

Détection des intrusions



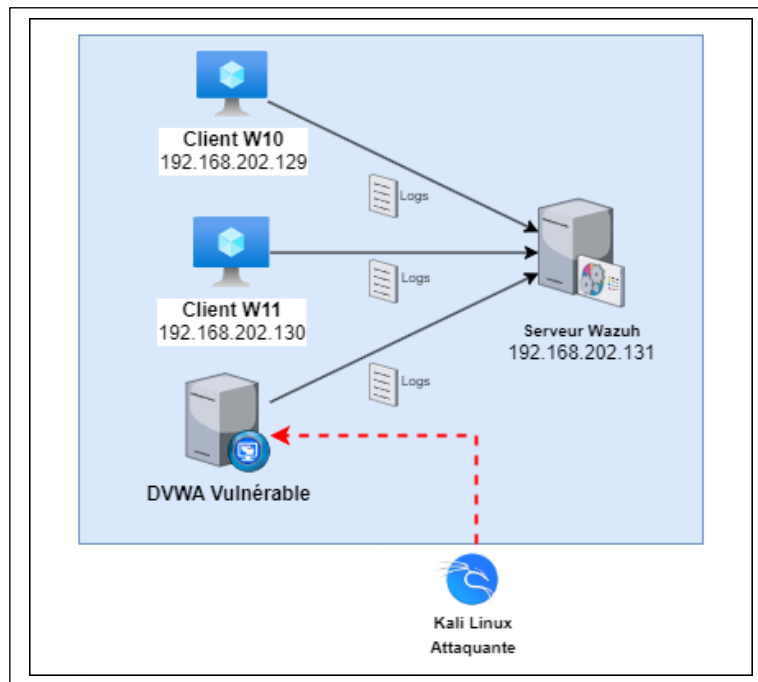
MBFA

Table des matières :

I.	Topologie Réseau – Présentation	2
	Installation et Configuration du Serveur Wazuh	2
II.	Installation de Wazuh	3
I.	Installation de l'indexer	3
I.	Installation de node :	4
II.	Configuration de l'indexer Wazuh :	4
III.	Deploying certificates	5
III.	Installation server	6
I.	Deploying certificates :	8
IV.	Installation Dashboard.....	9
I.	Configuring the Wazuh dashboard.....	9
II.	Deploying certificates :	9
V.	Lien Wazuh x Clients	11
VI.	Intégrer Windows Defender à Wazuh	13
VII.	Site Web vulnérable.....	15
VIII.	Attaque Brute force SSH	15
IX.	Détection / réponse à une attaque WEB.....	19
X.	Attaque sous Windows.....	24

I. Topologie Réseau – Présentation

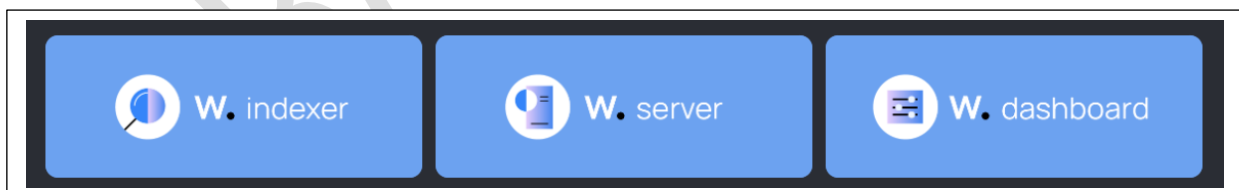
Pour ce TP voici la Topologie réseau qui a été mise en place :



(L'adresse IP de la machine DVWA change au bout d'un moment dans le TP)

II. Installation et Configuration du Serveur Wazuh

Wazuh est une plate-forme de sécurité qui offre une protection unifiée XDR et SIEM pour les points de terminaison et les charges de travail cloud. La solution est composée d'un seul agent universel et de trois composants centraux : le serveur Wazuh, l'indexeur Wazuh et le tableau de bord Wazuh.



- **Indexeur Wazuh** : moteur de recherche et d'analyse en texte intégral hautement évolutif. Ce composant central Wazuh indexe et stocke les alertes générées par le serveur Wazuh et fournit des capacités de recherche et d'analyse de données en temps quasi réel.
- **Wazuh Server** : Le serveur Wazuh analyse les données reçues des agents Wazuh, déclenchant des alertes lorsque des menaces ou des anomalies sont détectées. Il permet également de gérer à distance la configuration des agents et de surveiller leur statut.
- **Wazuh Dashboard** : Ce composant central est une interface Web flexible et intuitive pour extraire, analyser et visualiser les données de sécurité. Il fournit des tableaux de bord prêts à l'emploi, vous permettant de naviguer de manière transparente dans l'interface utilisateur.

II. Installation de Wazuh

I. Installation de l'indexer

Nous installons les différents packages nécessaires pour le fonctionnement de l'indexer :

```
curl -sO https://packages.wazuh.com/4.7/Wazuh-certs-tool.sh
```

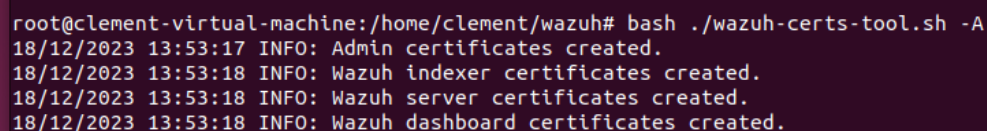
```
curl -sO https://packages.wazuh.com/4.7/config.yml
```

On modifie par la suite le fichier config.yml et ajoutons l'@IP du serveur Wazuh 192.168.202.131 sur l'indexer / server / dashboard. Notre serveur va héberger ces 3 infrastructures.



```
1 nodes:
2 # Wazuh indexer nodes
3 indexer:
4   - name: node-1
5     ip: "192.168.202.131"
6   #- name: node-2
7   # ip: "<indexer-node-ip>"
8   #- name: node-3
9   # ip: "<indexer-node-ip>"
10
11 # Wazuh server nodes
12 # If there is more than one Wazuh server
13 # node, each one must have a node_type
14 server:
15   - name: wazuh-1
16     ip: "192.168.202.131"
17   # node_type: master
18   #- name: wazuh-2
19   # ip: "<wazuh-manager-ip>"
20   # node_type: worker
21   #- name: wazuh-3
22   # ip: "<wazuh-manager-ip>"
23   # node_type: worker
24
25 # Wazuh dashboard nodes
26 dashboard:
27   - name: dashboard
28     ip: "192.168.202.131"
```

On crée les certificats essentiels au fonctionnement de Wazuh : `bash ./Wazuh-certs-tool.sh -A`



```
root@clement-virtual-machine:/home/clement/wazuh# bash ./wazuh-certs-tool.sh -A
18/12/2023 13:53:17 INFO: Admin certificates created.
18/12/2023 13:53:18 INFO: Wazuh indexer certificates created.
18/12/2023 13:53:18 INFO: Wazuh server certificates created.
18/12/2023 13:53:18 INFO: Wazuh dashboard certificates created.
```

On compresse les fichiers liés au certificat : `tar -cvf ./Wazuh-certificates.tar -C ./Wazuh-certificates/.`

I. Installation de node :

On va installer node indispensable pour le fonctionnement de l'indexer Wazuh :

```
apt-get install debconf adduser procps
```

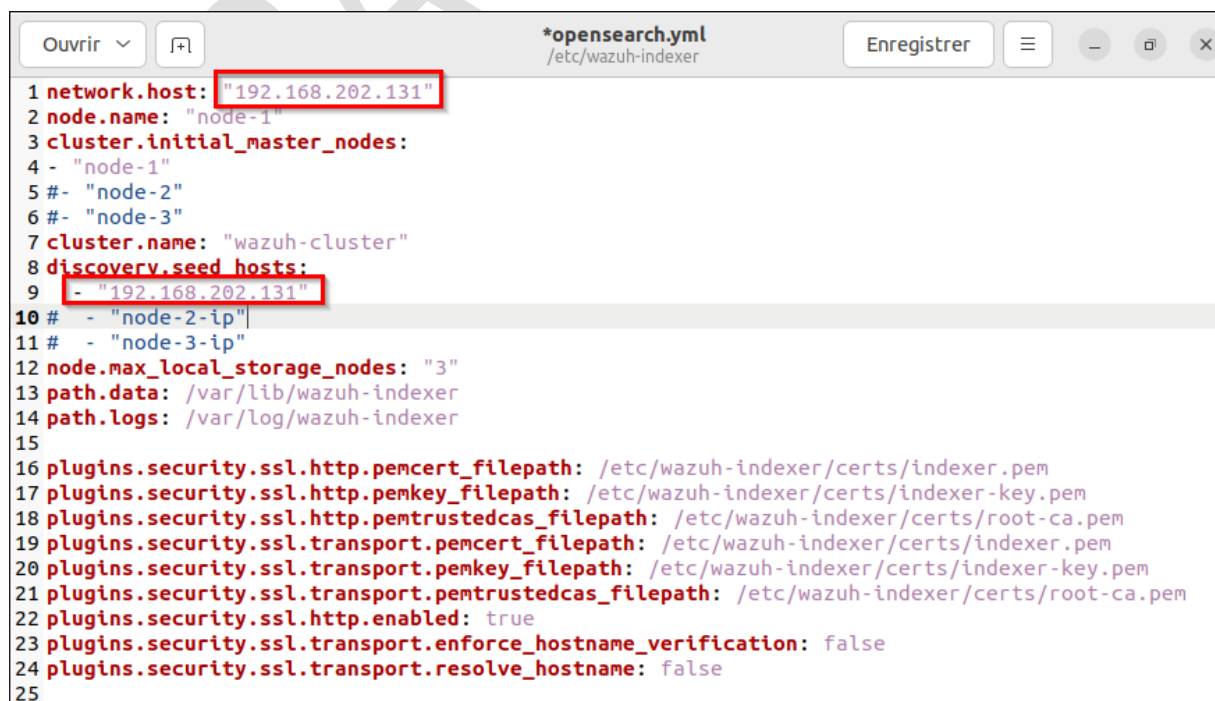
```
Installations des packages utiles : apt-get install gnupg apt-transport-https
```

```
Installation des clés GPO : curl -s https://packages.wazuh.com/key/GPG-KEY-WAZUH |  
gpg --no-default-keyring --keyring gnupg-ring:/usr/share/keyrings/Wazuh.gpg  
--import && chmod 644 /usr/share/keyrings/Wazuh.gpg
```

```
Nous ajoutons les différents dépôts : echo "deb [signed-  
by=/usr/share/keyrings/Wazuh.gpg] https://packages.wazuh.com/4.x/apt/  
stable main" | tee -a /etc/apt/sources.list.d/Wazuh.list
```

Finalement, installation du package indexer : `apt-get -y install Wazuh-indexer`

II. Configuration de l'indexer Wazuh :



```
Ouvrir v [+] *opensearch.yml /etc/wazuh-indexer Enregistrer [Menu] [Close] [Maximize] [Refresh]
1 network.host: "192.168.202.131"
2 node.name: "node-1"
3 cluster.initial_master_nodes:
4 - "node-1"
5 #- "node-2"
6 #- "node-3"
7 cluster.name: "wazuh-cluster"
8 discovery.seed_hosts:
9 - "192.168.202.131"
10 # - "node-2-ip"
11 # - "node-3-ip"
12 node.max_local_storage_nodes: "3"
13 path.data: /var/lib/wazuh-indexer
14 path.logs: /var/log/wazuh-indexer
15
16 plugins.security.ssl.http.pemcert_filepath: /etc/wazuh-indexer/certs/indexer.pem
17 plugins.security.ssl.http.pemkey_filepath: /etc/wazuh-indexer/certs/indexer-key.pem
18 plugins.security.ssl.http.pemtrustedcas_filepath: /etc/wazuh-indexer/certs/root-ca.pem
19 plugins.security.ssl.transport.pemcert_filepath: /etc/wazuh-indexer/certs/indexer.pem
20 plugins.security.ssl.transport.pemkey_filepath: /etc/wazuh-indexer/certs/indexer-key.pem
21 plugins.security.ssl.transport.pemtrustedcas_filepath: /etc/wazuh-indexer/certs/root-ca.pem
22 plugins.security.ssl.http.enabled: true
23 plugins.security.ssl.transport.enforce_hostname_verification: false
24 plugins.security.ssl.transport.resolve_hostname: false
25
```

Nous modifions le fichier indexer de Wazuh pour y rentrer l'adresse IP de notre serveur qui va servir à héberger celui-ci.

III. Deploying certificates

Nous allons déployer les certificats générés auparavant :

Commande à taper dans le terminal :

```
NODE_NAME=node-1
```

```
mkdir /etc/Wazuh-indexer/certs
```

```
tar -xf ./Wazuh-certificates.tar -C /etc/Wazuh-indexer/certs/  
./$NODE_NAME.pem ./$NODE_NAME-key.pem ./admin.pem ./admin-key.pem ./root-  
ca.pem
```

```
mv -n /etc/Wazuh-indexer/certs/$NODE_NAME.pem /etc/Wazuh-  
indexer/certs/indexer.pem
```

```
mv -n /etc/Wazuh-indexer/certs/$NODE_NAME-key.pem /etc/Wazuh-  
indexer/certs/indexer-key.pem
```

```
chmod 500 /etc/Wazuh-indexer/certs
```

```
chmod 400 /etc/Wazuh-indexer/certs/*
```

```
chown -R Wazuh-indexer:Wazuh-indexer /etc/Wazuh-indexer/certs
```

Puis nous pouvons démarrer le service :

```
systemctl daemon-reload
```

```
systemctl enable Wazuh-indexer
```

```
systemctl start Wazuh-indexer
```

```
root@clement-virtual-machine:~/home/clement/wazuh# systemctl status wazuh-indexer
● wazuh-indexer.service - Wazuh-Indexer
   Loaded: loaded (/lib/systemd/system/wazuh-indexer.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2023-12-18 14:10:17 CET; 35s ago
     Docs: https://documentation.wazuh.com
   Main PID: 4934 (java)
    Tasks: 75 (limit: 10585)
   Memory: 1.3G
      CPU: 40.743s
   CGroup: /system.slice/wazuh-indexer.service
           └─4934 /usr/share/wazuh-indexer/jdk/bin/java -Xshare:auto -Dopensearch.networkaddress.cache.ttl=60 -Dopensearch.networkaddress.cache.negative.ttl=10 -XX

déc. 18 14:10:06 clement-virtual-machine systemd[1]: Starting Wazuh-Indexer...
déc. 18 14:10:09 clement-virtual-machine systemd-entrypoint[4934]: WARNING: A terminally deprecated method in java.lang.System has been called
déc. 18 14:10:09 clement-virtual-machine systemd-entrypoint[4934]: WARNING: System::setSecurityManager has been called by org.opensearch.bootstrap.OpenSearch (file:/
déc. 18 14:10:09 clement-virtual-machine systemd-entrypoint[4934]: WARNING: Please consider reporting this to the maintainers of org.opensearch.bootstrap.OpenSearch
déc. 18 14:10:09 clement-virtual-machine systemd-entrypoint[4934]: WARNING: System::setSecurityManager will be removed in a future release
déc. 18 14:10:10 clement-virtual-machine systemd-entrypoint[4934]: WARNING: A terminally deprecated method in java.lang.System has been called
déc. 18 14:10:10 clement-virtual-machine systemd-entrypoint[4934]: WARNING: System::setSecurityManager has been called by org.opensearch.bootstrap.Security (file:/usr
déc. 18 14:10:10 clement-virtual-machine systemd-entrypoint[4934]: WARNING: Please consider reporting this to the maintainers of org.opensearch.bootstrap.Security
déc. 18 14:10:10 clement-virtual-machine systemd-entrypoint[4934]: WARNING: System::setSecurityManager will be removed in a future release
déc. 18 14:10:17 clement-virtual-machine systemd[1]: Started Wazuh-Indexer.
lines 1-21/23 (END)
```

L'indexer est fonctionnel et prêt à l'emploi !

III. Installation server

On installe le package du serveur : `apt-get -y install Wazuh-manager`

Puis on active le service :

```
systemctl daemon-reload
systemctl enable Wazuh-manager
systemctl start Wazuh-manager
```

Et on vérifie l'état du serveur :

```
systemctl status Wazuh-manager
```

```
root@clement-virtual-machine:/home/clement/wazuh# systemctl status wazuh-manager
● wazuh-manager.service - Wazuh manager
   Loaded: loaded (/lib/systemd/system/wazuh-manager.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2023-12-18 14:16:30 CET; 26s ago
     Process: 48519 ExecStart=/usr/bin/env /var/ossec/bin/wazuh-control start (code=exited, status=0/SUCCESS)
    Tasks: 219 (limit: 10585)
  Memory: 636.5M
     CPU: 28.642s
   CGroup: /system.slice/wazuh-manager.service
           └─48578 /var/ossec/framework/python/bin/python3 /var/ossec/api/scripts/wazuh-apid.py
             └─48619 /var/ossec/bin/wazuh-authd
               └─48668 /var/ossec/bin/wazuh-db
                 └─48692 /var/ossec/framework/python/bin/python3 /var/ossec/api/scripts/wazuh-apid.py
                   └─48695 /var/ossec/framework/python/bin/python3 /var/ossec/api/scripts/wazuh-apid.py
                     └─48698 /var/ossec/framework/python/bin/python3 /var/ossec/api/scripts/wazuh-apid.py
                       └─48739 /var/ossec/bin/wazuh-execd
                         └─48792 /var/ossec/bin/wazuh-analysisd
                           └─48957 /var/ossec/bin/wazuh-syscheckd
                             └─49001 /var/ossec/bin/wazuh-remoted
                               └─49064 /var/ossec/bin/wazuh-logcollector
                                 └─49083 /var/ossec/bin/wazuh-monitord
                                   └─49108 /var/ossec/bin/wazuh-modulesd

déc. 18 14:16:21 clement-virtual-machine env[48519]: Started wazuh-db...
déc. 18 14:16:22 clement-virtual-machine env[48519]: Started wazuh-execd...
déc. 18 14:16:23 clement-virtual-machine env[48519]: Started wazuh-analysisd...
déc. 18 14:16:24 clement-virtual-machine env[48519]: Started wazuh-syscheckd...
déc. 18 14:16:25 clement-virtual-machine env[48519]: Started wazuh-remoted...
déc. 18 14:16:26 clement-virtual-machine env[48519]: Started wazuh-logcollector...
déc. 18 14:16:27 clement-virtual-machine env[48519]: Started wazuh-monitord...
déc. 18 14:16:28 clement-virtual-machine env[48519]: Started wazuh-modulesd...
déc. 18 14:16:30 clement-virtual-machine env[48519]: Completed.
déc. 18 14:16:30 clement-virtual-machine systemd[1]: Started Wazuh manager.
root@clement-virtual-machine:/home/clement/wazuh#
```

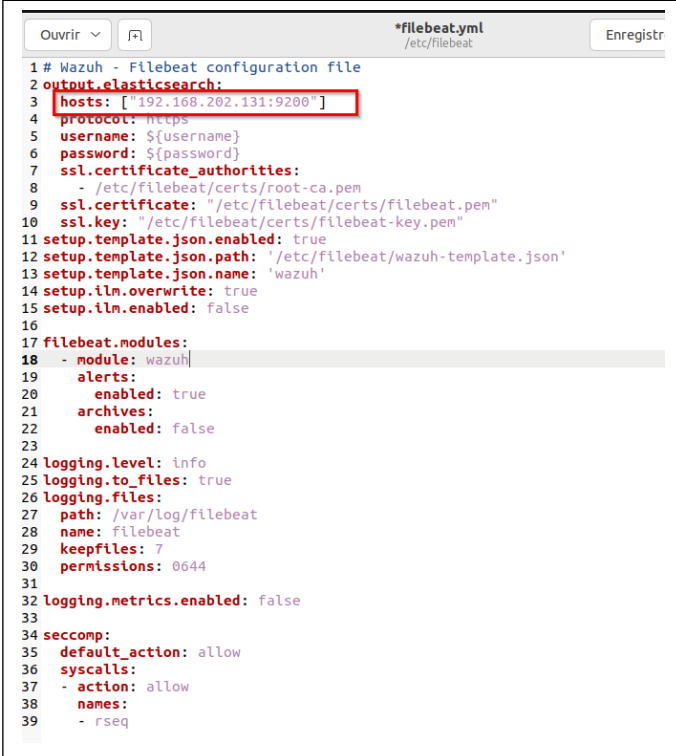
Le serveur est alors actif, nous devons le configurer

On installe ensuite filebeat : `apt-get -y install filebeat`

Configuration du filebeat:

```
curl -so /etc/filebeat/filebeat.yml
```

```
https://packages.wazuh.com/4.7/tp1/Wazuh/filebeat/filebeat.yml
```



```
1 # Wazuh - Filebeat configuration file
2 output.elasticsearch:
3   hosts: ["192.168.202.131:9200"]
4   protocol: https
5   username: ${username}
6   password: ${password}
7   ssl.certificate_authorities:
8     - /etc/filebeat/certs/root-ca.pem
9   ssl.certificate: "/etc/filebeat/certs/filebeat.pem"
10  ssl.key: "/etc/filebeat/certs/filebeat-key.pem"
11 setup.template.json.enabled: true
12 setup.template.json.path: '/etc/filebeat/wazuh-template.json'
13 setup.template.json.name: 'wazuh'
14 setup.ilm.overwrite: true
15 setup.ilm.enabled: false
16
17 filebeat.modules:
18   - module: wazuh
19     alerts:
20       enabled: true
21     archives:
22       enabled: false
23
24 logging.level: info
25 logging.to_files: true
26 logging.files:
27   path: /var/log/filebeat
28   name: filebeat
29   keepfiles: 7
30   permissions: 0644
31
32 logging.metrics.enabled: false
33
34 seccomp:
35   default_action: allow
36   syscalls:
37     - action: allow
38     names:
39       - rseq
```

On crée un Filebeat « Keystore » pour sécuriser le stockage de nos différentes clés d'authentification :

```
filebeat keystore create
```

Add the default username and password admin:admin to the secrets keystore :

```
echo admin | filebeat keystore add username --stdin --force
```

```
echo admin | filebeat keystore add password --stdin --force
```

Téléchargement de l'alerte template pour le Wazuh indexer :

```
curl -so /etc/filebeat/Wazuh-template.json
```

```
https://raw.githubusercontent.com/Wazuh/Wazuh/v4.7.0/extensions/elasticsearch/7.x/Wazuh-template.json
```

```
chmod go+r /etc/filebeat/Wazuh-template.json
```


Installation du module FileBeat pour Wazuh :

```
curl -s https://packages.wazuh.com/4.x/filebeat/Wazuh-filebeat-0.3.tar.gz |  
tar -xvz -C /usr/share/filebeat/module
```

I. Deploying certificates :

Dans le terminal nous tapons les commandes suivantes :

```
NODE_NAME=node-1  
  
mkdir /etc/filebeat/certs  
  
tar -xf ./Wazuh-certificates.tar -C /etc/filebeat/certs/ ./${NODE_NAME}.pem  
./${NODE_NAME}-key.pem ./root-ca.pem  
  
mv -n /etc/filebeat/certs/${NODE_NAME}.pem /etc/filebeat/certs/filebeat.pem  
mv -n /etc/filebeat/certs/${NODE_NAME}-key.pem /etc/filebeat/certs/filebeat-  
key.pem  
  
chmod 500 /etc/filebeat/certs  
chmod 400 /etc/filebeat/certs/*  
chown -R root:root /etc/filebeat/certs
```

Puis nous redémarrons le service filebeat :

```
systemctl daemon-reload  
systemctl enable filebeat  
systemctl start filebeat
```

On peut voir que filebeat est bien installé sur le serveur :

```
root@clement-virtual-machine:/home/clement/wazuh# filebeat test output  
elasticsearch: https://192.168.202.131:9200...  
parse url... OK  
connection...  
parse host... OK  
dns lookup... OK  
addresses: 192.168.202.131  
dial up... OK  
TLS...  
security: server's certificate chain verification is enabled  
handshake... OK  
TLS version: TLSv1.3  
dial up... OK  
talk to server... ERROR 503 Service Unavailable: OpenSearch Security not initialized.
```

On voit une erreur 503 Service Unavailable, j'exécute alors le script pour lancer le serveur :
`/usr/share/Wazuh-indexer/bin/indexer-security-init.sh`

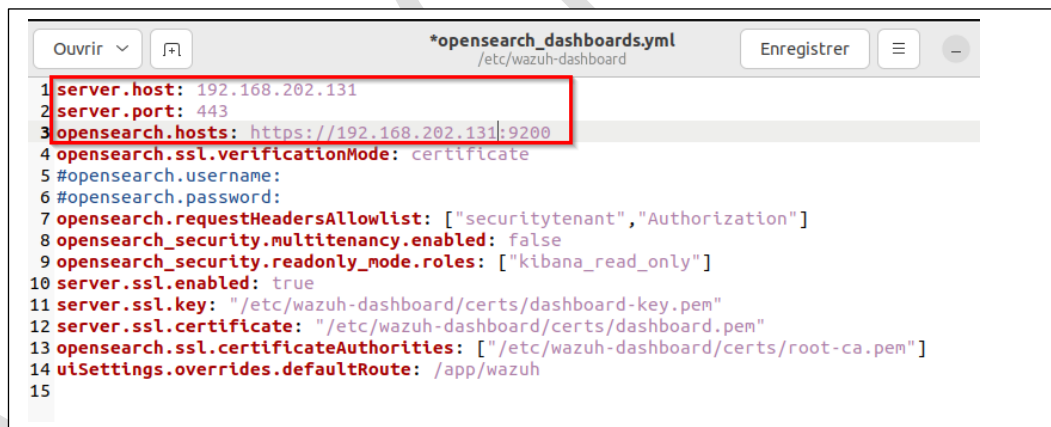
```
root@clement-virtual-machine:/home/clement/wazuh# filebeat test output
elasticsearch: https://192.168.202.131:9200...
parse url... OK
connection...
parse host... OK
dns lookup... OK
addresses: 192.168.202.131
dial up... OK
TLS...
security: server's certificate chain verification is enabled
handshake... OK
TLS version: TLSv1.3
dial up... OK
talk to server... OK
version: 7.10.2
```

Cette-fois ci tout est au vert, tout est bon dans notre installation de l'indexer / Wazuh serveur.

IV. Installation Dashboard

Nous installons le package: `apt-get -y install Wazuh-dashboard`

I. Configuring the Wazuh dashboard



```
Ouvrir  [icon] *opensearch_dashboards.yml  Enregistrer  [icon]  -
/etc/wazuh-dashboard
1 server.host: 192.168.202.131
2 server.port: 443
3 opensearch.hosts: https://192.168.202.131:9200
4 opensearch.ssl.verificationMode: certificate
5 #opensearch.username:
6 #opensearch.password:
7 opensearch.requestHeadersAllowlist: ["securitytenant", "Authorization"]
8 opensearch_security.multitenancy.enabled: false
9 opensearch_security.readonly_mode.roles: ["kibana_read_only"]
10 server.ssl.enabled: true
11 server.ssl.key: "/etc/wazuh-dashboard/certs/dashboard-key.pem"
12 server.ssl.certificate: "/etc/wazuh-dashboard/certs/dashboard.pem"
13 opensearch.ssl.certificateAuthorities: ["/etc/wazuh-dashboard/certs/root-ca.pem"]
14 uiSettings.overrides.defaultRoute: /app/wazuh
15
```

Nous mettons l'adresse IP de notre serveur hébergeant Wazuh dashboard.

II. Deploying certificates :

Dans le terminal nous tapons les commandes suivantes :

```
NODE_NAME=node-1
```

```
mkdir /etc/filebeat/certs
```

```
tar -xf ./Wazuh-certificates.tar -C /etc/filebeat/certs/ ./${NODE_NAME}.pem
./${NODE_NAME}-key.pem ./root-ca.pem

mv -n /etc/filebeat/certs/${NODE_NAME}.pem /etc/filebeat/certs/filebeat.pem
mv -n /etc/filebeat/certs/${NODE_NAME}-key.pem /etc/filebeat/certs/filebeat-
key.pem

chmod 500 /etc/filebeat/certs
chmod 400 /etc/filebeat/certs/*
chown -R root:root /etc/filebeat/certs
```

Puis je redémarre le service :

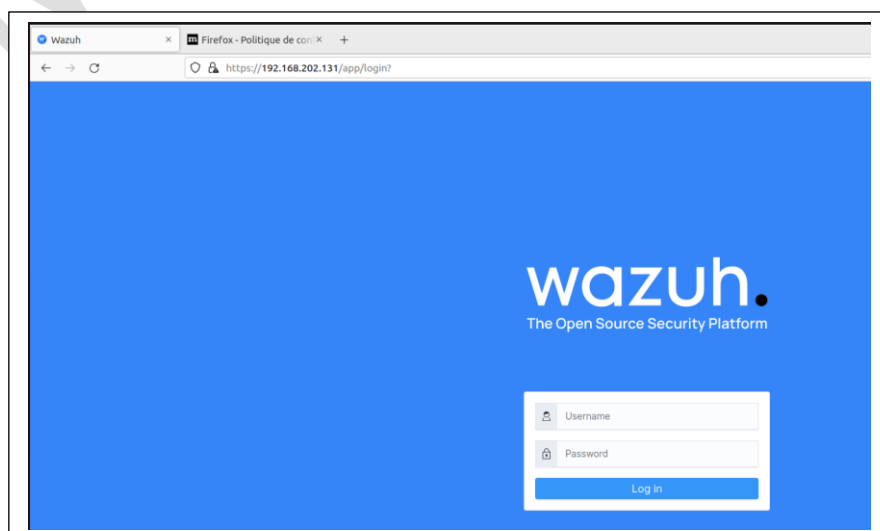
```
systemctl daemon-reload
systemctl enable Wazuh-dashboard
systemctl start Wazuh-dashboard
```

```
root@clement-virtual-machine:/home/clement/wazuh# systemctl status wazuh-dashboard
● wazuh-dashboard.service - wazuh-dashboard
   Loaded: loaded (/etc/systemd/system/wazuh-dashboard.service; enabled; vendor preset: enabl
   Active: active (running) since Mon 2023-12-18 14:33:53 CET; 25s ago
     Main PID: 52696 (node)
       Tasks: 11 (limit: 10585)
      Memory: 233.3M
         CPU: 4.947s
    CGroup: /system.slice/wazuh-dashboard.service
            └─52696 /usr/share/wazuh-dashboard/node/bin/node --no-warnings --max-http-header-s

déc. 18 14:33:56 clement-virtual-machine opensearch-dashboards[52696]: {"type": "log", "@timestan
déc. 18 14:33:56 clement-virtual-machine opensearch-dashboards[52696]: {"type": "log", "@timestan
déc. 18 14:33:56 clement-virtual-machine opensearch-dashboards[52696]: {"type": "log", "@timestan
déc. 18 14:33:56 clement-virtual-machine opensearch-dashboards[52696]: {"type": "log", "@timestan
déc. 18 14:33:56 clement-virtual-machine opensearch-dashboards[52696]: {"type": "log", "@timestan
déc. 18 14:33:57 clement-virtual-machine opensearch-dashboards[52696]: {"type": "log", "@timestan
déc. 18 14:33:57 clement-virtual-machine opensearch-dashboards[52696]: {"type": "log", "@timestan
déc. 18 14:33:57 clement-virtual-machine opensearch-dashboards[52696]: {"type": "log", "@timestan
déc. 18 14:33:57 clement-virtual-machine opensearch-dashboards[52696]: {"type": "log", "@timestan
lines 1-20/20 (END)
```

Wazuh Dashboard est alors prêt à l'emploi !

Nous accédons à l'interface WEB de Wazuh pour confirmer :



V. Lien Wazuh x Clients

Nous installons le client Wazuh sur notre poste client W10 et W11. Depuis la page Wazuh agent nous générons la commande utile :

```
PS C:\Windows\system32> Invoke-WebRequest -Uri https://packages.wazuh.com/4.x/windows/wazuh-agent-4.7.0-1.msi -OutFile {env:tmp}\wazuh-agent; msexec.exe /i ${env:tmp}\wazuh-agent /q WAZUH_MANAGER='192.168.202.131' WAZUH_REGISTRATION_SERVER='192.168.202.131'
PS C:\Windows\system32>
PS C:\Windows\system32> NET START WazuhSvc
Le service Wazuh démarre.
Le service Wazuh a démarré.
```

On retrouve bien nos 2 clients Windows sur notre Wazuh :

ID	Name	IP address	Group(s)	Operating system	Cluster node	Version	Status	Actions
001	DESKTOP-USQ829D	192.168.202.129	default	Microsoft Windows 10 Enterprise Evaluation 10.0.19045.2006	node01	v4.7.0	active	🔍 🔄
002	PC-ClientW11	192.168.202.130	default	Microsoft Windows 11 Pro 10.0.22000.318	node01	v4.7.0	active	🔍 🔄

Nous rajoutons maintenant le client Debian :

```
wget https://packages.wazuh.com/4.x/apt/pool/main/w/Wazuh-agent/Wazuh-agent_4.7.0-1_amd64.deb && sudo WAZUH_MANAGER='192.168.202.131' WAZUH_AGENT_NAME='Debian' dpkg -i ./Wazuh-agent_4.7.0-1_amd64.deb
```

Et nous démarrons le service sur le serveur :

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

```

wazuh-agent.service - Wazuh agent
Loaded: loaded (/lib/systemd/system/wazuh-agent.service; enabled; preset: enabled)
Active: active (running) since Mon 2023-12-18 15:58:54 CET; 13s ago
Process: 5139 ExecStart=/usr/bin/env /var/ossec/bin/wazuh-control start (code=exited, status=0/SUCCESS)
Tasks: 26 (limit: 2244)
Memory: 37.2M
CPU: 429ms
CGroup: /system.slice/wazuh-agent.service
├─5163 /var/ossec/bin/wazuh-execd
├─5175 /var/ossec/bin/wazuh-agentd
├─5188 /var/ossec/bin/wazuh-syscheckd
├─5202 /var/ossec/bin/wazuh-logcollector
└─5219 /var/ossec/bin/wazuh-modulesd

déc. 18 15:58:47 debian-poste1 systemd[1]: Starting wazuh-agent.service - Wazuh agent...
déc. 18 15:58:47 debian-poste1 env[5139]: Starting Wazuh v4.7.0...
déc. 18 15:58:48 debian-poste1 env[5139]: Started wazuh-execd...
déc. 18 15:58:49 debian-poste1 env[5139]: Started wazuh-agentd...
déc. 18 15:58:50 debian-poste1 env[5139]: Started wazuh-syscheckd...
déc. 18 15:58:51 debian-poste1 env[5139]: Started wazuh-logcollector...
déc. 18 15:58:52 debian-poste1 env[5139]: Started wazuh-modulesd...
déc. 18 15:58:54 debian-poste1 env[5139]: Completed.
déc. 18 15:58:54 debian-poste1 systemd[1]: Started wazuh-agent.service - Wazuh agent.
~
    
```

Sur Wazuh nous retrouvons bien nos 3 clients :

The screenshot shows the Wazuh dashboard interface. At the top, there's a 'STATUS' section with a circular progress indicator showing 3 active agents. Below this, a 'DETAILS' section shows 'Active: 3', 'Disconnected: 0', 'Pending: 0', and 'Never connected: 0'. An 'EVOLUTION' graph shows the number of active agents over time. The main part of the dashboard is a table listing the agents:

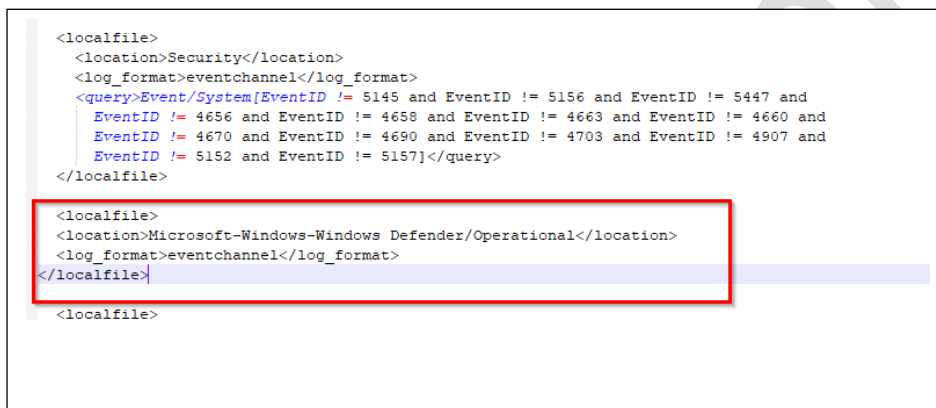
ID	Name	IP address	Group(s)	Operating system	Cluster node	Version	Status	Actions
001	DESKTOP-USGB28D	192.168.202.129	default	Microsoft Windows 10 Enterprise Evaluation 10.0.19045.2006	node01	v4.7.0	active	[Refresh] [Export]
002	PC-ClientW11	192.168.202.130	default	Microsoft Windows 11 Pro 10.0.22000.318	node01	v4.7.0	active	[Refresh] [Export]
003	Debian	192.168.202.132	default	Debian GNU/Linux 12	node01	v4.7.0	active	[Refresh] [Export]

VI. Intégrer Windows Defender à Wazuh

Sur nos clients Windows nous modifions le fichier : C:\Program Files (x86)\ossec-agent\ossec.conf

Et nous rajoutons le block suivant :

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



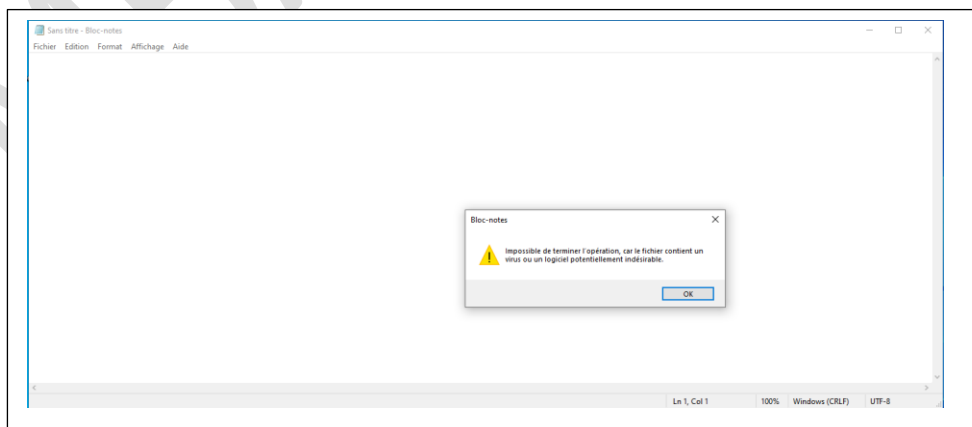
```
<localfile>
  <location>Security</location>
  <log_format>eventchannel</log_format>
  <query>Event/System[EventID != 5145 and EventID != 5156 and EventID != 5447 and
  EventID != 4656 and EventID != 4658 and EventID != 4663 and EventID != 4660 and
  EventID != 4670 and EventID != 4690 and EventID != 4703 and EventID != 4907 and
  EventID != 5152 and EventID != 5157]</query>
</localfile>

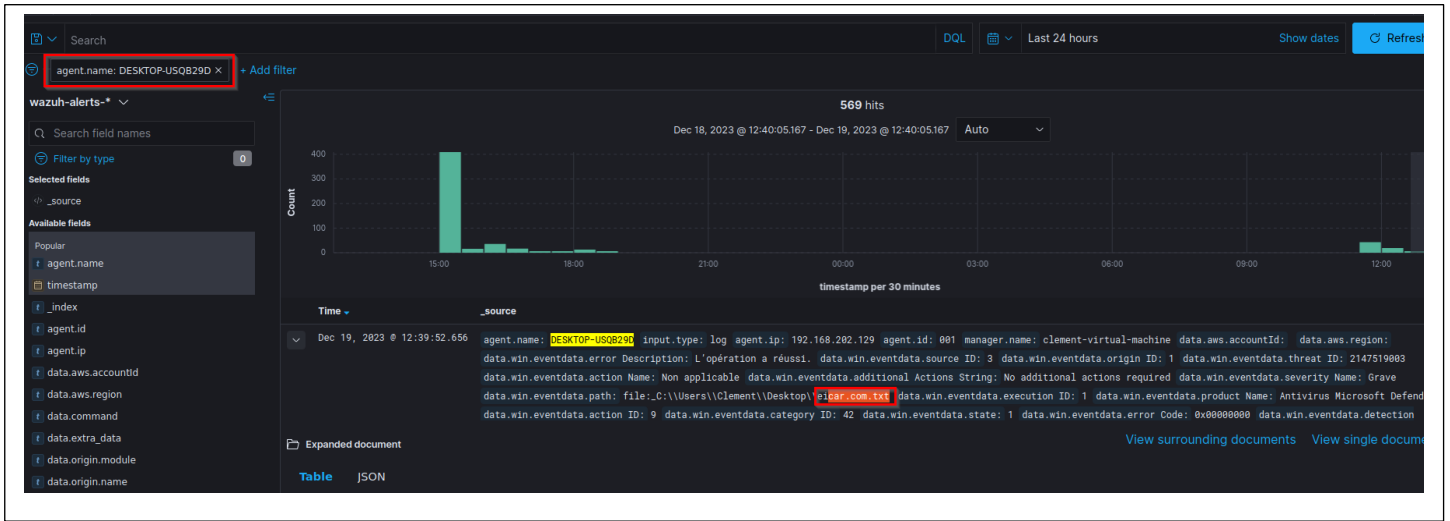
<localfile>
  <location>Microsoft-Windows-Windows Defender/Operational</location>
  <log_format>eventchannel</log_format>
</localfile>

<localfile>
```

Nous faisons la manipulation sur les 2 clients (W10 et W11).

Nous vérifions la bonne remontée des logs sur Wazuh grâce à un fichier eicar qui va trigger Windows Defender :





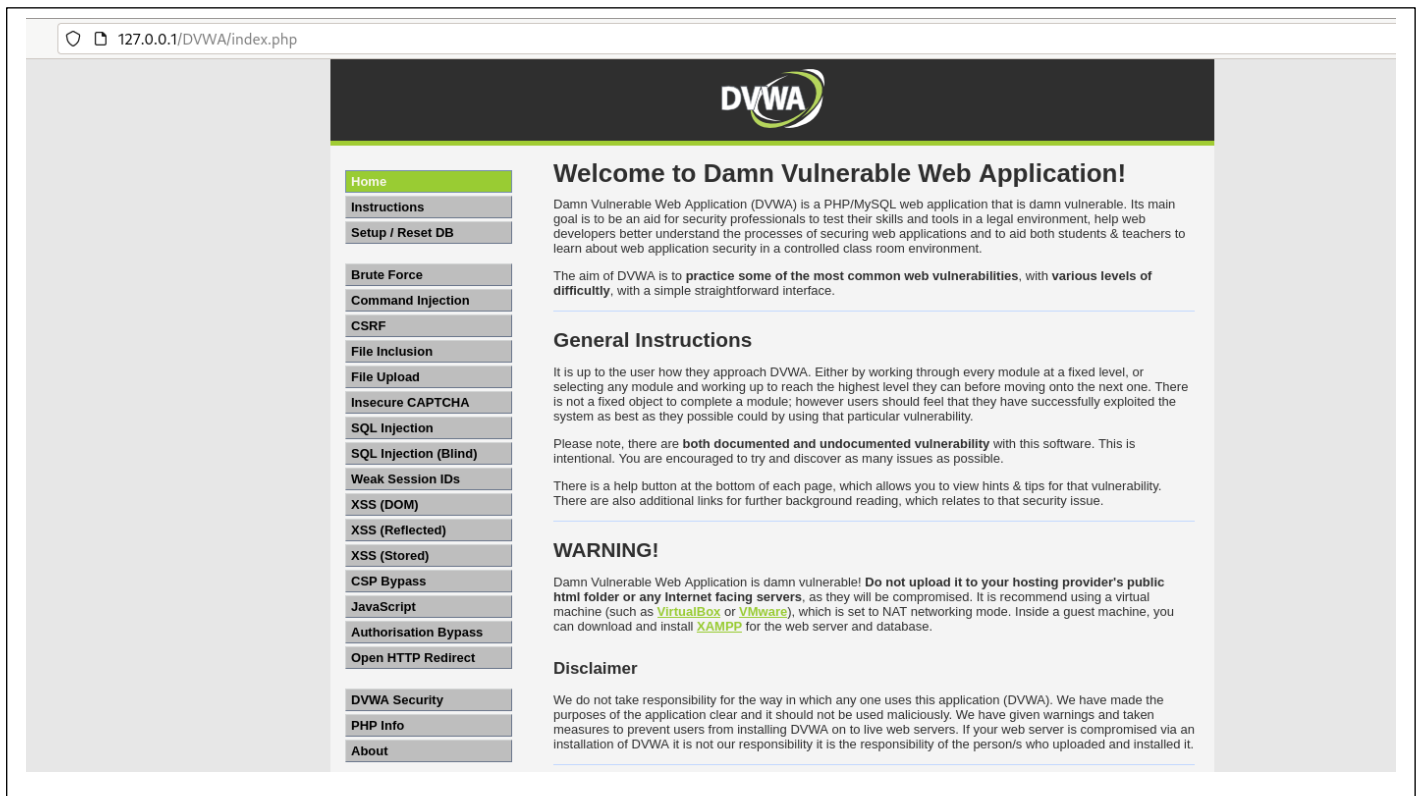
<code>data.win.eventdata.category ID</code>	42
<code>data.win.eventdata.category Name</code>	Virus
<code>data.win.eventdata.detection ID</code>	{E4069591-B0FD-4BE6-953B-127AF88149ED}
<code>data.win.eventdata.detection Time</code>	2023-12-19T11:39:50.165Z
<code>data.win.eventdata.detection User</code>	DESKTOP-USQB29D\\Clement
<code>data.win.eventdata.engine Version</code>	AM: 1.1.23110.2, NIS: 1.1.23110.2
<code>data.win.eventdata.error Code</code>	0x00000000
<code>data.win.eventdata.error Description</code>	L'opération a réussi.
<code>data.win.eventdata.execution ID</code>	1
<code>data.win.eventdata.execution Name</code>	Suspendu
<code>data.win.eventdata.fwLink</code>	https://go.microsoft.com/fwlink/?linkid=37020&name=Virus:DOS/EICAR_Test_File&threatid=2147519003&enterprise=0
<code>data.win.eventdata.origin ID</code>	1
<code>data.win.eventdata.origin Name</code>	Ordinateur local
<code>data.win.eventdata.path</code>	file:C:\\Users\\Clement\\Desktop\\eicar.com.txt

L'alerte WD est bien remontée sur Wazuh.

VII. Site Web vulnérable

Pour ce TP nous avons besoin d'un site WEB vulnérable, nous allons utiliser DVWA :
<https://github.com/digininja/DVWA>

Guide utilisé pour monter DVWA : <https://nooblinux.com/how-to-install-dvwa/>



The screenshot shows the DVWA homepage. The browser address bar displays '127.0.0.1/DVWA/index.php'. The page has a dark header with the DVWA logo. A left sidebar contains a menu of vulnerability categories, with 'Home' highlighted. The main content area is titled 'Welcome to Damn Vulnerable Web Application!' and contains several sections: a general introduction, 'General Instructions' explaining the application's purpose and usage, a 'WARNING!' section advising against public exposure, and a 'Disclaimer' regarding liability.

VIII. Attaque Brute force SSH

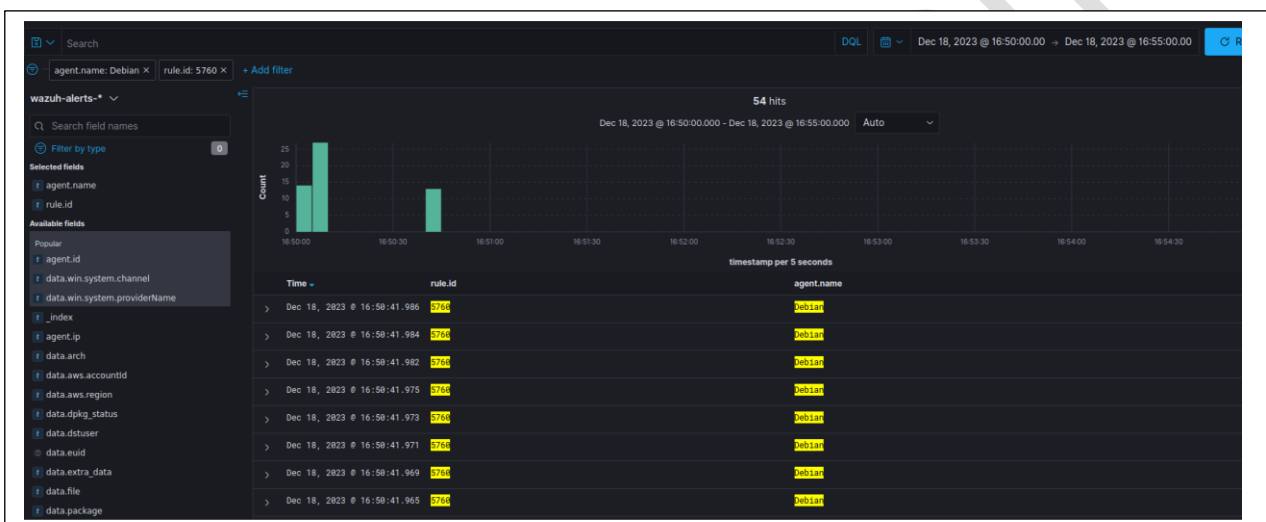
Pour cette partie on va détecter une attaque brute force ssh sur notre serveur debian et la stopper grâce à Wazuh.

Via notre Kali nous avons créé un fichier passwd.txt contenant 200 mots de passe dont 1 qui est celui de notre utilisateur Debian.

Grâce à l'outil Hydra nous lançons notre attaque et nous trouvons bien le mot de passe ssh :

```
(root@kali) ~ - [kali/Desktop]
# hydra -l debian1 -P /home/kali/Desktop/passwd.txt 192.168.202.132 ssh
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-12-18 10:49:51
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[DATA] max 16 tasks per 1 server, overall 16 tasks, 208 login tries (l:1/p:208), ~13 tries per task
[DATA] attacking ssh://192.168.202.132:22/
[22][ssh] host: 192.168.202.132 login: debian1 password: Pa$$word03
1 of 1 target successfully completed, 1 valid password found
[WARNING] Writing restore file because 1 final worker threads did not complete until end.
[ERROR] 1 target did not resolve or could not be connected
[ERROR] 0 target did not complete
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2023-12-18 10:50:42
```

Sur Wazuh nous trouvons bien les logs ssh de la tentative de brute force (plugin ID 5760) :



Nous allons maintenant faire une règle Wazuh pour bloquer ce type d'attaque :

Doc : <https://documentation.wazuh.com/current/user-manual/capabilities/active-response/ar-use-cases/blocking-ssh-brute-force.html>

Nous vérifions si dans le fichier /var/ossec/etc/ossec.conf si le block « Firewall-drop » est existant :

```
Ouvrir  ossec.conf
/var/ossec/etc

257 <name>disable-account</name>
258 <executable>disable-account</executable>
259 <timeout_allowed>yes</timeout_allowed>
260 </command>
261
262 <command>
263 <name>restart-wazuh</name>
264 <executable>restart-wazuh</executable>
265 </command>
266
267 <command>
268 <name>firewall-drop</name>
269 <executable>firewall-drop</executable>
270 <timeout_allowed>yes</timeout_allowed>
271 </command>
272
```


Nous relançons alors notre attaque SSH :

Nous pouvons voir que notre kali peut bien communiquer avec la machine Debian :

```
(root@kali)-[~/kali]
└─# ping 192.168.202.132
PING 192.168.202.132 (192.168.202.132) 56(84) bytes of data.
64 bytes from 192.168.202.132: icmp_seq=1 ttl=64 time=0.367 ms
64 bytes from 192.168.202.132: icmp_seq=2 ttl=64 time=0.566 ms
64 bytes from 192.168.202.132: icmp_seq=3 ttl=64 time=0.415 ms
64 bytes from 192.168.202.132: icmp_seq=4 ttl=64 time=0.501 ms
64 bytes from 192.168.202.132: icmp_seq=5 ttl=64 time=0.433 ms
64 bytes from 192.168.202.132: icmp_seq=6 ttl=64 time=0.416 ms
```

Nous lançons notre attaque (17h11) celle-ci est bien détectée par Wazuh :

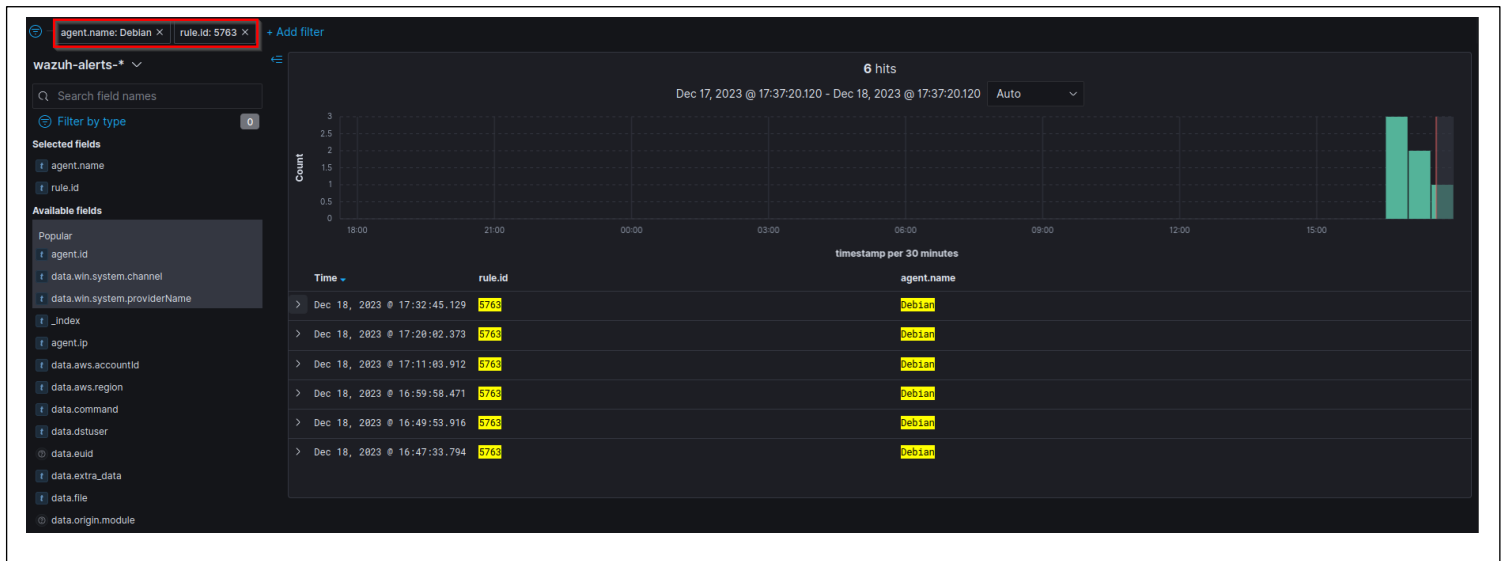
The screenshot shows the Wazuh dashboard interface. At the top, there are search filters for 'agent.name: Debian' and 'rule.id: 5760'. The main area displays a graph with 214 hits and a table of alerts. The table has columns for 'Time', 'rule.id', and 'agent.name'. The alerts listed are:

Time	rule.id	agent.name
Dec 18, 2023 @ 17:11:03.925	5760	Debian
Dec 18, 2023 @ 17:11:03.923	5760	Debian
Dec 18, 2023 @ 17:11:03.923	5760	Debian
Dec 18, 2023 @ 17:11:03.919	5760	Debian
Dec 18, 2023 @ 17:11:03.917	5760	Debian
Dec 18, 2023 @ 17:11:03.914	5760	Debian
Dec 18, 2023 @ 17:11:03.910	5760	Debian

Notre attaque n'aboutira pas Wazuh a bloqué l'IP de la machine d'attaque. Le ping ne fonctionne plus pendant une durée de 3 minutes (180 secondes mis dans le scénario de la réponse active).

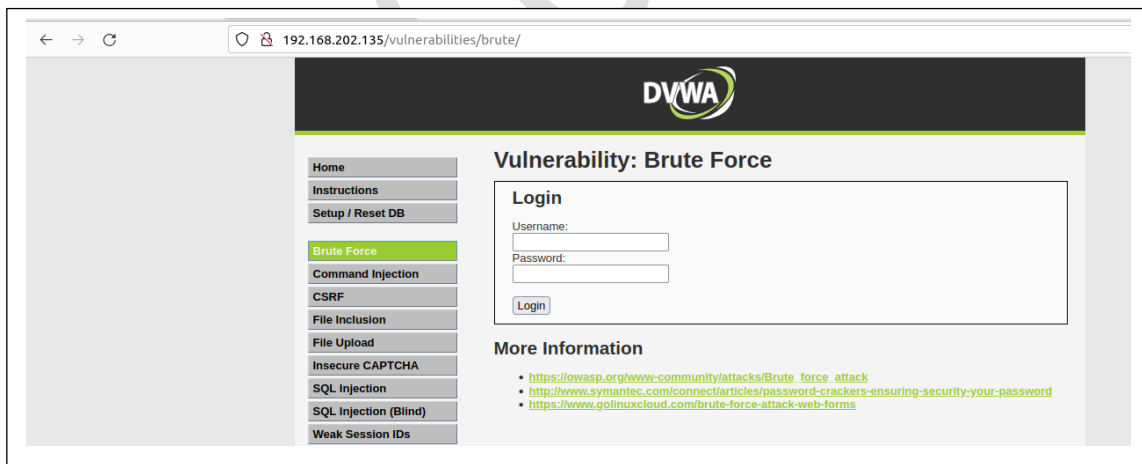
```
(root@kali)-[~/kali]
└─# ping 192.168.202.132
PING 192.168.202.132 (192.168.202.132) 56(84) bytes of data.
|
```

Et si nous filtrons sur Wazuh sur l'ID de notre réponse active (id : 5763) nous voyons bien qu'elle s'est activée X fois dès lorsqu'il a détecté une attaque SSH :



IX. Détection / réponse à une attaque WEB

Pour rappel nous avons monté un site web vulnérable.



Voyons déjà si les access.log sur le serveur apache fonctionne bien. Depuis a machine kali 192.168.202.133 nous allons accéder au site DVWA

```

192.168.202.133 - [19/Dec/2023:09:44:50 +0100] GET /vulnerabilities/upload/ HTTP/1.1" 302 528 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0"
192.168.202.133 - [19/Dec/2023:09:44:50 +0100] GET /login.php HTTP/1.1" 200 976 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0"
192.168.202.133 - [19/Dec/2023:09:44:50 +0100] GET /dwa/css/login.css HTTP/1.1" 200 741 "http://192.168.202.135/login.php" "Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0"
192.168.202.133 - [19/Dec/2023:09:44:50 +0100] GET /dwa/images/login_logo.png HTTP/1.1" 200 9374 "http://192.168.202.135/login.php" "Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0"
192.168.202.133 - [19/Dec/2023:09:44:50 +0100] GET /favicon.ico HTTP/1.1" 200 1706 "http://192.168.202.135/login.php" "Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0"
192.168.202.133 - [19/Dec/2023:09:44:50 +0100] POST /login.php HTTP/1.1" 200 337 "http://192.168.202.135/login.php" "Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0"
192.168.202.133 - [19/Dec/2023:09:44:55 +0100] GET /index.php HTTP/1.1" 200 2700 "http://192.168.202.135/login.php" "Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0"
192.168.202.133 - [19/Dec/2023:09:44:55 +0100] GET /dwa/js/dwaPage.js HTTP/1.1" 200 816 "http://192.168.202.135/index.php" "Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0"
192.168.202.133 - [19/Dec/2023:09:44:55 +0100] GET /dwa/js/add_event_listeners.js HTTP/1.1" 200 626 "http://192.168.202.135/index.php" "Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0"
192.168.202.133 - [19/Dec/2023:09:44:55 +0100] GET /dwa/css/main.css HTTP/1.1" 200 1445 "http://192.168.202.135/index.php" "Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0"
192.168.202.133 - [19/Dec/2023:09:44:55 +0100] GET /dwa/images/logo.png HTTP/1.1" 200 5330 "http://192.168.202.135/index.php" "Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0"
root@debian: /var/log/apache2#
    
```

Nous devons maintenant remonter les logs apache sur Wazuh. Pour cela, nous devons modifier le fichier `/var/ossec/etc/ossec.conf` du client Wazuh et nous rajoutons le bloc suivant :

```
<localfile>

<log_format>syslog</log_format>

<location>/var/log/apache2/access.log</location>

</localfile>
```

```
194 <localfile>
195 <log_format>syslog</log_format>
196 <location>/var/log/apache2/access.log</location>
197 </localfile>
198
199
```

Puis nous redémarons l'agent :

```
root@debian:/var/ossec/logs# sudo /var/ossec/bin/ossec-control restart
sudo: /var/ossec/bin/ossec-control : commande introuvable
root@debian:/var/ossec/logs# sudo systemctl start wazuh-agent
root@debian:/var/ossec/logs# sudo systemctl restart wazuh-agent
root@debian:/var/ossec/logs# sudo systemctl status wazuh-agent
● wazuh-agent.service - Wazuh agent
   Loaded: loaded (/lib/systemd/system/wazuh-agent.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2023-12-19 10:31:57 CET; 7s ago
     Process: 5148 ExecStart=/usr/bin/env /var/ossec/bin/wazuh-control start (code=exited, status=0/SUCCESS)
    Tasks: 32 (limit: 2264)
   Memory: 369.1M
      CPU: 11.103s
   CGroup: /system.slice/wazuh-agent.service
           └─5171 /var/ossec/bin/wazuh-execd
             └─5182 /var/ossec/bin/wazuh-agentd
               └─5196 /var/ossec/bin/wazuh-syscheckd
                 └─5210 /var/ossec/bin/wazuh-logcollector
                   └─5227 /var/ossec/bin/wazuh-modulesd

déc. 19 10:31:50 debian systemd[1]: wazuh-agent.service: Consumed 30.347s CPU time.
déc. 19 10:31:50 debian systemd[1]: Starting Wazuh agent...
déc. 19 10:31:50 debian env[5148]: Starting Wazuh v4.7.0...
déc. 19 10:31:51 debian env[5148]: Started wazuh-execd...
déc. 19 10:31:52 debian env[5148]: Started wazuh-agentd...
déc. 19 10:31:53 debian env[5148]: Started wazuh-syscheckd...
déc. 19 10:31:54 debian env[5148]: Started wazuh-logcollector...
déc. 19 10:31:55 debian env[5148]: Started wazuh-modulesd...
déc. 19 10:31:57 debian env[5148]: Completed.
déc. 19 10:31:57 debian systemd[1]: Started Wazuh agent.
```

On va lancer un Fuzzing du site WEB pour détecter les potentielles pages cachées / accessibles :

Sur la kali: `gobuster dir -u http://192.168.202.135/ -w /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt`

```
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

[+] Url:          http://192.168.202.135/
[+] Method:       GET
[+] Threads:      10
[+] Wordlist:     /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt
[+] Negative Status codes: 404
[+] User Agent:   gobuster/3.6
[+] Timeout:     10s

Starting gobuster in directory enumeration mode

/docs                (Status: 301) [Size: 317] [→ http://192.168.202.135/d
ocs/]
/tests              (Status: 301) [Size: 318] [→ http://192.168.202.135/t
ests/]
/database           (Status: 301) [Size: 321] [→ http://192.168.202.135/d
atabase/]
/external           (Status: 301) [Size: 321] [→ http://192.168.202.135/e
xternal/]
/config             (Status: 301) [Size: 319] [→ http://192.168.202.135/c
onfig/]
/vulnerabilities    (Status: 301) [Size: 328] [→ http://192.168.202.135/v
ulnerabilities/]
Progress: 87664 / 87665 (100.00%)
Finished
```

Sur le serveur hébergeant le site WEB dans les logs apache nous trouvons bien les logs associés au Fuzzing :

```
192.168.202.133 - - [19/Dec/2023:10:38:06 +0100] "GET /001742 HTTP/1.1" 404 438 "-" "gobuster/3.6"
192.168.202.133 - - [19/Dec/2023:10:38:06 +0100] "GET /revelations HTTP/1.1" 404 438 "-" "gobuster/3.6"
192.168.202.133 - - [19/Dec/2023:10:38:06 +0100] "GET /mainsite HTTP/1.1" 404 438 "-" "gobuster/3.6"
192.168.202.133 - - [19/Dec/2023:10:38:06 +0100] "GET /000787 HTTP/1.1" 404 438 "-" "gobuster/3.6"
192.168.202.133 - - [19/Dec/2023:10:38:06 +0100] "GET /stratplan HTTP/1.1" 404 438 "-" "gobuster/3.6"
192.168.202.133 - - [19/Dec/2023:10:38:06 +0100] "GET /000935 HTTP/1.1" 404 438 "-" "gobuster/3.6"
192.168.202.133 - - [19/Dec/2023:10:38:06 +0100] "GET /000904 HTTP/1.1" 404 438 "-" "gobuster/3.6"
192.168.202.133 - - [19/Dec/2023:10:38:06 +0100] "GET /000936 HTTP/1.1" 404 438 "-" "gobuster/3.6"
192.168.202.133 - - [19/Dec/2023:10:38:06 +0100] "GET /000874 HTTP/1.1" 404 438 "-" "gobuster/3.6"
192.168.202.133 - - [19/Dec/2023:10:38:06 +0100] "GET /hics5 HTTP/1.1" 404 438 "-" "gobuster/3.6"
192.168.202.133 - - [19/Dec/2023:10:38:06 +0100] "GET /001508 HTTP/1.1" 404 438 "-" "gobuster/3.6"
192.168.202.133 - - [19/Dec/2023:10:38:06 +0100] "GET /001220 HTTP/1.1" 404 438 "-" "gobuster/3.6"
192.168.202.133 - - [19/Dec/2023:10:38:06 +0100] "GET /000888 HTTP/1.1" 404 438 "-" "gobuster/3.6"
192.168.202.133 - - [19/Dec/2023:10:38:06 +0100] "GET /000810 HTTP/1.1" 404 438 "-" "gobuster/3.6"
root@debian: /var/ncsser/1ncss#
```

Sur Wazuh nous retrouvons bien les différentes tentatives d'accès aux pages web du site :

The screenshot shows the Wazuh dashboard interface. At the top, there is a search bar with the query 'location: /var/log/apache2/access.log'. Below the search bar, there are filter options and a 'wazuh-alerts-*' dropdown. The main area displays a time-series graph with 9,843 hits, showing a spike in activity around 18:00 on Dec 19, 2023. Below the graph is a table of alerts with columns for 'Time' and 'agent.name'. The table shows multiple entries for 'Dec 19, 2023 @ 10:38:25,949 DVWA'.

Table JSON	
† _index	wazuh-alerts-4.x-2023.12.19
† agent.id	004
† agent.ip	192.168.202.135
† agent.name	DVWA
† data.aws.accountId	
† data.aws.region	
† data.id	404
† data.protocol	GET
† data.srcip	192.168.202.133
† data.url	/easycclipse
† decoder.name	web-accesslog
† full_log	192.168.202.133 - - [19/Dec/2023:10:38:13 +0100] "GET /easycclipse HTTP/1.1" 404 438 "-" "gobuster/3.6"
† id	1702978705.5140712
† input.type	log
† location	/var/log/apache2/access_log
† manager.name	clement-virtual-machine
† rule.description	Web server 400 error code.
† rule.firedtimes	9,119

L'ID sur Wazuh qui va nous permettre de bloquer ce type d'attaque est l'ID du plugin 31101. Nous allons alors appliquer une réponse active qui dit que dès que cet ID match X fois, cela détecte une potentielle attaque de type FUZZING sur le site web et cela bloque l'attaquant.

Dans le fichier `/var/ossec/etc/ossec.conf` nous rajoutons cette règle :

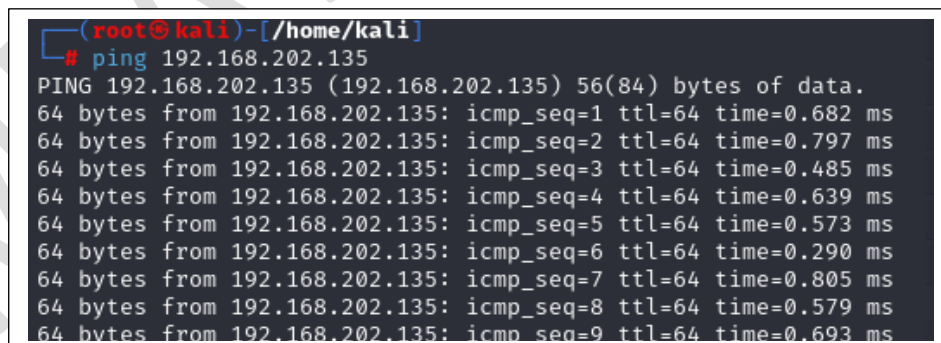
```
<ossec_config>
  <active-response>
    <command>firewall-drop</command>
    <location>local</location>
    <rules_id>31101</rules_id>
    <timeout>180</timeout>
  </active-response>
</ossec_config>
```



```
375 <ossec_config>
376   <active-response>
377     <command>firewall-drop</command>
378     <location>local</location>
379     <rules_id>5763</rules_id>
380     <rules_id>5760</rules_id>
381     <timeout>180</timeout>
382   </active-response>
383 </ossec_config>
384
385
386 <ossec_config>
387   <active-response>
388     <command>firewall-drop</command>
389     <location>local</location>
390     <rules_id>31101</rules_id>
391     <timeout>180</timeout>
392   </active-response>
393 </ossec_config>
394
```

Et nous redémarrons le serveur Wazuh pour appliquer le changement : `sudo systemctl restart Wazuh-manager`

Le ping entre la kali et le serveur hébergeant le site web fonctionne bien :



```
(root@kali)-[~/home/kali]
└─# ping 192.168.202.135
PING 192.168.202.135 (192.168.202.135) 56(84) bytes of data:
 64 bytes from 192.168.202.135: icmp_seq=1 ttl=64 time=0.682 ms
 64 bytes from 192.168.202.135: icmp_seq=2 ttl=64 time=0.797 ms
 64 bytes from 192.168.202.135: icmp_seq=3 ttl=64 time=0.485 ms
 64 bytes from 192.168.202.135: icmp_seq=4 ttl=64 time=0.639 ms
 64 bytes from 192.168.202.135: icmp_seq=5 ttl=64 time=0.573 ms
 64 bytes from 192.168.202.135: icmp_seq=6 ttl=64 time=0.290 ms
 64 bytes from 192.168.202.135: icmp_seq=7 ttl=64 time=0.805 ms
 64 bytes from 192.168.202.135: icmp_seq=8 ttl=64 time=0.579 ms
 64 bytes from 192.168.202.135: icmp_seq=9 ttl=64 time=0.693 ms
```

Nous relançons notre attaque :

```

gobuster dir -u http://192.168.202.135/ -w /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

[+] Url:          http://192.168.202.135/
[+] Method:       GET
[+] Threads:      10
[+] Wordlist:     /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt
[+] Negative Status codes: 404
[+] User Agent:   gobuster/3.6
[+] Timeout:     10s

Starting gobuster in directory enumeration mode

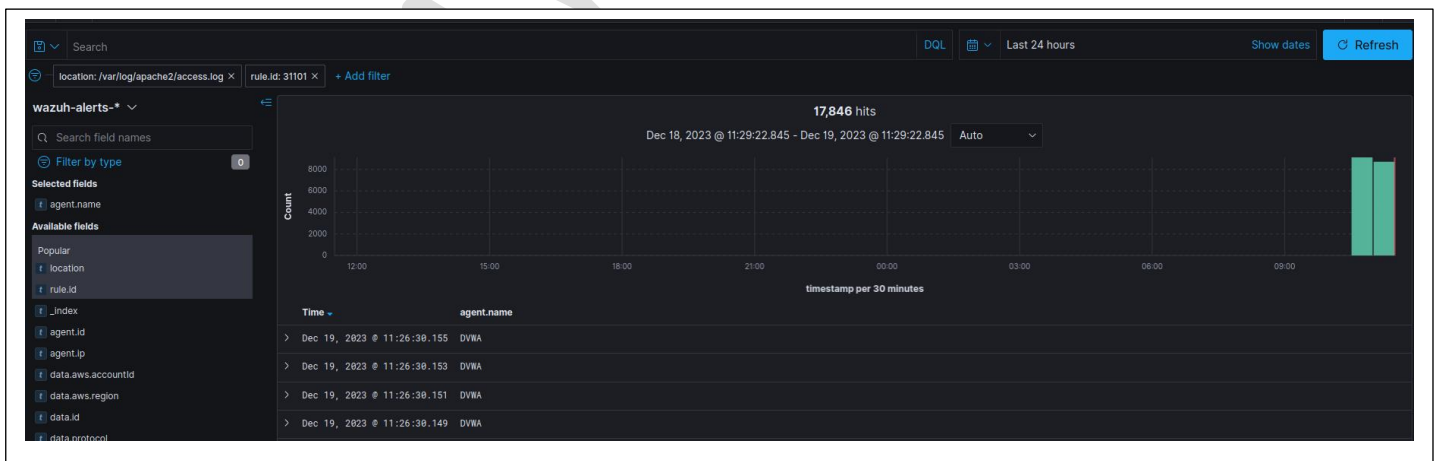
/docs          (Status: 301) [Size: 317] [→ http://192.168.202.135/docs/]
/tests         (Status: 301) [Size: 318] [→ http://192.168.202.135/tests/]
/database      (Status: 301) [Size: 321] [→ http://192.168.202.135/database/]
/external      (Status: 301) [Size: 321] [→ http://192.168.202.135/external/]
/config        (Status: 301) [Size: 319] [→ http://192.168.202.135/config/]
/vulnerabilities (Status: 301) [Size: 328] [→ http://192.168.202.135/vulnerabilities/]
Progress: 2950 / 87665 (3.37%) [ERROR] Get "http://192.168.202.135/ia": context deadline exceeded (Client.Timeout exceeded while awaiting headers)
[ERROR] Get "http://192.168.202.135/index_09": context deadline exceeded (Client.Timeout exceeded while awaiting headers)
[ERROR] Get "http://192.168.202.135/Servers": context deadline exceeded (Client.Timeout exceeded while awaiting headers)
[ERROR] Get "http://192.168.202.135/index_13": dial tcp 192.168.202.135:80: i/o timeout (Client.Timeout exceeded while awaiting headers)
[ERROR] Get "http://192.168.202.135/nero": context deadline exceeded (Client.Timeout exceeded while awaiting headers)
[ERROR] Get "http://192.168.202.135/l1": dial tcp 192.168.202.135:80: i/o timeout (Client.Timeout exceeded while awaiting headers)
[ERROR] Get "http://192.168.202.135/intranet": context deadline exceeded (Client.Timeout exceeded while awaiting headers)
[ERROR] Get "http://192.168.202.135/updated": dial tcp 192.168.202.135:80: i/o timeout (Client.Timeout exceeded while awaiting headers)
[ERROR] Get "http://192.168.202.135/passport": dial tcp 192.168.202.135:80: i/o timeout (Client.Timeout exceeded while awaiting headers)
[ERROR] Get "http://192.168.202.135/recruitment": context deadline exceeded (Client.Timeout exceeded while awaiting headers)
Progress: 2960 / 87665 (3.38%)
    
```

Au bout de 2 secondes, Wazuh a détecté l'attaque et a banni la source de celle-ci. Le ping entre la kali et la VM hébergeant le site ne fonctionne plus :

```

(root@kali)~/home/kali
# ping 192.168.202.135
PING 192.168.202.135 (192.168.202.135) 56(84) bytes of data.
    
```

L'attaque a alors été bloquée. Sur Wazuh nous retrouvons bien les traces de l'attaque :



3 minutes plus tard nous pouvons de nouveau pinguer le serveur depuis notre kali :

```

# ping 192.168.202.135
PING 192.168.202.135 (192.168.202.135) 56(84) bytes of data:
64 bytes from 192.168.202.135: icmp_seq=1 ttl=64 time=0.747 ms
64 bytes from 192.168.202.135: icmp_seq=2 ttl=64 time=0.541 ms
64 bytes from 192.168.202.135: icmp_seq=3 ttl=64 time=0.798 ms
64 bytes from 192.168.202.135: icmp_seq=4 ttl=64 time=0.589 ms
64 bytes from 192.168.202.135: icmp_seq=5 ttl=64 time=0.867 ms
    
```

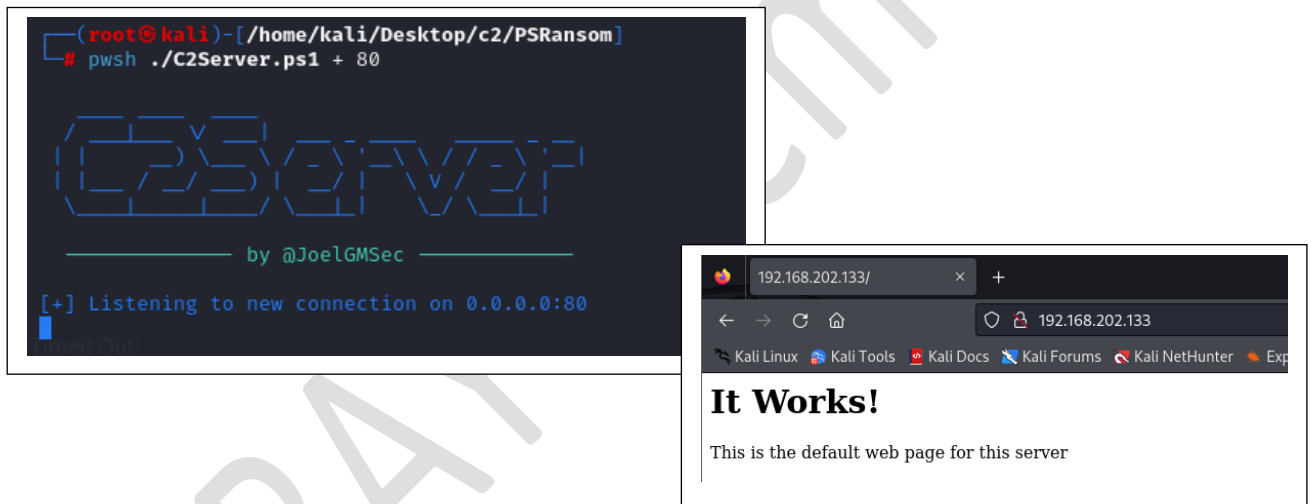

X. Attaque sous Windows

Dans cette partie nous allons détecter une attaque sur un de nos clients Windows. Pour cette partie nous utiliserons le client W10.

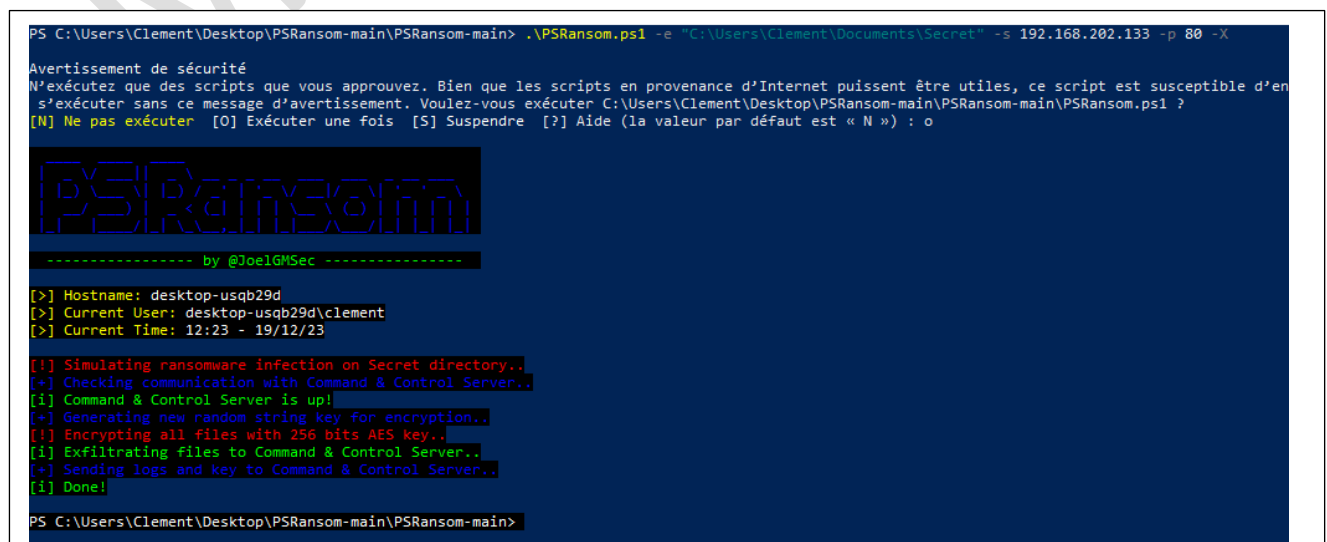
Nous allons simuler un ransomware sur notre agent Windows pour cela on va s'appuyer sur ce guide : <https://www.it-connect.fr/demo-ransomware-psransom-chiffrer-donnees/>

Sur la machine kali qui va nous servir de C2 on installe PSransom : git clone <https://github.com/JoelGMSec/PSRansom>

Et nous lançons notre C2 en mode écoute : pwsh ./C2Server.ps1 + 80



Sur mon poste client Windows je vais chiffrer un de mes dossiers et exfiltrer les données vers mon C2 (windows defender désactivé) :



Sur mon serveur C2 j'ai bien récupéré les fichiers du répertoire chiffré :

```
(root@kali) ~ # ./C2Server.ps1 + 80

C2SERVER
----- by @JoelGMsec -----

[+] Listening to new connection on 0.0.0.0:80
[!] New connection from 192.168.202.129:50934

[>] Hostname: desktop-usqb29d
[>] Current User: desktop-usqb29d\clement
[>] Current Time: 12:23 - 19/12/23

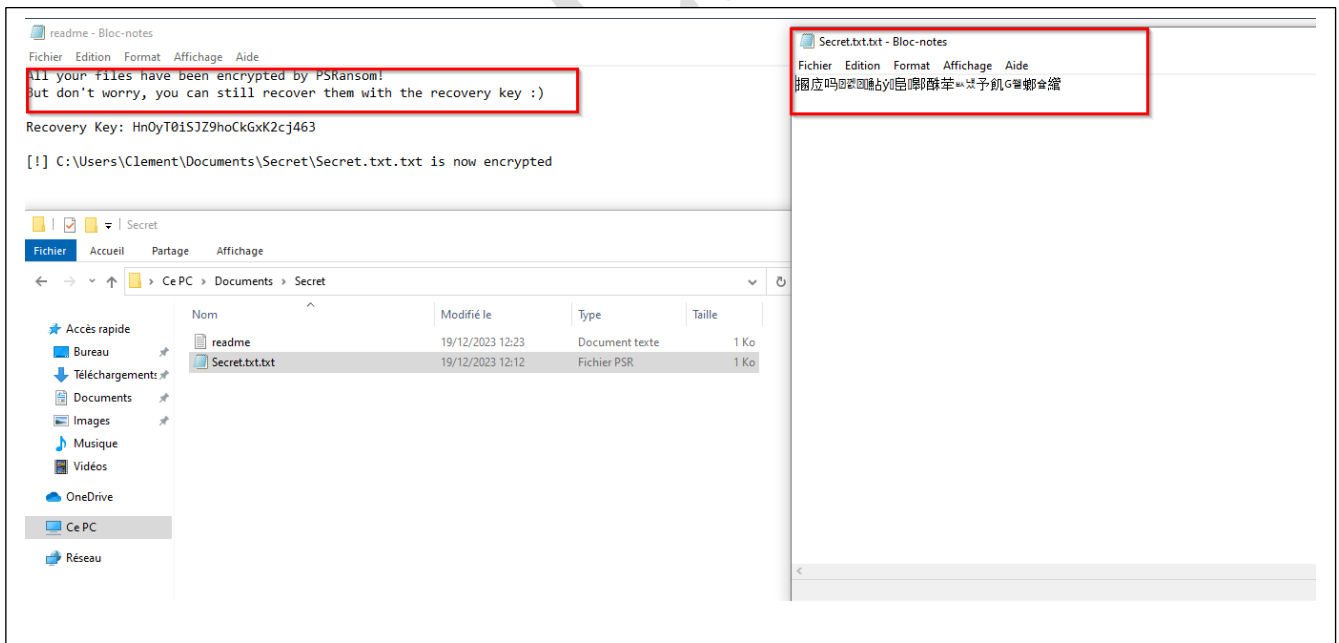
[i] Getting recovery key..
HnOyT0iSJZ9hoCkGxK2cj463

[i] Getting encrypted files list..
[!] C:\Users\Clement\Documents\Secret\Secret.txt.txt is now encrypted

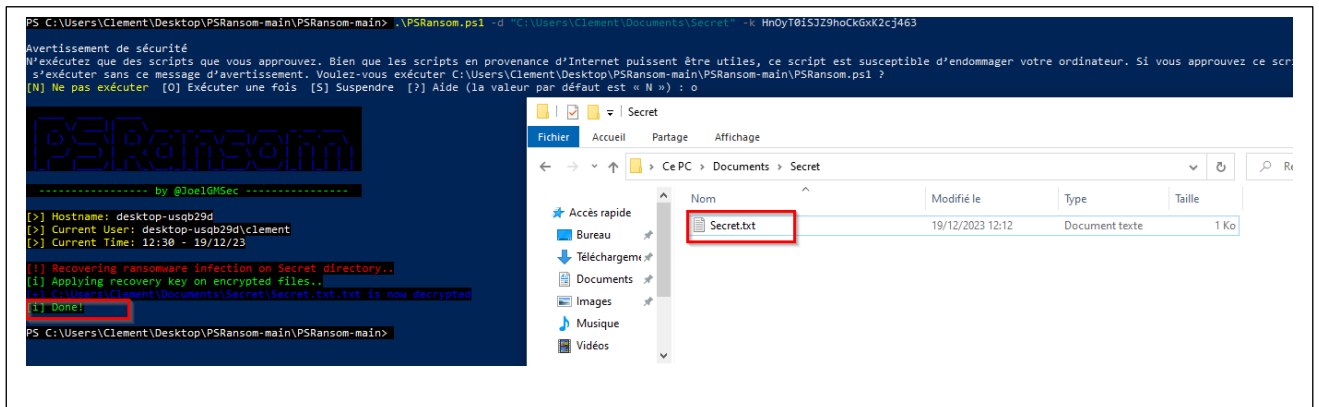
[i] Receiving exfiltrated files and decrypting..
[+] /home/kali/Desktop/c2/PSRansom/C2Files/Secret.txt.txt file received

[i] Done!
```

Si je veux accéder au contenu de mon fichier sur mon client Windows cela est illisible et bien chiffré :



Puis je peux le déchiffrer et récupérer mes données :



Notre but maintenant est de détecter cette attaque avec Wazuh et de la bloquer. Nous allons faire trigger Wazuh quand le fichier Secret dans le répertoire C:\Users\Clement\Documents\Secret est altéré.

Pour ça, nous allons suivre le guide suivant pour mettre en place le FIM :

<https://documentation.wazuh.com/current/proof-of-concept-guide/poc-file-integrity-monitoring.html>

Sur l'agent Windows je modifie le fichier C:\Program Files (x86)\ossec-agent\ossec.conf et j'ajoute le bloc suivant dans la section syscheck :

```
<directories check_all="yes"
report_changes="yes" realtime="yes">C:\Users\Clement\Documents\Secret</directories>
```



Puis je redémarre le client Wazuh du Windows.

Si je modifie mon fichier Secret.txt, Wazuh manager va me remonter une alarme comme quoi le fichier a été modifié :



```
data.aws.accountId
data.aws.region
decoder.name syscheck_integrity_changed
full_log > File 'c:\users\clement\documents\secret\secret.txt' modified
Mode: realtime
Changed attributes: size,mtime,md5,sha1,sha256
Size changed from '23' to '24'
Old modification time was: '1702992675', now it is '1702992953'
Old md5sum was: '391e9b9d0f271be441b6b861a837769c'
New md5sum is: 'c2d1387452bb7857d850096fdafb4b0b'
```

```
syscheck.changed_attributes size, mtime, md5, sha1, sha256
syscheck.diff < Ceci est un secret !!!!!
---
> Ceci est un secret !!!!!
syscheck.event modified
syscheck.md5_after c2d1387452bb7857d850096fdafb4b0b
syscheck.md5_before 391e9b9d0f271be441b6b861a837769c
```

Wazuh détecte bien la modification du fichier. Si je relance le script PowerShell qui permet de chiffrer le répertoire avec Windows Defender activé sur le client W10 celui-ci est bloqué par WD et est détecté comme une menace par Wazuh :

```
Dec 19, 2023 @ 15:16:55.379 agent.name: DESKTOP-USQ829D input.type: log agent.ip: 192.168.202.129 agent.id: 001 manager.name: clement-virtual-machine data.aws.accountId: data.aws.region: data.win.eventdata.error Description: L'opération a réussi. data.win.eventdata.source ID: 3 data.win.eventdata.origin ID: 1 data.win.eventdata.threat ID: 2147852129 data.win.eventdata.action Name: Non applicable data.win.eventdata.additional Actions String: No additional actions required data.win.eventdata.severity Name: Grave data.win.eventdata.path: file:C:\Users\Clement\Desktop\PSRansom-main\PSRansom-main\PSRansom.ps1 data.win.eventdata.execution ID: 1 data.win.eventdata.product Name: Antivirus Microsoft Defender data.win.eventdata.action ID: 9 data.win.eventdata.category ID: 50 data.win.eventdata.state: 1 data.win.eventdata.error Code: 0x00000000
```

```
data.win.eventdata.severity Name Grave
data.win.eventdata.source ID 3
data.win.eventdata.source Name Protection en temps réel
data.win.eventdata.state 1
data.win.eventdata.status Code 1
data.win.eventdata.threat ID 2147852129
data.win.eventdata.threat Name Ransom:PowerShell/MalgentMSR
data.win.eventdata.type ID 0
data.win.eventdata.type Name Concret
data.win.system.channel Microsoft-Windows-Windows Defender/Operational
data.win.system.computer DESKTOP-USQ829D
data.win.system.eventID 1116
data.win.system.eventRecordID 112
data.win.system.keywords 0x0000000000000000
data.win.system.level 3
data.win.system.message > "Antivirus Microsoft Defender a détecté un logiciel malveillant ou potentiellement indésirable.
Pour plus d'informations, reportez-vous aux éléments suivants :
https://go.microsoft.com/fwlink/?linkid=37020&name=Ransom:PowerShell/MalgentMSR&threatid=2147852129&enterprise=0
Non Ransom:PowerShell/MalgentMSR
ID : 2147852129
Gravité : Grave
```

Nous savons que si WD est actif la menace sera bloquée cependant, si WD est désactivé rien ne se passe, actuellement sur Wazuh on sait juste que le fichier a été modifié / chiffré :

```

> Dec 19, 2023 @ 15:24:51.094 agent_name: DESKTOP-USQ829D syscheck.mode: realtime syscheck.path: c:\users\clement\documents\secret\secret.txt
syscheck.sha1_after: 723a8e2fb4f849f8650df10e4849461b7b667bf1 syscheck.uname_after: Administrateurs syscheck.mtime_after: Dec 19, 2023 @ 16:10:43.000
syscheck.attrs_after: ARCHIVE syscheck.size_after: 23 syscheck.uid_after: S-1-5-32-544 syscheck.win_perm_after: { "allowed": [ "DELETE", "READ_CONTROL", "WRITE_DAC",
"WRITE_OWNER", "SYNCHRONIZE", "READ_DATA", "WRITE_DATA", "APPEND_DATA", "READ_EA", "WRITE_EA", "EXECUTE", "READ_ATTRIBUTES", "WRITE_ATTRIBUTES" ], "name": "Système" }, {
"allowed": [ "DELETE", "READ_CONTROL", "WRITE_DAC", "WRITE_OWNER", "SYNCHRONIZE", "READ_DATA", "WRITE_DATA", "APPEND_DATA", "READ_EA", "WRITE_EA", "EXECUTE",
> Dec 19, 2023 @ 15:24:51.041 agent_name: DESKTOP-USQ829D syscheck.mode: realtime syscheck.path: c:\users\clement\documents\secret\secret.txt.psr
syscheck.sha1_after: 8572f4620c0f3aed6452b27e9f0c56ab25930c839 syscheck.uname_after: Administrateurs syscheck.mtime_after: Dec 19, 2023 @ 16:10:43.000
syscheck.attrs_after: ARCHIVE syscheck.size_after: 48 syscheck.uid_after: S-1-5-32-544 syscheck.win_perm_after: { "allowed": [ "DELETE", "READ_CONTROL", "WRITE_DAC",
"WRITE_OWNER", "SYNCHRONIZE", "READ_DATA", "WRITE_DATA", "APPEND_DATA", "READ_EA", "WRITE_EA", "EXECUTE", "READ_ATTRIBUTES", "WRITE_ATTRIBUTES" ], "name": "Système" }, {
"allowed": [ "DELETE", "READ_CONTROL", "WRITE_DAC", "WRITE_OWNER", "SYNCHRONIZE", "READ_DATA", "WRITE_DATA", "APPEND_DATA", "READ_EA", "WRITE_EA", "EXECUTE",
> Dec 19, 2023 @ 15:24:50.989 agent_name: DESKTOP-USQ829D syscheck.mode: realtime syscheck.path: c:\users\clement\documents\secret\readme.txt
syscheck.sha1_after: 0f87a2fc51f335f91176905bf43dbec6d156ffd6 syscheck.uname_after: Administrateurs syscheck.mtime_after: Dec 19, 2023 @ 16:24:50.000
syscheck.attrs_after: ARCHIVE syscheck.size_after: 228 syscheck.uid_after: S-1-5-32-544 syscheck.win_perm_after: { "allowed": [ "DELETE", "READ_CONTROL", "WRITE_DAC",
"WRITE_OWNER", "SYNCHRONIZE", "READ_DATA", "WRITE_DATA", "APPEND_DATA", "READ_EA", "WRITE_EA", "EXECUTE", "READ_ATTRIBUTES", "WRITE_ATTRIBUTES" ], "name": "Système" }, {
"allowed": [ "DELETE", "READ_CONTROL", "WRITE_DAC", "WRITE_OWNER", "SYNCHRONIZE", "READ_DATA", "WRITE_DATA", "APPEND_DATA", "READ_EA", "WRITE_EA", "EXECUTE",

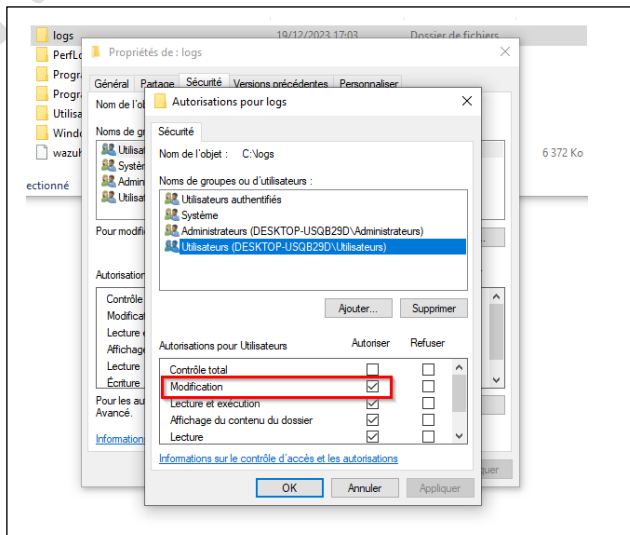
```

- Dans la partie 3 on voit la création du fichier readme.txt disant que notre répertoire a été chiffré et les instructions à suivre.
- Dans la partie 2 on voit que le fichier secret vient d’être chiffré -> secret.txt.psr
- Dans la partie 1 on voit que le fichier secret d’origine a été supprimé.

Pour mitiger cette attaque le plus possible, on va essayer de faire en sorte de bloquer tous les flux sortants du poste client et faire en sorte de l’isoler côté Réseau.

Voici la documentation qui va nous aider : <https://documentation.Wazuh.com/current/user-manual/capabilities/active-response/custom-active-response-scripts.html>

Nous allons faire dans un premier temps un script python qui va créer un fichier .txt dans le répertoire C:\logs. Nous devons accorder les droits aux utilisateurs de la machine à écrire sur ce répertoire :



Notre script python est le suivant :

Il va créer un fichier de log dans le répertoire C:\logs. Cela permet de voir dans un premier temps si le script est bien exécuté quand le plugin id 550 est détecté par le Wazuh manager

```
import subprocess

def configure_network():

    command = 'echo "a" > C:\logs\ici.txt'

    try:

        subprocess.run(command, check=True, shell=True)

        print("Command executed successfully.")

    except subprocess.CalledProcessError as e:

        print(f"Error executing command: {e}")

if name == "main":

    configure_network()
```

Une fois le script python fonctionnel, nous le convertissons en .exe puis nous recopions ce .exe dans C:\Program Files (x86)\ossec-agent\active-response\bin\

Nous redémarrons ensuite le client Wazuh sur le serveur windows. Côté Wazuh manager nous modifions le fichier /var/ossec/etc/ossec.conf et rajoutons ce bloc pour lui dire que dès qu'il détecte le plugin id 550 il exécute le script .exe précédemment créé et dépose dans \bin du client wazuh Windows.

```
<command>
  <name>windows-custom-ar</name>
  <executable>script.exe</executable>
  <timeout_allowed>yes</timeout_allowed>
</command>
```

```
<ossec_config>
:
:
: <command>
:   <name>windows-custom-ar</name>
:   <executable>script.exe</executable>
:   <timeout_allowed>yes</timeout_allowed>
: </command>
:
: <active-response>
:   <disabled>no</disabled>
:   <command>windows-custom-ar</command>
:   <location>local</location>
:   <rules_id>550</rules_id>
:   <timeout>60</timeout>
: </active-response>
:
:
: </ossec_config>
```

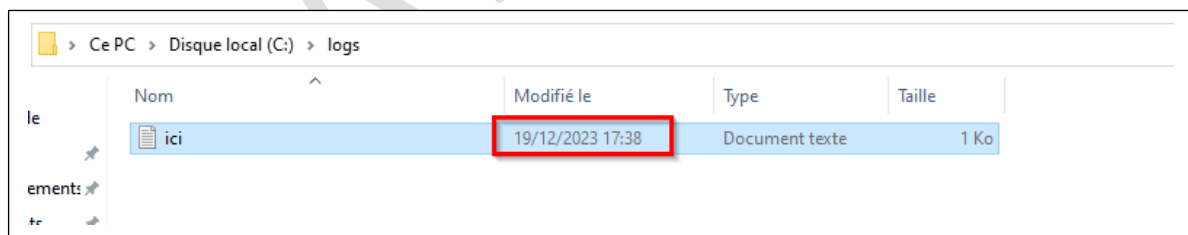
```
<active-response>
  <disabled>no</disabled>
  <command>windows-custom-ar</command>
  <location>local</location>
  <rules_id>550</rules_id>
  <timeout>60</timeout>
</active-response>
```

Test:

Je modifie alors notre fichier secret.txt pour qu'il trigger Wazuh (règle anciennement crée).

```
> Dec 19, 2023 @ 17:38:01.833 @agent.name: DESKTOP-USQB29U input.type: log agent.ip: 192.168.202.129 agent.id: 001 manager.name: clement-virtual-machine data.aws.accountId: data.aws.region:
data.win.eventdata.param3: Démarrage à la demande data.win.eventdata.param4: BITS data.win.eventdata.param1: Service de transfert intelligent en arrière-plan
data.win.eventdata.param2: Démarrage automatique data.win.system.eventID: 7840 data.win.system.eventSourceName: Service Control Manager
data.win.system.keywords: 0x8080000000000000 data.win.system.providerGuid: {559988d1-a6d7-4695-8e1e-26931d2012f4} data.win.system.level: 4 data.win.system.channel: System
data.win.system.opcode: 0 data.win.system.message: "Le type de démarrage du service Service de transfert intelligent en arrière-plan est passé de Démarrage automatique à
> Dec 19, 2023 @ 17:37:10.535 @agent.name: DESKTOP-USQB29U syscheck.size_before: 41 syscheck.uname_after: Administrateurs syscheck.mtime_after: Dec 19, 2023 @ 18:37:10.000 syscheck.size_after: 42
syscheck.md5_before: 4b1a057e4a0a5495e2aa1d79874b363b syscheck.win_perm_after: { "allowed": [ "DELETE", "READ_CONTROL", "WRITE_DAC", "WRITE_OWNER", "SYNCHRONIZE",
"READ_DATA", "WRITE_DATA", "APPEND_DATA", "READ_EA", "WRITE_EA", "EXECUTE", "READ_ATTRIBUTES", "WRITE_ATTRIBUTES" ], "name": "Système" }, { "allowed": [ "DELETE",
"READ_CONTROL", "WRITE_DAC", "WRITE_OWNER", "SYNCHRONIZE", "READ_DATA", "WRITE_DATA", "APPEND_DATA", "READ_EA", "WRITE_EA", "EXECUTE", "READ_ATTRIBUTES", "WRITE_ATTRIBUTES" ],
"name": "Administrateurs" }, { "allowed": [ "DELETE", "READ_CONTROL", "WRITE_DAC", "WRITE_OWNER", "SYNCHRONIZE", "READ_DATA", "WRITE_DATA", "APPEND_DATA", "READ_EA",
```

Wazuh détecte bien que le fichier à été modifié. Par conséquent il doit alors nous créer le fichier logs sur le client Windows (exécuter le script précédemment créé).



Notre réponse active custom fonctionne bien et crée bien le fichier lors dès que le fichier Secret.txt est modifié ! Nous allons alors essayer d'améliorer notre script pour qu'il exécute une commande netsh pour bloquer tous les flux sortants.

Nous avons fait deux règles sur notre FW local Windows pour bloquer les flux entrants et sortants :

```
netsh advfirewall firewall add rule name="Wazuh bloquer in" dir=in action=block
```

```
netsh advfirewall firewall add rule name="Wazuh bloquer out" dir=out action=block
```

Via notre script python nous allons lui dire d'activer ces deux règles pour bloquer tous les flux entrants et sortants du client :

```
netsh advfirewall firewall set rule name="Wazuh bloquer in" new enable=yes
```

```
netsh advfirewall firewall set rule name="Wazuh bloquer out" new enable=yes
```

```
import subprocess
import time

def configure_network():
    command = 'echo Menace detectee fermeture des flux sortants > C:\logs\logs.txt'
    command2 = 'netsh advfirewall firewall set rule name="Wazuh bloquer in" new enable=yes'
    command3 = 'netsh advfirewall firewall set rule name="Wazuh bloquer out" new enable=yes'

    try:
        subprocess.run(command, check=True, shell=True)

        time.sleep(2)

        subprocess.run(command2, check=True, shell=True)
        print("Command executed successfully.")

        time.sleep(2)

        subprocess.run(command3, check=True, shell=True)
        print("Command executed successfully.")
    except subprocess.CalledProcessError as e:
        print(f"Error executing command: {e}")

        time.sleep(2)

if __name__ == "__main__":
    configure_network()
```


Nous allons maintenant compiler ce script pour le transformer en .exe et refaire l'ancien test. C'est-à-dire modifier mon fichier secret. Wazuh va détecter la modification et va par conséquent bloquer tous les flux entrants et sortants. Je transforme mon script python en un .exe :

nom	modifié le	type	taille
script	19/12/2023 18:09	Application	7 090 Ko

Je dépose cet exe dans C:\Program Files (x86)\ossec-agent\active-response\bin\

Nom	Modifié le	Type	Taille
netsh	23/11/2023 17:06	Application	188 Ko
restart-wazuh	23/11/2023 17:06	Application	183 Ko
route-null	23/11/2023 17:06	Application	185 Ko
script	19/12/2023 18:09	Application	7 090 Ko

Et je redémarre l'agent Wazuh sur mon client. Pour cet exemple je vais réutiliser mon C2 qui va simuler un ransomware et chiffrer le fichier secret. Par conséquent je vais désactiver Windows Defender (le script fonctionne tout aussi bien si Windows Defender est activé). Le fait de désactiver WD et de pouvoir faire la démo avec le serveur C2 et l'exfiltration (sans la désactivation WD, celui-ci supprimera le fichier C2 PowerShell car il le verra comme une menace).

Sur ma kali je redémarre mon serveur C2 :

```
(root@kali) - [~/home/kali/Desktop/c2/PSRansom]
# pwsh ./C2Server.ps1 + 80

C2SERVER
by @JoelGMSec

[+] Listening to new connection on 0.0.0.0:80
```

Tout au long de cette manipulation depuis mon client W11 je ping le serveur Wazuh (cela indique que les flux sortants ne sont pas bloqués) :

```
Réponse de 192.168.202.131 : octets=32 temps<1ms TTL=64
Réponse de 192.168.202.131 : octets=32 temps<1ms TTL=64
Réponse de 192.168.202.131 : octets=32 temps<1ms TTL=64
Réponse de 192.168.202.131 : octets=32 temps<1ms TTL=64
Réponse de 192.168.202.131 : octets=32 temps=1 ms TTL=64
Réponse de 192.168.202.131 : octets=32 temps<1ms TTL=64
Réponse de 192.168.202.131 : octets=32 temps<1ms TTL=64
Réponse de 192.168.202.131 : octets=32 temps<1ms TTL=64
Réponse de 192.168.202.131 : octets=32 temps<1ms TTL=64
Réponse de 192.168.202.131 : octets=32 temps<1ms TTL=64
Réponse de 192.168.202.131 : octets=32 temps<1ms TTL=64
Réponse de 192.168.202.131 : octets=32 temps=1 ms TTL=64
Réponse de 192.168.202.131 : octets=32 temps<1ms TTL=64
Réponse de 192.168.202.131 : octets=32 temps<1ms TTL=64
Réponse de 192.168.202.131 : octets=32 temps<1ms TTL=64
```


Sur mon C2, les flux étants bloqués je ne reçoit pas le fichier secret qui devait être extrait et le script plante. Le fichier n'est pas extrait/envoyé à l'attaquant :

```
(root@kali)~/home/kali/Desktop/c2/PSRansom
# pwsh ./C2Server.ps1 + 80

          O S O W A T
          by @JoelGMSec

[+] Listening to new connection on 0.0.0.0:80
[!] New connection from 192.168.202.129:55809

[>] Hostname: desktop-usqb29d
[>] Current User: desktop-usqb29d\clement
[>] Current Time: 18:34 - 19/12/23

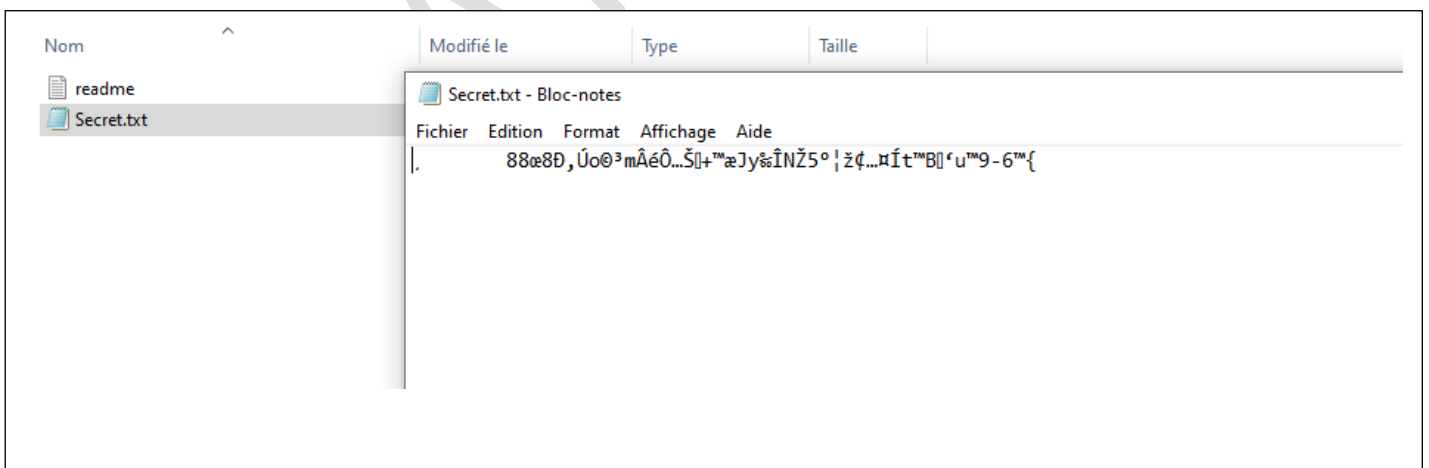
[i] Getting recovery key..
m7taeKDKNs86GM30HJFOVBQx

[i] Getting encrypted files list..
[!] C:\Users\Clement\Documents\Secret\Secret.txt is now encrypted

[i] Recieving exfiltrated files and decrypting..
```

```
(root@kali)~/home/.../Desktop/c2/PSRansom/C2Files
# ls -l
total 0
ne default web page for this server
```

Pendant le fichier lui a eu le temps d'être chiffré :



Nous avons alors isolé notre client W10 victime d'une attaque par Ransomware ! Celle-ci a été détectée par Wazuh et bloquée grâce à celui-ci. Et notre fichier logs a bien été créé.¹

¹ Fin du Compte Rendu

