檢測並刪除惡意軟件使用 VirusTotal 集成

Wazuh 使用 integrator 模組來連接到外部 API 和警報工具，如 VirusTotal。

在這個使用案例中，您使用 Wazuh 文件完整性監控（FIM）模組來監控目錄的變化，並使用 VirusTotal API 掃描目錄中的文件。然後，配置 Wazuh 來觸發主動響應腳本並刪除 VirusTotal 標識為惡意的文件。我們在 Ubuntu 和 Windows 端點上測試這個使用案例。

在這個使用案例中，您需要一個 VirusTotal API 密鑰，以便 Wazuh 能夠通過 VirusTotal API 進行驗證。

有關該集成的更多信息，請查閱文檔的 VirusTotal 集成部分。

基礎架構

端點

描述

Ubuntu 22.04

這是 Linux 端點，您在其中下載惡意文件。Wazuh 在 VirusTotal 將其標記為惡意文件時觸發一個主動響應腳本來刪除該文件。

Windows 11

這是 Windows 端點，您在其中下載惡意文件。Wazuh 在 VirusTotal 將其標記為惡意文件時觸發一個主動響應腳本來刪除該文件。

Ubuntu 端點配置

根據以下配置在 Ubuntu 端點上進行環境設置，以測試 Ubuntu 端點的使用案例。這些步驟同樣適用於其他 Linux 發行版。

Ubuntu 端點

執行以下步驟來配置 Wazuh，以在 Ubuntu 端點的/root 目錄中監控近實時變化。這些步驟還會安裝必要的軟件包並創建用於刪除惡意文件的主動響應腳本。

尋找 Wazuh 代理配置文件/var/ossec/etc/ossec.conf 中的<syscheck>塊。確保<disabled>設置為 no，這將使 Wazuh FIM 能夠監控目錄變化。

在<syscheck>塊中添加一個條目，以便在近實時中監控一個目錄。在這個例子中，您正在監控/root 目錄：

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

安裝 jq，這是一個處理主動響應腳本中的 JSON 輸入的實用工具。

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

創建/var/ossec/active-response/bin/remove-threat.sh 主動響應腳本，用於從端點刪除惡意文件：

```
#!/bin/bash

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

PWD=`pwd`

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

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

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

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

exit 0;
```

更改/var/ossec/active-response/bin/remove-threat.sh 文件的所有者和權限：

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

重新啟動 Wazuh 代理以應用更改：

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

Wazuh 服务器

在 Wazuh 服务器上执行以下步骤，以警告有关端点目录中的更改，并启用 VirusTotal 集成。这些步骤还启用并在检测到可疑文件时触发主动响应脚本。

将以下规则添加到 Wazuh 服务器上的/var/ossec/etc/rules/local_rules.xml 文件中。这些规则在通过 FIM 扫描检测到/root 目录中的更改时发出警报：

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

将以下配置添加到 Wazuh 服务器的/var/ossec/etc/ossec.conf 文件中，以启用 VirusTotal 集成。将<YOUR_VIRUS_TOTAL_API_KEY>替换为您的 VirusTotal API 密钥。这将允许在触发规则100200和100201中的任何一个时触发 VirusTotal 查询：

注意：免费的 VirusTotal API 每分钟限制请求为4次。如果您有高频率查询允许的高级 VirusTotal API 密钥，您可以添加更多规则，除了这两个规则之外，也可以配置 Wazuh 来监视更多目录。

将以下块附加到 Wazuh 服务器的/var/ossec/etc/ossec.conf 文件中。这将启用主动响应并在 VirusTotal 标记文件为恶意时触发 remove-threat.sh 脚本：

```
<ossec_config>
  <command>
```

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

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

将以下规则添加到 Wazuh 服务器的/var/ossec/etc/rules/local_rules.xml 文件中，以警告有关主动响应结果的信息：

```
<group name="virustotal,">
  <rule id="100092" level="12">
    <if_sid>657</if_sid>
    <match>Successfully removed threat</match>
    <description>$(parameters.program) removed threat located at
$(parameters.alert.data.virustotal.source.file)</description>
  </rule>

  <rule id="100093" level="12">
    <if_sid>657</if_sid>
    <match>Error removing threat</match>
    <description>Error removing threat located at
$(parameters.alert.data.virustotal.source.file)</description>
  </rule>
</group>
```

重新启动 Wazuh 管理器以应用配置更改：

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

攻击模拟

在 Ubuntu 端点的/root 目录下载 EICAR 测试文件：

```
sudo cd /root
sudo curl -LO https://secure.eicar.org/eicar.com && ls -lah eicar.com
```

可视化警报

您可以在 Wazuh 仪表板中可视化警报数据。要做到这一点，请转到“Security events”模块，并在搜索栏中添加筛选器以查询警报。

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

Windows 端點的配置

Windows 端點

按照以下步驟配置 Wazuh，以監視 /Downloads 目錄中的近實時變更。這些步驟還會安裝必要的軟件包並創建主動響應腳本以刪除惡意文件。

在 Wazuh 代理 C:\Program Files (x86)\ossec-agent\ossec.conf 文件中尋找 <syscheck> 區塊。確保 <disabled> 設置為 no。這將啟用 Wazuh FIM 模組，以監視目錄變更。

在 <syscheck> 區塊內添加一個條目，以配置目錄在近實時中進行監視。在此用例中，您將配置 Wazuh 監視 C:\Users<USER_NAME>\Downloads 目錄。將 <USER_NAME> 變數替換為適當的用戶名：

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

從 Python 官方網站下載 Python 可執行安裝程序。

下載後運行 Python 安裝程序。請確保勾選以下框：

為所有用戶安裝啟動器
將 Python 3.X 添加到 PATH（將解釋器添加到執行路徑）
安裝程序完成後，在管理員 PowerShell 終端中使用 pip 安裝 PyInstaller：

```
> pip install pyinstaller
> pyinstaller –version
```

在這裡使用 PyInstaller 將主動響應 Python 腳本轉換為可以在 Windows 端點上運行的可執行應用程序。

創建一個名為 remove-threat.py 的主動響應腳本，以從 Windows 端點中刪除文件：

```
#!/usr/bin/python3
# Copyright (C) 2015-2022, Wazuh Inc.
# All rights reserved.

import os
import sys
```

```
import json
import datetime

if os.name == 'nt':
    LOG_FILE = "C:\\Program Files (x86)\\ossec-agent\\active-response\\active-responses.log"
else:
    LOG_FILE = "/var/ossec/logs/active-responses.log"

ADD_COMMAND = 0
DELETE_COMMAND = 1
CONTINUE_COMMAND = 2
ABORT_COMMAND = 3

OS_SUCCESS = 0
OS_INVALID = -1

class message:
    def __init__(self):
        self.alert = ""
        self.command = 0

def write_debug_file(ar_name, msg):
    with open(LOG_FILE, mode="a") as log_file:
        log_file.write(str(datetime.datetime.now().strftime('%Y/%m/%d %H:%M:%S')) + " " +
ar_name + ": " + msg +"\n")

def setup_and_check_message(argv):

    # get alert from stdin
    input_str = ""
    for line in sys.stdin:
        input_str = line
        break



    try:
        data = json.loads(input_str)
    except ValueError:
        write_debug_file(argv[0], 'Decoding JSON has failed, invalid input format')
        message.command = OS_INVALID
        return message

    message.alert = data

    command = data.get("command")

    if command == "add":
        message.command = ADD_COMMAND
    elif command == "delete":
        message.command = DELETE_COMMAND
    else:
        message.command = OS_INVALID
```

```
        write_debug_file(argv[0], 'Not valid command: ' + command)

    return message


def send_keys_and_check_message(argv, keys):

    # build and send message with keys
    keys_msg = json.dumps({"version": 1,"origin":{"name": argv[0],"module":"active-
response"},"command":"check_keys","parameters":{"keys":keys}})

    write_debug_file(argv[0], keys_msg)

    print(keys_msg)
    sys.stdout.flush()

    # read the response of previous message
    input_str = ""
    while True:
        line = sys.stdin.readline()
        if line:
            input_str = line
            break

    # write_debug_file(argv[0], input_str)

    try:
        data = json.loads(input_str)
    except ValueError:
        write_debug_file(argv[0], 'Decoding JSON has failed, invalid input format')
        return message

    action = data.get("command")

    if "continue" == action:
        ret = CONTINUE_COMMAND
    elif "abort" == action:
        ret = ABORT_COMMAND
    else:
        ret = OS_INVALID
        write_debug_file(argv[0], "Invalid value of 'command'")

    return ret

def main(argv):

    write_debug_file(argv[0], "Started")

    # validate json and get command
    msg = setup_and_check_message(argv)

    if msg.command < 0:
```

```
        sys.exit(OS_INVALID)

    if msg.command == ADD_COMMAND:
        alert = msg.alert["parameters"]["alert"]
        keys = [alert["rule"]["id"]]
        action = send_keys_and_check_message(argv, keys)

        # if necessary, abort execution
        if action != CONTINUE_COMMAND:

            if action == ABORT_COMMAND:
                write_debug_file(argv[0], "Aborted")
                sys.exit(OS_SUCCESS)
            else:
                write_debug_file(argv[0], "Invalid command")
                sys.exit(OS_INVALID)

        try:
            os.remove(msg.alert["parameters"]["alert"]["data"]["virustotal"]["source"]["file"])
            write_debug_file(argv[0], json.dumps(msg.alert) + " Successfully removed threat")
        except OSError as error:
            write_debug_file(argv[0], json.dumps(msg.alert) + "Error removing threat")


    else:
        write_debug_file(argv[0], "Invalid command")

    write_debug_file(argv[0], "Ended")

    sys.exit(OS_SUCCESS)

if __name__ == "__main__":
    main(sys.argv)
```

將主動響應的 Python 腳本 remove-threat.py 轉換為 Windows 可執行的應用程序。以系統管理員身份運行以下 PowerShell 命令，創建可執行文件：

```
> pyinstaller -F \path_to_remove-threat.py
```

請記下 pyinstaller 創建 remove-threat.exe 的路徑。

將可執行文件 remove-threat.exe 移動到 C:\Program Files (x86)\ossec-agent\active-response\bin 目錄。

重啟 Wazuh 代理以應用更改。以系統管理員身份運行以下 PowerShell 命令：

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

Wazuh 服務器

按照以下步驟在 Wazuh 服務器上配置 VirusTotal 整合。這些步驟還會啟用並在檢測到可疑文件時觸發主動響應腳本。

在 Wazuh 服務器的 /var/ossec/etc/ossec.conf 文件中添加以下配置，以啟用 VirusTotal 整合。將 <YOUR_VIRUS_TOTAL_API_KEY> 替換為您的 VirusTotal API 密鑰。這允許在 FIM syscheck 組中的任何規則被觸發時觸發 VirusTotal 查詢：

```
<ossec_config>
  <integration>
    <name>virustotal</name>
    <api_key><YOUR_VIRUS_TOTAL_API_KEY></api_key> <!-- 將其替換為您的 VirusTotal API 密鑰 -->
    <group>syscheck</group>
    <alert_format>json</alert_format>
  </integration>
</ossec_config>
```

注意：免費的 VirusTotal API 限制每分鐘請求數量為 4 次。如果您有高頻率允許的高級 VirusTotal API 密鑰，可以添加除這兩個外的更多規則。您還可以配置 Wazuh 監視更多目錄，而不僅僅是 C:\Users<USER_NAME>\Downloads。

在 Wazuh 服務器的 /var/ossec/etc/ossec.conf 文件末尾添加以下塊。這將啟用主動響應並在 VirusTotal 查詢返回威脅的正確匹配時觸發 remove-threat.exe 可執行文件：

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

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

在 Wazuh 服務器的 /var/ossec/etc/rules/local_rules.xml 文件中添加以下規則，以警告有關主動響應結果：

```
<group name="virustotal,">
  <rule id="100092" level="12">
      <if_sid>657</if_sid>
      <match>成功刪除威脅</match>
      <description>$(parameters.program) 成功刪除位於 $(parameters.alert.data.virustotal.source.file) 的威脅</description>
  </rule>

  <rule id="100093" level="12">
```

```
    <if_sid>657</if_sid>
    <match>刪除威脅失敗</match>
    <description>刪除位於 $(parameters.alert.data.virustotal.source.file) 的威脅失敗</description>
  </rule>
</group>
```

重啟 Wazuh 管理員以應用配置更改：

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

攻擊模擬

在 Windows 端點的 C:\Users<USER_NAME>\Downloads 目錄下下載 EICAR 測試文件：

```
> Invoke-WebRequest -Uri https://secure.eicar.org/eicar.com.txt -OutFile eicar.txt
> cp .\eicar.txt C:\Users\<USER_NAME>\Downloads
```

這將觸發 VirusTotal 查詢並生成警報。此外，主動響應腳本會自動刪除該文件。

警報視覺化

您可以在 Wazuh 儀表板中查看警報數據。為此，請轉到安全事件模塊並在搜索欄中添加過濾器以查詢警報。

Windows - rule.id：是 554、100092、553、87105 中的一個。