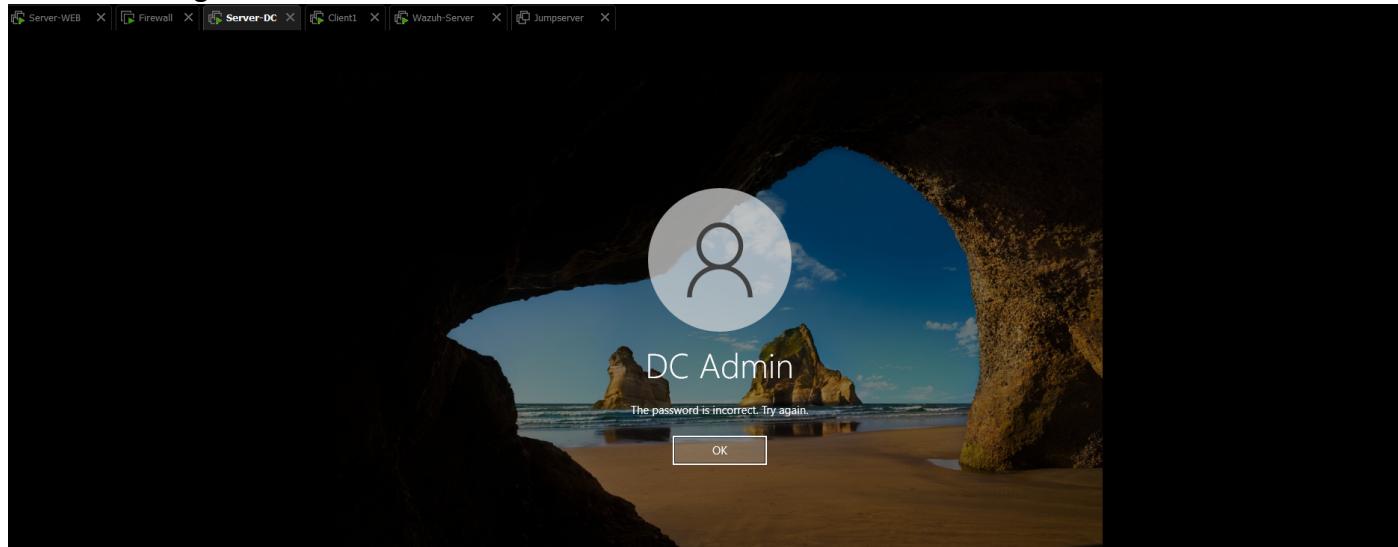
ID	Name	IP address	Group(s)	Operating system	Cluster node	Version	Status	Actions
001	DC	172.16.10.10	default	Microsoft Windows Server 2022 Standard Evaluation 10.0.20348.587	node01	v4.7.3	active	Details
003	Firewall.cs301.local	172.16.10.2	default	BSD 13.2	node01	v4.7.3	active	Details
004	web	172.16.10.11	default	Oracle Linux Server 8.9	node01	v4.7.3	active	Details
005	Client-1	172.16.20.10	default	Microsoft Windows 10 Pro 10.0.19045.4170	node01	v4.7.3	active	Details
006	jump	172.16.30.10	default	Microsoft Windows Server 2022 Standard Evaluation 10.0.20348.587	node01	v4.7.3	active	Details



Testing agents

3.1 Testing DC

In order to test our **DC** agent if it sends security logs or not, just try to login with wrong credentials as shown below in the figure.



Now open your **Wazuh server GUI** in your browser and enter to the **DC** agent. Here open “**Security events**” and observe the logs. As you see our unsuccessful login attempt is displayed here.

Time	Technique(s)	Tactic(s)	Description	Level	Rule ID
Apr 23, 2024 @ 14:17:15.378	T1078 T1531	Defense Evasion, Persistence, Privilege Escalation, Initial Access, Impact	Logon failure - Unknown user or bad password.	5	60122
Apr 23, 2024 @ 14:17:12.484			Windows User Logoff.	3	60137
Apr 23, 2024 @ 14:17:10.082			Windows User Logoff.	3	60137
Apr 23, 2024 @ 14:17:10.065	T1078	Defense Evasion, Persistence, Privilege Escalation, Initial Access	Windows logon success.	3	60106
Apr 23, 2024 @ 14:17:10.050			Windows User Logoff.	3	60137
Apr 23, 2024 @ 14:17:03.527	T1078	Defense Evasion, Persistence, Privilege Escalation, Initial Access	Windows logon success.	3	60106
Apr 23, 2024 @ 14:17:03.521	T1078	Defense Evasion, Persistence, Privilege Escalation, Initial Access	Windows logon success.	3	60106
Apr 23, 2024 @ 14:16:42.531	T1550.002 T1078.002 T1021.001	Defense Evasion, Lateral Movement, Persistence, Privilege Escalation, Initial Access	Successful Remote Logon Detected - User:ANONYMOUS LOGON - NTLM authentication, possible pass-the-hash attack - Possible RDP connection. Verify that CLIENT-1 is allowed to perform RDP connections	6	92657
Apr 23, 2024 @ 14:16:16.984	T1550.002 T1078.002 T1021.001	Defense Evasion, Lateral Movement, Persistence, Privilege Escalation, Initial Access	Successful Remote Logon Detected - User:ANONYMOUS LOGON - NTLM authentication, possible pass-the-hash attack - Possible RDP connection. Verify that JUMP is allowed to perform RDP connections	6	92657



3.2 Testing OPNsense

In the previous capture we enabled **OPNsense IDS** mode and sent its security event to our **Wazuh** server, and in this capture we will test it.

Open the firewall **GUI** environment and go to the **Services=>Intrusion Detection=>Rules** section, in the search box type **urlhaus** to filter only for **URLhaus** rules and press the any rule's edit button.

In the popped up window, copy the **reference_html** address and open it in any of machines from **SERVERS** or **CLIENTS** networks (we activated firewall **IDS** mode only these two networks) as shown in the figure.

Signature Id	83687121
Revision	1
Message	URLhaus Known malware download URL detected (2824021)
ClassType	trojan-activity
created_at	2024_04_23
reference_html	urlhaus.abuse.ch/url/2824021/
source	abuse.ch.urlhaus.rules
status	enabled
Action	Alert

In the new page, copy the **URL** and open it in the new window.



Database Entry

ID:	2824021	Actions
URL:	http://42.234.179.141:50247/	
URL Status:	Online (spreading malware for 11 minutes)	
Host:	42.234.179.141	
Date added:	2024-04-23 09:58:07 UTC	
Threat:	Malware download	
Reporter:	geenensp	

Once you go to the copied address, the testing malware file will be downloaded into your machine.

Then go back to your firewall **GUI** and observe the alerts in the **Services => Intrusion Detection => Administration => Alerts** section. As you see, our **IDS** service detected the malicious file downloading .

Now go back to the **Wazuh GUI** in your browser, and open the **Firewall** agent. In the agent window, open the “**Security events**” section and observe the logs. You can see that all alert logs are belong to our **IDS** service in **OPNsense** firewall. For detailed information just click on the rule.

Time	Technique(s)	Tactic(s)	Description	Level	Rule ID
Apr 23, 2024 @ 14:12:15.823			Suricata: Alert - ET POLICY Executable and linking format (ELF) file download	3	86601
Apr 23, 2024 @ 14:11:11.321			Suricata: Alert - ET POLICY Executable and linking format (ELF) file download	3	86601
Apr 23, 2024 @ 14:11:11.278			Suricata: Alert - ET POLICY Executable and linking format (ELF) file download	3	86601



Wazuh Proof of Concept

4.1 Blocking a known malicious actor

In this capture we will implement checklist feature into **Wazuh**, which will allow us to block malicious IPs in order to protect our web server.

Original article: <https://documentation.wazuh.com/current/proof-of-concept-guide/block-malicious-actor-ip-reputation.html>

As you know we have the web server in our infrastructure and we have to protect it from malicious actors. **Wazuh** lets us to add another security layer to make our infrastructure much more secure. This layer is checking IP addresses of clients who try to connect our web server, and block them if they are in our blacklist. In this lab, we will demonstrate it with two machines, one is our web server and the other one is the blocked source. First we will download a ready blacklist and appened our malicious IP address in it.

All of **Wazuh** configuration files are located under `/var/ossec` path and checklists are placed in `/var/ossec/etc/lists` directory. We must download our blacklist here.

```
[root@wazuh ~]# curl https://raw.githubusercontent.com/firehol/blocklist-ipsets/master/alienvault_reputation.ipset -o /var/ossec/etc/lists/alienvault_reputation.ipset -k
  % Total    % Received % Xferd  Average Speed   Time     Time      Current
                                         Dload  Upload Total Spent   Left  Speed
100  9495  100  9495    0     0  3561      0  0:00:02  0:00:02  ---:---  3561
[root@wazuh ~]# ll /var/ossec/etc/lists/alienvault_reputation.ipset
-rw-r--r--. 1 root root 9495 Apr 28 11:53 /var/ossec/etc/lists/alienvault_reputation.ipset
```

This free blacklist contain public IP addresses of malicious actors around the world, but we will test this feature in our local environment. So we need to add our choosen machine's (attacker) IP into this checklist. In this situation our attacker's IP is **192.168.1.133**.

```
[root@wazuh ~]# tail -n 2 /var/ossec/etc/lists/alienvault_reputation.ipset
223.155.34.126
223.159.88.8
[root@wazuh ~]# echo "192.168.1.133" >> /var/ossec/etc/lists/alienvault_reputation.ipset
[root@wazuh ~]# tail -n 2 /var/ossec/etc/lists/alienvault_reputation.ipset
223.159.88.8
192.168.1.133
[root@wazuh ~]#
```

Our blacklist has to be in ".cdb" file format, oterwise **Wazuh** couldn't read it. For converting the ".ipset", we need to download a script tool called **iplist-to-cdblist.py**. The figure demonstrate the converting process, the **python3** binary must be used to run the downloaded script.

```
[root@wazuh ~]# curl https://wazuh.com/resources/iplist-to-cdblist.py -o /tmp/iplist-to-cdblist.py -k
  % Total    % Received % Xferd  Average Speed   Time     Time      Current
                                         Dload  Upload Total Spent   Left  Speed
100  1570  100  1570    0     0  1227      0  0:00:01  0:00:01  ---:---  1227
[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 ~]# ll /var/ossec/etc/lists/blacklist-alienvault
-rw-r--r--. 1 wazuh wazuh 9320 Apr 28 11:57 /var/ossec/etc/lists/blacklist-alienvault
[root@wazuh ~]#
```

The next step is to define a rule to trigger the Wazuh defualt scripts to block the blacklist addresses. **Wazuh** system rules are located under `/var/ossec/etc/rules` path, and the default custom rule file is `local_rules.xml`. Open this file and add the rule shown in the figure.

```
[root@wazuh ~]# vim /var/ossec/etc/rules/local_rules.xml
```



```
<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>
"/var/ossec/etc/rules/local_rules.xml" 57L, 1739C written
```

After defining rule, we need to add our blacklist location into **Wazuh** configuration file, **ossec.conf** which is placed under **/var/ossec/etc/ossec.conf** path.

```
[root@wazuh ~]# vim /var/ossec/etc/ossec.conf
```

This information has to be in 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/blacklist-alienVault</list>
  <list>etc/lists/blacklist-alienVault</list>

  <!-- User-defined ruleset -->
  <decoder_dir>etc/decoders</decoder_dir>
  <rule_dir>etc/rules</rule_dir>
</ruleset>

"/var/ossec/etc/ossec.conf" 413L, 11135C written
```

Inside this configuration file, we need to define the **Wazuh** server's default script **firewall-drop** to be activated by our custom rule (100100) which we added into the **local_rules.xml** file above. This information has to be in the **<active-response>** block. **Wazuh**'s active response binaries are located under **/var/ossec/etc/active-response/bin** path.

```
<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" 422L, 11334C written
```

After configuring all files, the **wazuh-manager** service must be restarted.

```
[root@wazuh ~]# systemctl restart wazuh-manager.service
```

The final step is testing our active response feature. Open attacker machine whose **IP** address is **192.168.1.133**, and run the curl command to send **GET** request to our web server.

NOTE: In our infrastructure, **WEB** server's **IP** address is **172.16.10.11** and it uses custom **VMNET1** adapter in the **SERVERS** network. All traffic with external networks goes through **172.16.10.1** **IP** address and although we send requests from different **IP** addresses, only this address shown as a source address in the access



logs of web service. So in order to demonstrate only this lab, I added another network adapter to **WEB** server and hosted it in the **NAT** network with **192.168.1.138** IP address.

```
(root㉿kali)-[~]
# hostname -I
192.168.1.133

(root㉿kali)-[~]
# curl http://192.168.1.138
```

In the WEB agent, open **/var/log/httpd/access_log** file and observe the **HTTP GET** requests.

```
[root@web ~]# tail -n 1 /var/log/httpd/access_log
192.168.1.133 - - [28/Apr/2024:12:09:00 +0400] "GET / HTTP/1.1" 302 - "-" "curl/8.5.0"
```

We can see that our request is added to the access logs. Now open the **Wazuh GUI** in your browser and go to the **web** agent and open the “**Security events**”.

Time	Technique(s)	Tactic(s)	Description	Level	Rule ID
Apr 28, 2024 @ 12:09:02.276			Host Blocked by firewall-drop Active Response	3	651
Apr 28, 2024 @ 12:09:01.193			IP address found in AlienVault reputation database.	10	100100

Event Data					
Table	JSON	Rule			
	@timestamp: 2024-04-28T08:09:02.276Z _id: IMbAI48BZOG4Xyal7Gg agent.id: 008 agent.ip: 192.168.1.138 agent.name: web data.command: add data.origin.module: wazuh-execd data.origin.name: node01 data.parameters.alert.agent.id: 008 data.parameters.alert.agent.ip: 192.168.1.138 data.parameters.alert.agent.name: web data.parameters.alert.data.id: 302 data.parameters.alert.data.protocol: GET data.parameters.alert.data.script: 192.168.1.133 data.parameters.alert.data.url: / data.parameters.alert.decoder.name: web-accesslog data.parameters.alert.full_log: 192.168.1.133 - - [28/Apr/2024:12:09:00 +0400] "GET / HTTP/1.1" 302 - "-" "curl/8.5.0" data.parameters.alert.id: 1714281741.2352276 data.parameters.alert.location: /var/log/httpd/access_log				

Once we run the **curl** command, our **GET** request appeared in the **Apache2** access log with our attacker's **IP** address and in the next step Wazuh match this address with blacklist then trigger the active response script, **firewall-drop** and block the matched address.



4.2 Responding to network attacks with Suricata and Wazuh XDR

In firewall integration capture, we integrated firewall **IDS** feature to our **Wazuh SIEM** and monitored alerts via security events. This feature controlled two networks, **SERVERS** and **CLIENTS**, but, in this lab we will implement **Suricata** service to our web server in host based. **Suricata** will be integrated to **Wazuh** active response module, **XDR**, and any malicious action will be blocked by **XDR** in order to protect our web server.

Original article: <https://wazuh.com/blog/responding-to-network-attacks-with-suricata-and-wazuh-xdr/>

First we need to install **Suricata** to our server, but we will not criticize this process in this lab. You can find full installation guide from the source below.

Suricata deploy: https://github.com/Yagub-Hajiyev/Lab_Setup/blob/710ab7546f9ca38f91f676be03bf526565c10615/Suricata_installation.pdf

After successfully deploying **Suricata** as **IDS**, we must download and new rules and add them to our **Suricata** service in order to detect malicious activities.

Rules: <https://rules.emergingthreats.net/open/suricata-6.0.17/emerging.rules.tar.gz>

In the lab we will use only **emerging-scan.rules** to detect **Nmap** scans and block the action with **Wazuh XDR** service. Download compressed file and extract it, then copy the ruleset into **Suricata** default rule path as displayed in the figure.

```
[root@web ~]# ll
total 4168
-rw----- 1 root root 1034 Mar 31 16:30 anaconda-ks.cfg
-rw-r--r-- 1 root root 4258851 Apr 27 21:39 emerging.rules.tar.gz
-rw-r--r-- 1 root root 1261 Mar 31 16:49 initial-setup-ks.cfg
[root@web ~]# tar -zxf emerging.rules.tar.gz
[root@web ~]# ll
total 4172
-rw----- 1 root root 1034 Mar 31 16:30 anaconda-ks.cfg
-rw-r--r-- 1 root root 4258851 Apr 27 21:39 emerging.rules.tar.gz
-rw-r--r-- 1 root root 1261 Mar 31 16:49 initial-setup-ks.cfg
drwxr-xr-x 2 root root 4096 Apr 27 00:29 rules
[root@web ~]# ll rules/emerging-scan.rules
-rw-r--r-- 1 root root 144847 Apr 27 00:29 rules/emerging-scan.rules
[root@web ~]# cp rules/emerging-scan.rules /var/lib/suricata/rules/
[root@web ~]# ll /var/lib/suricata/rules/
total 220
-rw-r--r-- 1 root suricata 3228 Apr 27 21:40 classification.config
-rw-r--r-- 1 root suricata 144847 Apr 27 22:40 emerging-scan.rules
-rw-r--r-- 1 root suricata 70386 Apr 27 21:40 suricata.rules
[root@web ~]#
```

Later open the **suricata.yaml** file and define the new rule name in the configuration, then restart the **suricata.service** to reload new configuration.

```
[root@web ~]# vim /etc/suricata/suricata.yaml
```

```
default-rule-path: /var/lib/suricata/rules
rule-files:
- suricata.rules
```

```
default-rule-path: /var/lib/suricata/rules
rule-files:
- suricata.rules
- emerging-scan.rules
```

```
[root@web ~]# systemctl restart suricata.service
```



Now open the Wazuh server and create a new group, named Suricata, as shown in the figure. **Wazuh** has default scripts to manage agents and groups, and one of them is **agent_groups** binary. You can manage them also from **GUI** in your browser. After creating new group, use **manage_agents** binary to add our web server into that **Suricata** group.

```
[root@wazuh ~]# /var/ossec/bin/agent_groups -a -g Suricata -q
Group 'Suricata' created.
[root@wazuh ~]# /var/ossec/bin/manage_agents -l

Available agents:
  ID: 006, Name: jump, IP: any
  ID: 003, Name: Firewall.cs301.local, IP: any
  ID: 004, Name: web, IP: any
  ID: 005, Name: Client-1, IP: any
  ID: 007, Name: DC, IP: any

[root@wazuh ~]# /var/ossec/bin/agent_groups -a -i 4 -g Suricata -q
Group 'Suricata' added to agent '004'.
[root@wazuh ~]#
```

Every group has its own configuration file, **agent.conf** which affect to all agents in it. As you know our default configuration file is **ossec.conf** and every agent has this file in its host. Now we will append new configuration to our **Suricata** group (where we have only one agent yet) to say reading **IDS** logs in **json** format.

```
[root@wazuh ~]# vim /var/ossec/etc/shared/Suricata/agent.conf
```

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

  <!-- Shared agent configuration here -->
</agent_config>
```

In the next step, new decoder configurations has to be added to our default **local_decoder.xml** file to map source **IP** addresses from logs to our active response script, **firewall-drop**.

```
[root@wazuh ~]# vim /var/ossec/etc/decoders/local_decoder.xml
```

```
<decoder name="local_decoder_example">
  <program_name>local_decoder_example</program_name>
</decoder>

<decoder name="json">
  <prematch>^{\s*</prematch>
</decoder>

<decoder name="json_child">
  <parent>json</parent>
  <regex type="pcre2">"src_ip": "([^\"]+)"</regex>
  <order>srcip</order>
</decoder>

<decoder name="json_child">
  <parent>json</parent>
  <plugin_decoder>JSON_Decoder</plugin_decoder>
</decoder>
-- INSERT --
```



We need a custom rule for **Wazuh** active response module to trigger **firewall-drop** script on our **WEB** server agent. This rule will detect **Nmap** scan activities by help of **Suricata** and our **XDR** will block the malicious actor. Open the default custom rule file, **local_rules.xml**, in the **Wazuh** server and add the following rule to it.

```
[root@wazuh ~]# vim /var/ossec/etc/rules/local_rules.xml
```

```
<rule id="100204" level="12">
<if_sid>86600</if_sid>
<field name="event_type">^alert$</field>
<match>ET SCAN Nmap Scripting Engine User-Agent Detected (Nmap Scripting Engine)</match>
<description>Nmap scripting engine detected. </description>
<mitre>
  <id>T1595</id>
</mitre>
</rule>
</group>
```

```
"/var/ossec/etc/rules/local_rules.xml" 118L, 3760C written
```

Later open the **Wazuh** server configuration file, **ossec.conf**, and add the following active response block to trigger **firewall-drop** script and restart the **wazuh-manager** service for applying new configuration.

```
[root@wazuh ~]# vim /var/ossec/etc/ossec.conf
```

```
<ossec_config>
<active-response>
<command>firewall-drop</command>
<location>local</location>
<rules_id>100204</rules_id>
<timeout>180</timeout>
</active-response>
</ossec_config>
```

```
"/var/ossec/etc/ossec.conf" 461L, 12277C written
```

```
[root@wazuh ~]# vim /var/ossec/etc/ossec.conf
[root@wazuh ~]# systemctl restart wazuh-manager.service
[root@wazuh ~]# |
```

Now we are ready to test our integration, go to your **Kali Linux** machine and first test your connectivity with the web server. Later run the Nmap scan with the script as shown below in the figure.

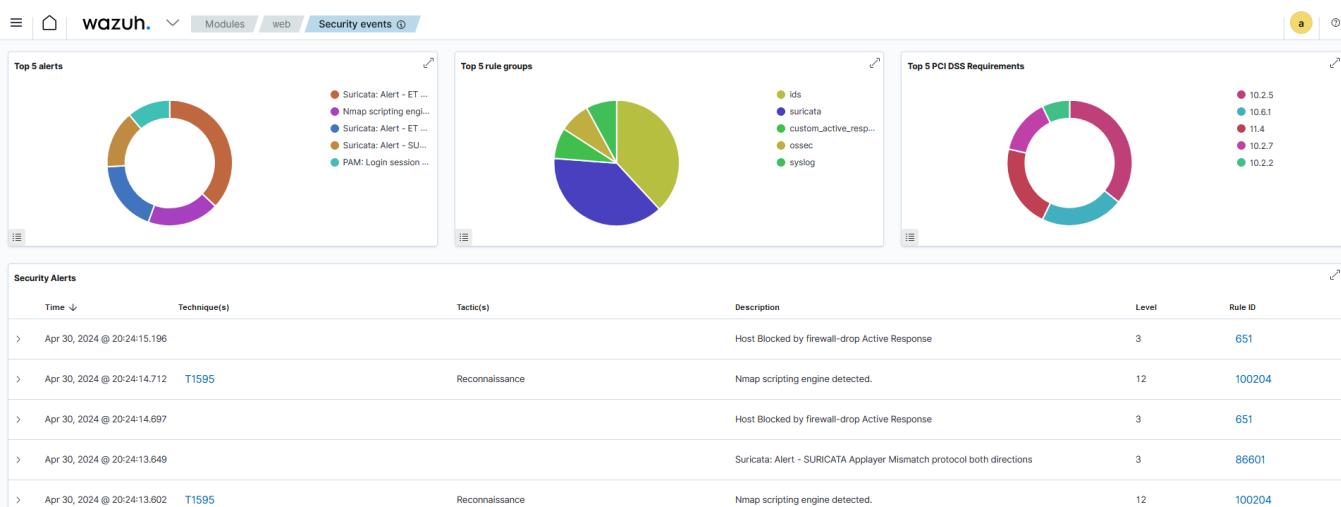


```
(root㉿kali)-[~]
└─# nmap -sS --script=vuln 172.16.10.11
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-04-30 20:23 +04
Nmap scan report for 172.16.10.11
Host is up (0.00077s latency).
Not shown: 999 filtered tcp ports (no-response)
PORT      STATE SERVICE
80/tcp    open  http
| http-cookie-flags:
|   /:
|     PHPSESSID:
|       httponly flag not set
|_http-aspnet-debug: ERROR: Script execution failed (use -d to debug)
|_http-csrf: Couldn't find any CSRF vulnerabilities.
|_http-dombased-xss: Couldn't find any DOM based XSS.
|_http-vuln-cve2014-3704: ERROR: Script execution failed (use -d to debug)
|_http-stored-xss: Couldn't find any stored XSS vulnerabilities.

Nmap done: 1 IP address (1 host up) scanned in 242.25 seconds

(root㉿kali)-[~]
└─#
```

Later open your **Wazuh GUI** and select the web server agent where you installed **Suricata** service and analyse the “**Security events**”. You can see that the attacker machine is blocked by **Wazuh** active response module.





4.3 Detecting malware using YARA integration

YARA integration with **Wazuh** allows us to detect malwares based on their textual or binary patterns. **YARA** requires **Wazuh FIM** module to scan our selected directories in realtime. When we add or modify any file under **FIM** directory, **YARA** scans patterns with its rules and when it match any of files then it trigger **Wazuh** active response module and delete the matched file with **yara.sh** script.

Original article: <https://documentation.wazuh.com/current/proof-of-concept-guide/detect-malware-yara-integration.html>

In this lab, we will install **YARA** into our web server. Lets start with installing some required packages into our agent.

```
[root@web ~]# yum install -y make automake gcc autoconf libtool openssl-devel pkg-config jq
Last metadata expiration check: 6:08:11 ago on Sun 28 Apr 2024 11:19:51 AM +04.
Package pkgconf-pkg-config-1.4.2-1.el8.x86_64 is already installed.
Package jq-1.6-7.0.3.el8.x86_64 is already installed.
Dependencies resolved.
=====
Package           Architecture   Version      Repository    Size
=====
Installing:
autoconf          noarch        2.69-29.el8   ol8_appstream 710 k
automake          noarch        1.16.1-8.el8  ol8_appstream 713 k
gcc               x86_64       8.5.0-20.0.3.el8 ol8_appstream 23 M
libtool            x86_64       2.4.6-25.el8  ol8_appstream 709 k
make              x86_64       1:4.2.1-11.el8 ol8_baseos_latest 498 k
openssl-devel     x86_64       1:1.1.1k-12.el8_9 ol8_baseos_latest 2.3 M
Upgrading:
libgcc             x86_64       8.5.0-20.0.3.el8 ol8_baseos_latest 91 k
libgomp            x86_64       8.5.0-20.0.3.el8 ol8_baseos_latest 217 k
openssl            x86_64       1:1.1.1k-12.el8_9 ol8_baseos_latest 710 k
openssl-libs       x86_64       1:1.1.1k-12.el8_9 ol8_baseos_latest 1.5 M
Installing dependencies:
cpp                x86_64       8.5.0-20.0.3.el8 ol8_appstream 10 M
glibc-devel         x86_64       2.28-236.0.1.el8_7 ol8_baseos_latest 87 k
isl                x86_64       0.16.1-6.el8   ol8_appstream 841 k
keyutils-libs-devel x86_64       1.5.10-9.el8   ol8_baseos_latest 48 k
krb5-devel          x86_64       1.18.2-25.0.1.el8_8 ol8_baseos_latest 562 k
```

After installing packages, download the yara installation file from **GitHub** repository and decomress it such as in the figure.

```
[root@web ~]# curl -LO https://github.com/VirusTotal/yara/archive/v4.2.3.tar.gz
% Total    % Received % Xferd  Average Speed   Time   Time   Current
          Dload  Upload Total   Spent    Left  Speed
  0     0    0     0    0     0      0 --:--:--  0:00:02 --:--:--    0
100 1258k  0 1258k  0     0   214k      0 --:--:--  0:00:05 --:--:--  410k
[root@web ~]# ll v4.2.3.tar.gz
-rw-r--r-- 1 root root 1288334 Apr 28 17:41 v4.2.3.tar.gz
[root@web ~]# tar -xvzf v4.2.3.tar.gz -C /usr/local/bin/ && rm -f v4.2.3.tar.gz
```

Later go into the decompressed **YARA** directory and run the following commands to install the tool.

```
[root@web ~]# cd /usr/local/bin/yara-4.2.3/
[root@web yara-4.2.3]# ./bootstrap.sh && ./configure && make && make install && make check
libtoolize: putting auxiliary files in AC_CONFIG_AUX_DIR, 'build-aux'.
libtoolize: copying file 'build-aux/ltmain.sh'
libtoolize: putting macros in AC_CONFIG_MACRO_DIRS, 'm4'.
libtoolize: copying file 'm4/libtool.m4'
libtoolize: copying file 'm4/ltoptions.m4'
```



```
PASS: test-arena
PASS: test-alignment
PASS: test-atoms
PASS: test-api
PASS: test-rules
PASS: test-pe
PASS: test-elf
PASS: test-version
PASS: test-bitmask
PASS: test-math
PASS: test-stack
PASS: test-re-split
PASS: test-async
PASS: test-exception
PASS: test-dotnet
=====
Testsuite summary for yara 4.2.3
=====
# TOTAL: 15
# PASS: 15
# SKIP: 0
# XFAIL: 0
# FAIL: 0
# XPASS: 0
# ERROR: 0
```

Run the command **yara** and download it.

```
[root@web yara-4.2.3]# yara
bash: yara: command not found...
Install package 'yara' to provide command 'yara'? [N/y] y

* Waiting in queue...
* Loading list of packages....
The following packages have to be installed:
yara=4.2.3-1.el8.x86_64           Pattern matching Swiss knife for malware researchers
Proceed with changes? [N/y] y

* Waiting in queue...
* Waiting for authentication...
* Waiting in queue...
* Downloading packages...
* Requesting data...
* Testing changes...
* Installing packages...
yara: wrong number of arguments
Usage: yara [OPTION]... [NAMESPACE:]RULES_FILE... FILE | DIR | PID

Try `--help` for more options

[root@web yara-4.2.3]#
```

After the installation again run the command, **yara**, and make sure it works properly.

```
[root@web yara-4.2.3]# yara
yara: wrong number of arguments
Usage: yara [OPTION]... [NAMESPACE:]RULES_FILE... FILE | DIR | PID

Try '--help' for more options
[root@web yara-4.2.3]#
```

Create a directory path `/tmp/yara/rules` and run the following commands to download **YARA** detection rules.



Now create a new script named `yara.sh` under `/var/ossec/active-response/bin` path, copy the example script from given article [link](#) and paste in it.

```
[root@web ~]# vim /var/ossec/active-response/bin/yara.sh
```

```
while [ ${size} -ne ${actual_size} ]; do
    sleep 1
    size=$actual_size
    actual_size=$(stat -c %s ${FILENAME})
done

#----- Analyze parameters -----
if [[ ! $YARA_PATH ]] || [[ ! $YARA_RULES ]]
then
    echo "wazuh-yara: ERROR - Yara active response error. Yara path and rules parameters are mandatory." >> ${LOG_FILE}
}
exit 1
fi

#----- Main workflow -----
# Execute Yara scan on the specified filename
yara_output=$(("$YARA_PATH"/yara -w -r "$YARA_RULES" "$FILENAME"))

if [[ $yara_output != "" ]]
then
    # Iterate every detected rule and append it to the LOG_FILE
    while read -r line; do
        echo "wazuh-yara: INFO - Scan result: $line" >> ${LOG_FILE}
    done <<< "$yara_output"
fi

exit 0;
"/var/ossec/active-response/bin/yara.sh" [New] 51L, 1486C written
```

51,7 Bot

Later assign the group owner script to `wazuh` and give the following permissions to run the script properly.

```
[root@web ~]# vim /var/ossec/active-response/bin/yara.sh
[root@web ~]# chown root:wazuh /var/ossec/active-response/bin/yara.sh
[root@web ~]# chmod 750 /var/ossec/active-response/bin/yara.sh
[root@web ~]# ll /var/ossec/active-response/bin/yara.sh
-rwxr-x--- 1 root wazuh 1486 Apr 28 17:53 /var/ossec/active-response/bin/yara.sh
[root@web ~]#
```

We will use `/tmp/yara/malware` directory to test our integration. First we need to define this path in our agent's configuration file to monitor activities with FIM integration. Open `ossec.conf` file and add the following `syscheck` block in it and restart the `wazuh-agent.service`.

```
[root@web ~]# vim /var/ossec/etc/ossec.conf
```

```
<syscheck>
    <disabled>no</disabled>
    <directories realtime="yes">/tmp/yara/malware</directories>
</syscheck>

</ossec_config>
"/var/ossec/etc/ossec.conf" 229L, 6250C written
```

```
[root@web ~]# vim /var/ossec/etc/ossec.conf
[root@web ~]# systemctl restart wazuh-agent.service
[root@web ~]#
```



At this point we are done with agent configuration, go to the **Wazuh** server and open **local_rules.xml** file and add following custom rules blocks in it shown in the figure below.

```
[root@wazuh ~]# vim /var/ossec/etc/rules/local_rules.xml
```

```
<group name="syscheck">
  <rule id="100300" level="7">
    <if_sid>550</if_sid>
    <field name="file">/tmp/yara/malware/</field>
    <description>File modified in /tmp/yara/malware/ directory.</description>
  </rule>
  <rule id="100301" level="7">
    <if_sid>554</if_sid>
    <field name="file">/tmp/yara/malware/</field>
    <description>File added to /tmp/yara/malware/ 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>
"/var/ossec/etc/rules/local_rules.xml" 112L, 3541C written
```

112,8

In order to extract information from **YARA** scan results, open the **local_decoder.xml** file and add following configuration in it.

```
[root@wazuh ~]# vim /var/ossec/etc/decoders/local_decoder.xml
```

```
<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>
"/var/ossec/etc/decoders/local_decoder.xml" 52L, 1398C written
```

Open **Wazuh** server configuration file add the information shown in the figure and restart the **wazuh-manager.service**.

```
[root@wazuh ~]# vim /var/ossec/etc/ossec.conf
```

```
<ossec_config>
  <command>
    <name>yara_linux</name>
    <executable>yara.sh</executable>
    <extra_args>-yara_path /usr/local/bin -yara_rules /tmp/yara/rules/yara_rules.yar</extra_args>
    <timeout_allowed>no</timeout_allowed>
  </command>

  <active-response>
    <command>yara_linux</command>
    <location>local</location>
    <rules_id>100300,100301</rules_id>
  </active-response>
</ossec_config>
"/var/ossec/etc/ossec.conf" 462L, 12335C written
```



All configurations are done and we are ready to test our integrations. Create the /tmp/yara/malware directory which is monitored by **Wazuh FIM** module and new script for downloadin malware samples.

```
[root@web ~]# mkdir -p /tmp/yara/malware
[root@web ~]# vim /tmp/yara/malware/malware_downloader.sh
```

```
# Mirai
echo "# Mirai: https://en.wikipedia.org/wiki/Mirai_(malware)"
echo "Downloading malware sample..."
fetch_sample "https://wazuh-demo.s3-us-west-1.amazonaws.com/mirai" "/tmp/yara/malware/mirai" && echo "Done!" || echo "Error while downloading."
echo

# Xbash
echo "# Xbash: https://unit42.paloaltonetworks.com/unit42-xbash-combines-botnet-ransomware-coinmining-worm-targets-linux-windows/"
echo "Downloading malware sample..."
fetch_sample "https://wazuh-demo.s3-us-west-1.amazonaws.com/xbash" "/tmp/yara/malware/xbash" && echo "Done!" || echo "Error while downloading."
echo

# VPNFilter
echo "# VPNFilter: https://news.sophos.com/en-us/2018/05/24/vpnfilter-botnet-a-sophoslabs-analysis/"
echo "Downloading malware sample..."
fetch_sample "https://wazuh-demo.s3-us-west-1.amazonaws.com/vpn_filter" "/tmp/yara/malware/vpn_filter" && echo "Done!" || echo "Error while downloading."
echo

# Webshell
echo "# WebShell: https://github.com/SecWiki/WebShell-2/blob/master/Php/Worse%20Linux%20Shell.php"
echo "Downloading malware sample..."
fetch_sample "https://wazuh-demo.s3-us-west-1.amazonaws.com/webshell" "/tmp/yara/malware/webshell" && echo "Done!" || echo "Error while downloading."
echo
fi
"/tmp/yara/malware/malware_downloader.sh" [New] 46L, 1819C written
```

46,2

Bot

Now run the new script to start attack emulation. This script will donwload malware sample into the **/tmp/yara/malware** directoy and our **FIM** module will trigger the custom rules which we append into the **local_rules.xml** file in previous configurations. Once the rulea will be activated, they will trigger the **yara.sh** script to compare the patterns of file with **YARA** rules.

```
[root@web ~]# bash /tmp/yara/malware/malware_downloader.sh
WARNING: Downloading Malware samples, please use this script with caution.
Do you want to continue? (y/n)y

# Mirai: https://en.wikipedia.org/wiki/Mirai_(malware)
Downloading malware sample...
Done!

# Xbash: https://unit42.paloaltonetworks.com/unit42-xbash-combines-botnet-ransomware-coinmining-worm-targets-linux-windows/
Downloading malware sample...
Done!

# VPNFilter: https://news.sophos.com/en-us/2018/05/24/vpnfilter-botnet-a-sophoslabs-analysis/
Downloading malware sample...
Done!

# WebShell: https://github.com/SecWiki/WebShell-2/blob/master/Php/Worse%20Linux%20Shell.php
Downloading malware sample...
Done!

[root@web ~]#
```



After running the emulation script, go to the **Wazuh GUI** in your browser and monitor the events logs in the web agent's "Security events" section.

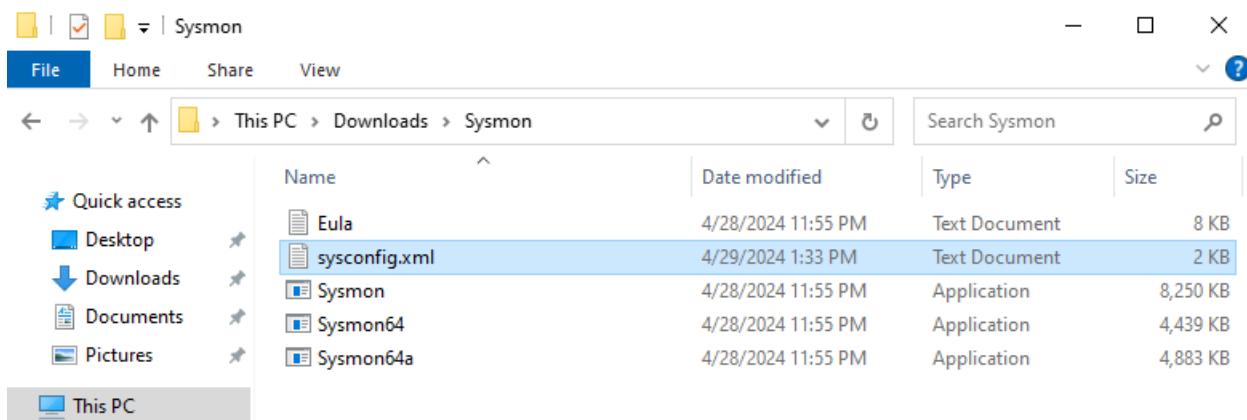
Time	Technique(s)	Tactic(s)	Description	Level	Rule ID
Apr 28, 2024 @ 18:30:43.260			File "/tmp/yara/malware/xbash" is a positive match. Yara rule: MAL_Xbash_PY_Sep18_RID2038	12	108001
Apr 28, 2024 @ 18:30:43.219			File "/tmp/yara/malware/xbash" is a positive match. Yara rule: MAL_Xbash_PY_Sep18_RID2038	12	108001
Apr 28, 2024 @ 18:30:41.668			File modified in /tmp/yara/malware/ directory.	7	100300
Apr 28, 2024 @ 18:30:41.219			File "/tmp/yara/malware/xbash" is a positive match. Yara rule: MAL_Xbash_PY_Sep18_RID2038	12	108001
Apr 28, 2024 @ 18:30:39.257			File "/tmp/yara/malware/xbash" is a positive match. Yara rule: MAL_Xbash_PY_Sep18_RID2038	12	108001
Apr 28, 2024 @ 18:30:39.215			File "/tmp/yara/malware/xbash" is a positive match. Yara rule: MAL_Xbash_PY_Sep18_RID2038	12	108001
Apr 28, 2024 @ 18:30:38.594			File modified in /tmp/yara/malware/ directory.	7	100300
Apr 28, 2024 @ 18:30:38.371			File modified in /tmp/yara/malware/ directory.	7	100300
Apr 28, 2024 @ 18:30:38.352			File modified in /tmp/yara/malware/ directory.	7	100300
Apr 28, 2024 @ 18:30:38.169			File modified in /tmp/yara/malware/ directory.	7	100300

4.4 Detect threats on Windows by monitoring Sysmon events

Sysmon is a **Microsoft** endpoint log collection tool which record system activity and detect malicious activity in these event logs. In this lab we will integrate this service with our **Wazuh**, and monitor some anomalies through defined channels on **Sysmon**.

Original article: <https://wazuh.com/blog/learn-to-detect-threats-on-windows-by-monitoring-sysmon-events/>

We can download **Sysmon** files from **Microsoft official webpage**. After downlaoding it, unzip the compressed file, go into the folder and create the **sysconfig.xml** file for **Sysmon** configuration. Open the file and paste following configuration from original article.





```
*sysconfig.xml - Notepad
File Edit Format View Help
<!--SYSMON EVENT ID 6 : DRIVER LOADED INTO KERNEL-->
<DriverLoad onmatch="include" />
<!--SYSMON EVENT ID 7 : DLL (IMAGE) LOADED BY PROCESS-->
<ImageLoad onmatch="include" />
<!--SYSMON EVENT ID 8 : REMOTE THREAD CREATED-->
<CreateRemoteThread onmatch="include">
    <SourceImage condition="contains">mimikatz.exe</SourceImage>
</CreateRemoteThread>
<!--SYSMON EVENT ID 9 : RAW DISK ACCESS-->
<RawAccessRead onmatch="include" />
<!--SYSMON EVENT ID 10 : INTER-PROCESS ACCESS-->
<ProcessAccess onmatch="include">
    <SourceImage condition="contains">mimikatz.exe</SourceImage>
</ProcessAccess>
<!--SYSMON EVENT ID 11 : FILE CREATED-->
<FileCreate onmatch="include" />
<!--SYSMON EVENT ID 12 & 13 & 14 : REGISTRY MODIFICATION-->
<RegistryEvent onmatch="include" />
<!--SYSMON EVENT ID 15 : ALTERNATE DATA STREAM CREATED-->
<FileCreateStreamHash onmatch="include" />
<PipeEvent onmatch="include" />
</EventFiltering>
</Sysmon>
```

Later open **Power Shell** as an administrator and go into the downloaded folder and run ".\Sysmon64.exe -accepteula -i .\sysconfig.xml" command to install **Sysmon** tool into your **Windows** machine.

```
Administrator: Windows PowerShell
PS C:\Users\Administrator> cd .\Downloads\Sysmon\
PS C:\Users\Administrator\Downloads\Sysmon> dir

Directory: C:\Users\Administrator\Downloads\Sysmon

Mode                LastWriteTime         Length Name
----                -----          ---- -  
-a---        4/28/2024 11:55 PM           7490 Eula.txt
-a---        4/29/2024 1:33 PM          1804 sysconfig.xml.txt
-a---        4/28/2024 11:55 PM        8447792 Sysmon.exe
-a---        4/28/2024 11:55 PM        4545344 Sysmon64.exe
-a---        4/28/2024 11:55 PM        4999984 Sysmon64a.exe

PS C:\Users\Administrator\Downloads\Sysmon> .\Sysmon64.exe -accepteula -i .\sysconfig.xml.txt

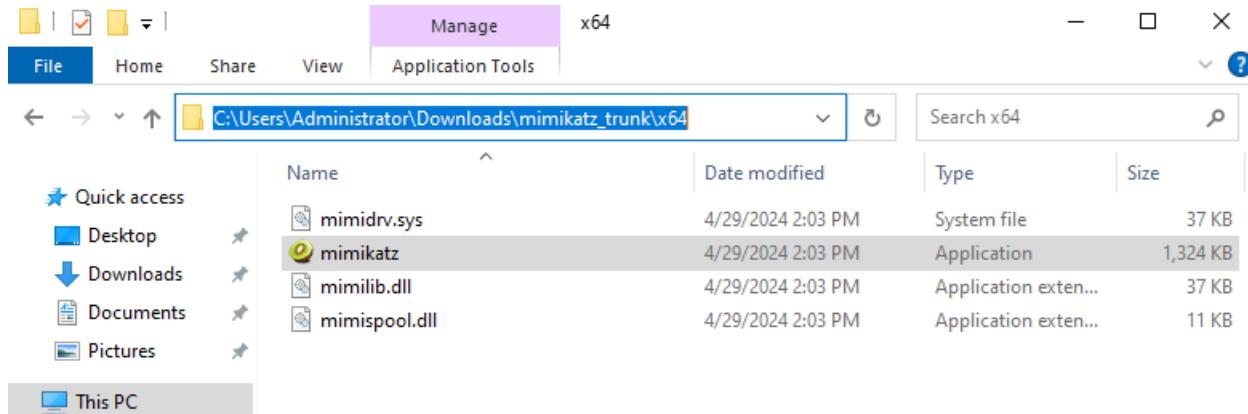
System Monitor v15.14 - System activity monitor
By Mark Russinovich and Thomas Garnier
Copyright (C) 2014-2024 Microsoft Corporation
Using libxml2. libxml2 is Copyright (C) 1998-2012 Daniel Veillard. All Rights Reserved.
Sysinternals - www.sysinternals.com

Loading configuration file with schema version 4.10
Sysmon schema version: 4.90
Configuration file validated.
Sysmon64 installed.
SysmonDrv installed.
Starting SysmonDrv.
SysmonDrv started.
Starting Sysmon64..
Sysmon64 started.
PS C:\Users\Administrator\Downloads\Sysmon>
```



Our **Sysmon** service running properly and we are ready to test it. For this purpose we will use a post exploitation tool **Mimikatz** which dumps password hashes of users, **PINs**, **Kerberos** tickets etc. Download the compressed file from followin link and decomress it.

Mimikatz tool: <https://github.com/gentilkiwi/mimikatz/releases/>



Later open your **Power Shell** as an administrator and change directory into the donwloaded **Mimikatz** folder and run the ".\mimikatz.exe" command to start **Mimikatz** tool.

```
mimikatz 2.2.0 x64 (oe.eo)
PS C:\Users\Administrator> cd .\Downloads\mimikatz_trunk\x64\
PS C:\Users\Administrator\Downloads\mimikatz_trunk\x64> dir

Directory: C:\Users\Administrator\Downloads\mimikatz_trunk\x64

Mode                LastWriteTime         Length Name
----                -              -          -
-a---  4/29/2024 2:03 PM      37208 mimidrv.sys
-a---  4/29/2024 2:03 PM    1355264 mimikatz.exe
-a---  4/29/2024 2:03 PM      37376 mimilib.dll
-a---  4/29/2024 2:03 PM      10752 mimispool.dll

PS C:\Users\Administrator\Downloads\mimikatz_trunk\x64> .\mimikatz.exe

.#####. mimikatz 2.2.0 (x64) #19041 Sep 19 2022 17:44:08
.## ^ ##. "A La Vie, A L'Amour" - (oe.eo)
## / \ ## /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
## \ / ## > https://blog.gentilkiwi.com/mimikatz
'## v ##' Vincent LE TOUX ( vincent.letoux@gmail.com )
'#####' > https://pingcastle.com / https://mysmartlogon.com ***/
mimikatz #
```

Once we run **Mimikatz** tool, the **Sysmon** tool will generate an informational logs about this action. This is a kind of anomalies because **Mimikatz** is well knows cyber attack tool and its signature easily can be detected by our monitoring tool, **Sysmon**. In order to analyse event logs, open “**Event Viewer**” on your **Windows** agent machine and go to the “**Applications and Service log/Microsoft/Windows/Sysmon**” path in it and enter into the **Operational** section to see the generated logs. For the detailed information double click on the log, then you will see additional information about **Mimikatz** starting event.



Event Viewer

File Action View Help

Windows

Name
StorageSpaces-Driver
StorageSpaces-ManagementAgent
StorageSpaces-Parser
StorageSpaces-SpaceManager
StorDiag
Store
StorPort
Storsvc
Sysmon
SystemDataArchiver
SystemSettingsThreshold
TaskScheduler
TCP/IP
TerminalServices-ClientActiveXCore

Actions

- Windows
 - Open Saved Log...
 - Create Custom View...
 - Import Custom View...
 - View
 - Refresh
 - Help
- Sysmon
 - Open
 - Help

Event Viewer

File Action View Help

Sysmon

Name	Type	Number of Events	Size
Operational	Operational	3	68 KB

Actions

- Sysmon
 - Open Saved Log...
 - Create Custom View...
 - Import Custom View...
 - View
 - Refresh
 - Help
- Operational
 - Open
 - Properties
 - Help

Event Viewer

File Action View Help

Operational Number of events: 3

Level	Date and Time	Source	Event ID	Type
Information	4/29/2024 2:15:49 PM	Sysmon	1	P
Information	4/29/2024 2:14:57 PM	Sysmon	4	S
Information	4/29/2024 2:14:57 PM	Sysmon	16	S

Actions

- Operational
 - Open Saved Log...
 - Create Custom View...
 - Import Custom View...
 - Clear Log...
 - Filter Current Log...
 - Properties
 - Disable Log
 - Find...
 - Save All Events As...
 - Attach a Task To this Log
 - View
 - Refresh
 - Help
- Event 1, Sysmon
 - Event Properties
 - Attach Task To This Log



We monitored one of the anomalies related to **Mimikatz** on Windows agent machine, but the purpose of this lab is to observe events logs on our **SIEM**. As you remember, any integration needs a configuration change on **Wazuh** agent and server. Open **Windows** agent's configuration file and add following configuration block in it. You can do it by opening “**Wazuh Agent Manager**” tool and enter “**View Config**” option or open directly configuration file, **ossec.conf**, under **C:\Program Files (x86)\ossec-agent** path.

```

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

</ossec_config>

<!-- END of Default Configuration. --&gt;
</pre>

```

After the configuration changes, restart your **Wazuh** agent service to apply.

Now we are done with agent configuration, go to the **Wazuh** server and open **local_rules.xml** file to add our custom rules to detect **Mimikatz** events and restart **wazuh-manager.service** to perform new rules.

```
[root@wazuh ~]# vim /var/ossec/etc/rules/local_rules.xml
```



```
<group name="windows, sysmon, sysmon_process-anomalies, ">
<rule id="100010" level="12">
<if_group>sysmon_event1</if_group>
<field name="win.eventdata.image">mimikatz.exe</field>
<description>Sysmon - Suspicious Process - mimikatz.exe</description>
</rule>

<rule id="100011" level="12">
<if_group>sysmon_event8</if_group>
<field name="win.eventdata.sourceImage">mimikatz.exe</field>
<description>Sysmon - Suspicious Process mimikatz.exe created a remote thread</description>
</rule>

<rule id="100012" level="12">
<if_group>sysmon_event_10</if_group>
<field name="win.eventdata.sourceImage">mimikatz.exe</field>
<description>Sysmon - Suspicious Process mimikatz.exe accessed $(win.eventdata.targetImage)</description>
</rule>
</group>
"/var/ossec/etc/rules/local_rules.xml" 132L, 4341C written
116,18
```

[root@wazuh ~]# systemctl restart wazuh-manager.service

Again go back to the **Windows** agent and open **Power Shell** as an administrator, then run the following command ".\mimikatz.exe" in the downloaded **Mimikatz** folder. Once the program started run "lsadump::lsa /inject" command to dump users password hashes.

```
mimikatz 2.2.0 x64 (oe.eo)
PS C:\Users\Administrator\Downloads\mimikatz_trunk\x64> .\mimikatz.exe

.####. mimikatz 2.2.0 (x64) #19041 Sep 19 2022 17:44:08
.## ^ ##. "A La Vie, A L'Amour" - (oe.eo)
## / \ ## /** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
## \ / ## > https://blog.gentilkiwi.com/mimikatz
'## v ##' Vincent LE TOUX ( vincent.letoux@gmail.com )
'#####' > https://pingcastle.com / https://mysmartlogon.com **/


mimikatz # lsadump::lsa /inject
Domain : CS301 / S-1-5-21-1612885287-2075756089-633124396

RID : 000001f4 (500)
User : Administrator

* Primary
NTLM : 15877a8a3ca26c04d5b00516a38623f8
```

In order to see the generated event logs, open "**Event Viewer**" in **Windows** machine and go to the **Sysmon** logs.

Level	Date and Time	Source	Event ID	Task Category
Information	4/29/2024 3:18:43 PM	Sysmon	8	CreateRemoteThread detected (rule: CreateRemoteThread)
Information	4/29/2024 3:18:42 PM	Sysmon	8	CreateRemoteThread detected (rule: CreateRemoteThread)
Information	4/29/2024 3:18:41 PM	Sysmon	8	CreateRemoteThread detected (rule: CreateRemoteThread)
Information	4/29/2024 3:18:40 PM	Sysmon	8	CreateRemoteThread detected (rule: CreateRemoteThread)
Information	4/29/2024 3:18:39 PM	Sysmon	8	CreateRemoteThread detected (rule: CreateRemoteThread)
Information	4/29/2024 3:18:39 PM	Sysmon	8	CreateRemoteThread detected (rule: CreateRemoteThread)
Information	4/29/2024 3:18:38 PM	Sysmon	8	CreateRemoteThread detected (rule: CreateRemoteThread)
Information	4/29/2024 3:18:38 PM	Sysmon	8	CreateRemoteThread detected (rule: CreateRemoteThread)
Information	4/29/2024 3:18:37 PM	Sysmon	8	CreateRemoteThread detected (rule: CreateRemoteThread)
Information	4/29/2024 3:18:37 PM	Sysmon	10	Process accessed (rule: ProcessAccess)
Information	4/29/2024 3:18:02 PM	Sysmon	1	Process Create (rule: ProcessCreate)
Information	4/29/2024 3:10:45 PM	Sysmon	8	CreateRemoteThread detected (rule: CreateRemoteThread)
Information	4/29/2024 3:10:44 PM	Sysmon	8	CreateRemoteThread detected (rule: CreateRemoteThread)
Information	4/29/2024 3:10:44 PM	Sysmon	8	CreateRemoteThread detected (rule: CreateRemoteThread)

Event 8, Sysmon

General Details

CreateRemoteThread detected:
RuleName: -
UtcTime: 2024-04-29 11:18:43.106

Log Name: Microsoft-Windows-Sysmon/Operational
Source: Sysmon
Event ID: 8
Level: Information
User: SYSTEM
Logged: 4/29/2024 3:18:43 PM
Task Category: CreateRemoteThread detected (rule: CreateRemoteThread)
Keywords:
Computer: DC.cs301.local

Actions

- Operational
- Open Saved Log...
- Create Custom View...
- Import Custom View...
- Clear Log...
- Filter Current Log...
- Properties
- Disable Log
- Find...
- Save All Events As...
- Attach a Task To This Log...
- View
- Refresh
- Help

Event 8, Sysmon

- Event Properties
- Attach Task To This Event...
- Copy
- Save Selected Events...
- Refresh
- Help



Now we can monitor the logs in our **SIEM**, go to the **Wazuh GUI** in your browser, select the attack performed **Windows** agent and open “**Security events**” to see generated logs. For detailed information, just click on the any choosen log and compare it with the **Sysmon** logs in **Windows** agent itself.

wazuh. ▾						Modules	DC	Security events ⓘ	
>	Apr 29, 2024 @ 15:18:38.757	T1055	Defense Evasion, Privilege Escalation	Local Security Authority Subsystem Service (LSASS) process was accessed by C:\Users\ Administrator Downloads mimikatz_trunk x64 mimikatz.exe, possible code injection for credential dumping	12	92403		credential dumping	a
>	Apr 29, 2024 @ 15:18:42.543	T1055	Defense Evasion, Privilege Escalation	Local Security Authority Subsystem Service (LSASS) process was accessed by C:\Users\ Administrator Downloads mimikatz_trunk x64 mimikatz.exe, possible code injection for credential dumping	12	92403			
>	Apr 29, 2024 @ 15:18:38.685			Sysmon - Suspicious Process mimikatz.exe accessed C:\Windows\system32\lsass.exe	12	100012			
>	Apr 29, 2024 @ 15:18:40.201	T1055	Defense Evasion, Privilege Escalation	Local Security Authority Subsystem Service (LSASS) process was accessed by C:\Users\ Administrator Downloads mimikatz_trunk x64 mimikatz.exe, possible code injection for credential dumping	12	92403			
>	Apr 29, 2024 @ 15:18:41.294	T1055	Defense Evasion, Privilege Escalation	Local Security Authority Subsystem Service (LSASS) process was accessed by C:\Users\ Administrator Downloads mimikatz_trunk x64 mimikatz.exe, possible code injection for credential dumping	12	92403			
>	Apr 29, 2024 @ 15:18:42.547	T1055	Defense Evasion, Privilege Escalation	Local Security Authority Subsystem Service (LSASS) process was accessed by C:\Users\ Administrator Downloads mimikatz_trunk x64 mimikatz.exe, possible code injection for credential dumping	12	92403			
>	Apr 29, 2024 @ 15:18:02.828			Sysmon - Suspicious Process - mimikatz.exe	12	100010			
>	Apr 29, 2024 @ 15:09:42.109			Sysmon - Suspicious Process - mimikatz.exe	12	100010			

wazuh. ▾						Modules	DC	Security events ⓘ			
▼ Apr 29, 2024 @ 15:18:38.685						Sysmon - Suspicious Process mimikatz.exe accessed C:\Windows\system32\lsass.exe				12	100012
Table	JSON	Rule									
@timestamp	2024-04-29T11:18:38.685Z										
_id	a4uUKYBbwvVywEhhnDpNj										
agent.id	007										
agent.ip	172.16.10.10										
agent.name	DC										
data.win.eventdata.callTrace	C:\Windows\SYSTEM32\ntdll.dll+9f3b4[C:\Windows\System32\KERNELBASE.dll+2aafe C:\Users\Administrator Downloads mimikatz_trunk x64 mimikatz.exe+a2a3d C:\Users\Administrator Downloads mimikatz_trunk x64 mimikatz.exe+a25c7 C:\Users\Administrator Downloads mimikatz_trunk x64 mimikatz.exe+85a44 C:\Users\Administrator Downloads mimikatz_trunk x64 mimikatz.exe+8587 C:\Users\Administrator Downloads mimikatz_trunk x64 mimikatz.exe+85647 C:\Users\Administrator Downloads mimikatz_trunk x64 mimikatz.exe+c97a5 C:\Windows\System32\KERNEL32.DLL+14e0 C:\Windows\SYSTEM32\ntdll.dll+e39b										
data.win.eventdata.grantedAccess	0x143a										
data.win.eventdata.sourceImage	C:\Users\Administrator Downloads mimikatz_trunk x64 mimikatz.exe										
data.win.eventdata.sourceProcessGUID	(36c3da46-81ea-662f-cb02-000000002500)										
data.win.eventdata.sourceProcessId	1224										
data.win.eventdata.sourceThreadId	5724										
data.win.eventdata.sourceUser	CS301\Administrator										
data.win.eventdata.targetImage	C:\Windows\System32\lsass.exe										
data.win.eventdata.targetProcessGUID	(36c3da46-6021-662f-cb02-000000002500)										
data.win.eventdata.targetProcessId	760										
data.win.eventdata.targetUser	NT AUTHORITY\SYSTEM										
data.win.eventdata.utcTime	2024-04-29 11:18:37.918										
data.win.system.channel	Microsoft-Windows-Sysmon/Operational										
data.win.system.computer	DC.cs301.local										

As a result, we performed an attack amulation with **Mimikatz** in our local machine where the **Wazuh** agent is installed and redirected **Sysmon** logs from this agent to our **Wazuh** server in order to analyse and compare the content of logs with our defined custom rules. Once the our rules were matched with the **Mimikatz** signature, **Wazuh** displayed suspicious event alerts to us.