Surveillance de l'intégrité des fichiers et YARA

YARA est un outil open source qui identifie les logiciels malveillants en se basant sur des modèles textuels ou binaires appelés règles **YARA**.

Lorsqu'il est combiné avec **Wazuh**, notre système de détection de menaces, il renforce la détection des logiciels malveillants.

Wazuh surveille les fichiers sur les points de terminaison, déclenchant une analyse **YARA** lorsque des modifications sont détectées.

Si des correspondances sont trouvées avec les règles **YARA**, **Wazuh** génère des alertes pour signaler les menaces potentielles.

Cette approche permet une détection efficace tout en optimisant l'utilisation des ressources.

I - Configuration des points de terminaison Linux

1. On télécharge, compile et installe **YARA** :

```
sudo apt update
sudo apt install -y make gcc autoconf libtool libssl-dev pkg-config
sudo curl -LO https://github.com/VirusTotal/yara/archive/v4.2.3.tar.gz
sudo tar -xvzf v4.2.3.tar.gz -C /usr/local/bin/ && rm -f v4.2.3.tar.gz
cd /usr/local/bin/yara-4.2.3/
sudo ./bootstrap.sh && sudo ./configure && sudo make && sudo make install && sudo make check
```

2. On teste que **YARA** fonctionne correctement.

```
yara
```

Résultat attendu :

```
yara: wrong number of arguments
Usage: yara [OPTION]... [NAMESPACE:]RULES_FILE... FILE | DIR | PID

Try `--help` for more options
```

Si le message d'erreur ci-dessous s'affiche :

```
/usr/local/bin/yara: error while loading shared libraries: libyara.so.9: cannot open shared object file: No such file or directory.
```

Cela signifie que le chargeur ne trouve pas la bibliothèque **libyara** habituellement située dans **/usr/local/lib**. On ajoute alors le chemin d'accès **/usr/local/lib** au fichier de configuration du chargeur **/etc/ld.so.conf** pour résoudre ce problème avec :

```
sudo su
echo "/usr/local/lib" >> /etc/ld.so.conf
ldconfig
```

3. On télécharge les règles de détection **YARA** (/!\ La 2e commande commence à **sudo** jusqu'à **yara_rules.yar**) :

```
sudo mkdir -p /tmp/yara/rules
sudo curl 'https://valhalla.nextron-systems.com/api/v1/get' -H 'Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8' -H 'Accept-Language: en-US,en;q=0.5' --
compressed -H 'Referer: https://valhalla.nextron-systems.com/' -H 'Content-Type: application/x-www-form-
urlencoded' -H 'DNT: 1' -H 'Connection: keep-alive' -H 'Upgrade-Insecure-Requests: 1' --data
'demo=demo&apikey=1111111111111111111111111111111111111111111111111111111111111111&format=text' -o
/tmp/yara/rules/yara_rules.yar
```

4. On crée le script bash **/var/ossec/active-response/bin/yara.sh** et on ajoute le contenu ci-dessous :

```
#!/bin/bash
# Wazuh - Yara active response
# Copyright (C) 2015-2022, Wazuh Inc.
#
# This program is free software; you can redistribute it
# and/or modify it under the terms of the GNU General Public
# License (version 2) as published by the FSF - Free Software
# Foundation.


#------------------------ Gather parameters -------------------------#

# Extra arguments
read INPUT_JSON
YARA_PATH=$(echo $INPUT_JSON | jq -r .parameters.extra_args[1])
YARA_RULES=$(echo $INPUT_JSON | jq -r .parameters.extra_args[3])
FILENAME=$(echo $INPUT_JSON | jq -r .parameters.alert.syscheck.path)

# Set LOG_FILE path
LOG_FILE="logs/active-responses.log"

size=0
actual_size=$(stat -c %s ${FILENAME})
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;
```

Il s'agit du script de réponse actif qui exécute les analyses **YARA** lorsque **FIM** détecte des modifications dans le répertoire surveillé.

Surveillance de l'intégrité des fichiers et YARA

5. On modifie la propriété et les autorisations du script avec les commandes suivantes :

```
sudo chmod 750 /var/ossec/active-response/bin/yara.sh
sudo chown root:wazuh /var/ossec/active-response/bin/yara.sh
```

6. On installe l'utilitaire **jq** pour traiter les données **JSON** des alertes **FIM** :

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

7. On ajoute ce qui suit dans le bloc **<syscheck>** du fichier de configuration **/var/ossec/etc/ossec.conf** de l'agent **Wazuh** pour surveiller le répertoire **/root/** :

```
<directories check_all="yes" realtime="yes" report_changes="yes" whodata="yes">/root/</directories>
```

8. On redémarre l'agent **Wazuh** pour appliquer les modifications de configuration :

```
sudo systemctl restart wazuh-agent
```

II - Configuration du serveur Wazuh

1. On ajoute les décodeurs personnalisés suivants au fichier **/var/ossec/etc/decoders/local_decoder.xml**. Cela extrait les informations des résultats de l'analyse **YARA** :

```
<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>
```

2. On ajoute les règles personnalisées suivantes au fichier **/var/ossec/etc/rules/local_rules.xml** :

```
<group name="syscheck,">
  <rule id="100200" level="7">
    <if_sid>550</if_sid>
    <field name="file">/root/</field>
    <description>File modified in /root directory.</description>
  </rule>
  <rule id="100201" level="7">
    <if_sid>554</if_sid>
    <field name="file">/root/</field>
    <description>File added to /root directory.</description>
  </rule>
</group>

<group name="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>
```

Surveillance de l'intégrité des fichiers et YARA

3. On configure l'exécution du script **YARA** lorsque des fichiers sont ajoutés ou modifiés dans un répertoire surveillé. Pour cela, on modifie le fichier de configuration du serveur **Wazuh /var/ossec/etc/ossec.conf** en ajoutant ce qui suit :

```
<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>100200,100201</rules_id>
  </active-response>
</ossec_config>
```

4. On redémarre le gestionnaire **Wazuh** pour appliquer les modifications de configuration :

```
sudo systemctl restart wazuh-manager
```

III – Test de la configuration

Pour tester que tout fonctionne correctement, nous utilisons les exemples de logiciels malveillants **Mirai** et **Xbash**.

/!\ Ces fichiers malveillants sont dangereux, on les utilise donc uniquement à des fins de test. À ne surtout pas utiliser dans des environnements de production.

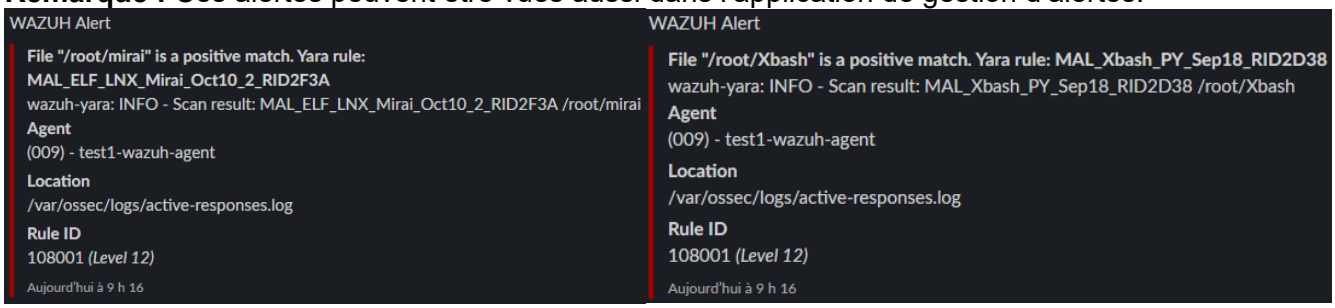
1. On télécharge les échantillons de logiciels malveillants et on les déplace dans le répertoire **/root/** du point de terminaison surveillé :

```
curl https://wazuh-demo.s3-us-west-1.amazonaws.com/mirai --output ~/mirai
curl https://wazuh-demo.s3-us-west-1.amazonaws.com/xbash --output ~/Xbash
sudo mv ~/mirai /root/
sudo mv ~/Xbash /root/
```

2. On regarde les alertes sur l'interface web de **Wazuh**. Pour cela, on se rend dans l'onglet **Security Events** du tableau de bord **Wazuh** pour visualiser les alertes.

›	26 mars 2024 à 09:16:28.350	009	test1-wazuh-agent	Le fichier "/root/mirai" est une correspondance positive. Règle Yara : MAL_ELF_LNX_Mirai_Oct10_2_RID2F3A	12	108001
›	26 mars 2024 à 09:16:28.350	009	test1-wazuh-agent	Le fichier "/root/Xbash" est une correspondance positive. Règle Yara : MAL_Xbash_PY_Sep18_RID2D38	12	108001

Remarque : Ces alertes peuvent être vues aussi dans l'application de gestion d'alertes.



Surveillance de l'intégrité des fichiers et YARA

IV - Configuration du point de terminaison Windows

1. On télécharge le programme d'installation de l'exécutable Python (et éventuellement les paquets intégrables) depuis le site Web officiel de **Python**.

2. On exécute le programme d'installation **Python** une fois téléchargé et on s'assure de cocher les cases suivantes :
   - **Installer le lanceur pour tous les utilisateurs**
   - **Ajoutez Python 3.X à PATH.** Cela place l'interpréteur dans le chemin d'exécution.

3. On télécharge et installe le dernier package redistribuable Visual C++.

4. On ouvre un **PowerShell** avec les privilèges d'administrateur pour télécharger et extraire **YARA** :

```
Invoke-WebRequest -Uri https://github.com/VirusTotal/yara/releases/download/v4.2.3/yara-4.2.3-2029-win64.zip -OutFile v4.2.3-2029-win64.zip

Expand-Archive v4.2.3-2029-win64.zip; Remove-Item v4.2.3-2029-win64.zip
```

5. On crée un répertoire appelé **yara** dans **C:\Program Files (x86)\ossec-agent\active-response\bin** et on copie l'exécutable **YARA** :

```
mkdir 'C:\Program Files (x86)\ossec-agent\active-response\bin\yara\'
cp .\v4.2.3-2029-win64\yara64.exe 'C:\Program Files (x86)\ossec-agent\active-response\bin\yara\'
```

6. On installe le module **valhallaAPI** :

```
pip install valhallaAPI
```

7. On crée le script **download_yara_rules.py** et on copie le contenu suivant puis on l'enregistre :

```
from valhallaAPI.valhalla import ValhallaAPI

v = ValhallaAPI(api_key="1111111111111111111111111111111111111111111111111111111111111111")
response = v.get_rules_text()

with open('yara_rules.yar', 'w') as fh:
    fh.write(response)
```

8. On exécute les commandes suivantes pour télécharger les règles et on les place dans le répertoire **C:\Program Files (x86)\ossec-agent\active-response\bin\yara\rules** :

```
python.exe download_yara_rules.py
mkdir 'C:\Program Files (x86)\ossec-agent\active-response\bin\yara\rules\'
cp yara_rules.yar 'C:\Program Files (x86)\ossec-agent\active-response\bin\yara\rules\'
```

Surveillance de l'intégrité des fichiers et YARA

V - Configurer la réponse active et FIM

1. On crée le script **yara.bat** dans le répertoire **C:\Program Files (x86)\ossec-agent\active-response\bin**. Ceci est nécessaire pour les analyses de réponse active **Wazuh-YARA** :

```
@echo off

setlocal enableDelayedExpansion

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

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

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

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

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

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

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

exit /b
```

2. On ajoute le répertoire **C:\Users\<USER_NAME>\Downloads** parmi les répertoires surveillés au sein du bloc **<syscheck>** dans le fichier de configuration de l'agent **Wazuh** (**C:\Program Files (x86)\ossec-agent\ossec.conf**). On remplace **<USER_NAME>** par le nom d'utilisateur du point de terminaison :

```
<directories realtime="yes">C:\Users\<USER_NAME>\Downloads</directories>
```

3. On redémarre l'agent **Wazuh** pour appliquer les modifications de configuration soit :
   - dans le **Gestionnaire des Tâches**
   - avec la commande à exécuter dans **PowerShell** :

```
Restart-Service -Name wazuh
```

Surveillance de l'intégrité des fichiers et YARA

VI - Configuration du serveur Wazuh

1. On ajoute les décodeurs suivants au fichier **/var/ossec/etc/decoders/local_decoder.xml** du serveur **Wazuh.** Cela permet d'extraire les informations des résultats de l'analyse **YARA** :

```
<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>
```

2. On ajoute les règles suivantes au fichier **/var/ossec/etc/rules/local_rules.xml** du serveur **Wazuh**. On remplace **<USER_NAME>** par le nom d'utilisateur du point de terminaison. Les règles détectent les événements **FIM** dans le répertoire surveillé. Ils alertent également lorsqu'un malware est détecté par l'intégration **YARA** :

```
<group name="syscheck,">
 <rule id="100303" level="7">
  <if_sid>550</if_sid>
  <field name="file">C:\\Users\\<USER_NAME>\\Downloads</field>
  <description>File modified in C:\Users\<USER_NAME>\Downloads directory.</description>
 </rule>
 <rule id="100304" level="7">
  <if_sid>554</if_sid>
  <field name="file">C:\\Users\\<USER_NAME>\\Downloads</field>
  <description>File added to C:\Users\<USER_NAME>\Downloads  directory.</description>
 </rule>
</group>

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

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

3. On ajoute la configuration suivante au fichier **/var/ossec/etc/ossec.conf** du serveur **Wazuh** :

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

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

4. On redémarre le gestionnaire **Wazuh** pour appliquer les modifications de configuration :

```
sudo systemctl restart wazuh-manager
```

VII – Teste de la configuration

Note : À des fins de test, nous téléchargeons le fichier de test anti-malware ***EICAR*** comme indiqué ci-dessous. Il est recommandé de tester dans un bac à sable et non dans un environnement de production.

1. On désactive temporairement la protection **Microsoft** contre les virus et les menaces ou autres anti-virus si possible pour éviter la suppression du fichier **EICAR**.

2. Avec un **PowerShell**, on télécharge le fichier zip EICAR :

```
Invoke-WebRequest -Uri https://secure.eicar.org/eicar_com.zip -OutFile eicar.zip
```

3. On le décompresse :

```
Expand-Archive .\eicar.zip
```

4. On copie le fichier **EICAR** dans le répertoire surveillé :

```
cp .\eicar\eicar.com C:\Users\<USER_NAME>\Downloads
```

5. On regarde les alertes sur l'interface web de **Wazuh**. Pour cela, on se rend dans l'onglet « **Security Events** » du tableau de bord **Wazuh** pour visualiser les alertes.

Remarque : Ces alertes peuvent être vues aussi dans l'application de gestion d'alertes.