How Wazuh detects and responds to Mint Stealer | Wazuh

By Benjamin Nworah

Published: 2024-09-26 · Archived: 2026-04-05 14:37:50 UTC

Mint Stealer is a Python-based malware that steals data from web browsers, cryptocurrency wallets, VPN clients, mail clients, game applications, and more. Mint Stealer is sold as a malware-as-a-service (MaaS), designed to covertly exfiltrate sensitive information from infected Windows endpoints to a command and control (C2) server.

Mint Stealer uses encryption and obfuscation techniques to evade detection and enhance its effectiveness on infected endpoints. This malware is sold on multiple dedicated websites and with support provided to users through Telegram.

This blog post demonstrates how Wazuh detects and responds to Mint Stealer malware on infected Windows endpoints.

Behavioral analysis of Mint Stealer

Mint Stealer malware exhibits the following behaviors during execution.

- Mint Stealer extracts and uses content from its resource section as the next stage payload.
- The malware creates a folder in the `C:\Users\<USER_NAME>\AppData\Local\Temp` folder. The folder created starts with `onefile` followed by the process ID of `Setup.exe` and the system time retrieved using the `GetSystemTimeAsFileTime` API.
- It creates DLL (Dynamic link library) files, Pyd (Python dynamic modules) files, and an executable file usually named `vadimloader.exe` in the `C:\Users\<USER_NAME>\AppData\Local\Temp\onefile_<PID>_<SYSTEM_TIME>` folder.
- Mint Stealer uses `Setup.exe` to run the executable file as a child process. The executable file reads all the files in `C:\Users\<USER_NAME>\AppData\Local\Temp\onefile_<PID>_<SYSTEM_TIME>` and also loads the required libraries and code into the process memory for its operation.
- The malware collects information from web browsers, cryptocurrency wallets, VPN clients, FTP clients, messaging applications, and clipboards, among others. Mint Stealer also collects system and clipboard information using wmic and PowerShell commands respectively.
- It creates a folder inside `C:\Users\<USER_NAME>\AppData\Local\Temp\onefile_<PID>_<SYSTEM_TIME>` that starts with `Save-` followed by a randomly generated string. It saves all stolen data into this folder.
- Mint Stealer compresses the `Save-<RANDOMLY_GENERATED_STRING>` into a ZIP archive with a name that starts with `Save-` followed by a different random string.

Infrastructure

We use the following infrastructure to demonstrate how to detect and respond to Mint Stealer with Wazuh**:**

- **A pre-built, ready-to-use Wazuh OVA 4.9.0**: Follow this guide to download the virtual machine (VM). This VM hosts the Wazuh central components (Wazuh server, Wazuh indexer, and Wazuh dashboard)
- **A Windows 10 endpoint**: The victim endpoint has the Wazuh agent 4.9.0 installed and enrolled to the Wazuh server. Refer to the following guide to install the Wazuh agent.

Detection with Wazuh

In this blog post, we use the following techniques to detect the presence of Mint Stealer on a Windows 10 endpoint.

- **Wazuh detection rules**: This technique is used to detect malicious activities performed by Mint Stealer malware.
- **YARA integration with Wazuh**: This technique is used to detect the presence of Mint Stealer malware and remove it before it can do any damage.

Wazuh detection rules

We use Sysmon to monitor several system events and create rules on the Wazuh server to detect the malicious activities performed by Mint Stealer malware.

Windows endpoint

Perform the following steps to configure Sysmon on the monitored endpoint and forward logs in the Sysmon event channel to the Wazuh server for analysis.

1. Download Sysmon from the Microsoft Sysinternals page.

2. Extract the compressed Sysmon file to your preferred location.

3. Download the Sysmon configuration file – sysmonconfig.xml using PowerShell. Replace `<SYSMON_EXECUTABLE_PATH>` with the path to your Sysmon executable.

> wget -Uri https://wazuh.com/resources/blog/emulation-of-attack-techniques-and-detection-with-wazuh/sysmonconfig.xml -OutFile <SYSMON_EXECUTABLE_PATH>\sysmonconfig.xml

> wget -Uri https://wazuh.com/resources/blog/emulation-of-attack-techniques-and-detection-with-wazuh/sysmonconfig.xml -OutFile <SYSMON_EXECUTABLE_PATH>\sysmonconfig.xml

```
> wget -Uri https://wazuh.com/resources/blog/emulation-of-attack-techniques-and-detection-with-wazuh
```

4. Switch to the directory with the Sysmon executable. Run the command below to install and start Sysmon using PowerShell with Administrator privileges:

> .\Sysmon64.exe -accepteula -i .\sysmonconfig.xml

> .\Sysmon64.exe -accepteula -i .\sysmonconfig.xml

```
> .\Sysmon64.exe -accepteula -i .\sysmonconfig.xml
```

5. Add the following configuration within the `<ossec_config>` block of the `C:\Program Files (x86)\ossec-agent\ossec.conf` file to forward Sysmon events to the Wazuh server:

<!-- Configure Wazuh agent to receive events from Sysmon -->

<location>Microsoft-Windows-Sysmon/Operational</location>

<log_format>eventchannel</log_format>

<!-- Configure Wazuh agent to receive events from Sysmon --> <localfile> <location>Microsoft-Windows-Sysmon/Operational</location> <log_format>eventchannel</log_format> </localfile>

```
<!-- Configure Wazuh agent to receive events from Sysmon -->
<localfile>
  <location>Microsoft-Windows-Sysmon/Operational</location>
  <log_format>eventchannel</log_format>
</localfile>
```

6. Restart the Wazuh agent for the changes to take effect:

> Restart-Service -Name wazuh

> Restart-Service -Name wazuh

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

Wazuh server

Perform the following steps to configure rules to detect malicious activities of the Mint Stealer malware.

1. Create a new file `mint_stealer_malware.xml` in the `/var/ossec/etc/rules/` directory:

touch /var/ossec/etc/rules/mint_stealer_malware.xml

touch /var/ossec/etc/rules/mint_stealer_malware.xml

```
# touch /var/ossec/etc/rules/mint_stealer_malware.xml
```

2. Edit the file `/var/ossec/etc/rules/mint_stealer_malware.xml` and include the following detection rules for Mint Stealer malware:

<group name="windows,sysmon,mint_stealer,">

<!-- Mint Stealer creates a malicious executable file -->

<rule id="100190" level="8">

<field name="win.system.eventID">11</field>

<field name="win.eventdata.image" type="pcre2">(?i)\\\\.+exe</field>

<field name="win.eventdata.targetFilename" type="pcre2">(?
i)\\\\Local\\\\Temp\\\\onefile_\d+_\d+\\\\.+exe</field>

<description>Possible Mint Stealer malware detected. Malware creates a malicious executable $(win.eventdata.targetFilename).</description>

<!-- Mint Stealer loads DLL or Pyd files -->

<rule id="100191" level="8">

<field name="win.eventdata.image" type="pcre2">(?i)\\\\.+exe</field>

<field name="win.eventdata.imageLoaded" type="pcre2">(?i)\\\\Local\\\\Temp\\\\onefile_\d+_\d+\\\\.+(dll|pyd)
</field>

<description>Possible Mint Stealer malware detected. $(win.eventdata.imageLoaded) file loaded by $(win.eventdata.image).</description>

<!-- Mint Stealer gathers victim host information using wmic command -->

<rule id="100192" level="10">

<match type="pcre2">(?i)\\\\cmd.exe /c \\"wmic (os|csproduct|cpu|computersystem) get </match>

<description>Possible Mint Stealer malware detected. Malware steals system information using wmic command.
</description>

<!-- Mint Stealer attempts to steal clipboard data using PowerShell -->

<rule id="100193" level="8" ignore="1200">

<field name="win.eventdata.parentImage" type="pcre2">(?i)\\\\.+exe</field>

<match type="pcre2">(?i)powershell get-clipboard</match>

<description>Possible Mint Stealer malware detected. Malware attempts to steal clipboard data using PowerShell.
</description>

<group name="windows,sysmon,mint_stealer,"> <!-- Mint Stealer creates a malicious executable file --> <rule id="100190" level="8"> <if_sid>92213</if_sid> <field name="win.system.eventID">11</field> <field name="win.eventdata.image" type="pcre2">(?i)\\\\.+exe</field> <field name="win.eventdata.targetFilename" type="pcre2">(?i)\\\\Local\\\\Temp\\\\onefile_\d+_\d+\\\\.+exe</field> <description>Possible Mint Stealer malware detected. Malware creates a malicious executable $(win.eventdata.targetFilename).</description>

</rule> <!-- Mint Stealer loads DLL or Pyd files --> <rule id="100191" level="8"> <if_sid>61609</if_sid> <field name="win.eventdata.image" type="pcre2">(?i)\\\\.+exe</field> <field name="win.eventdata.imageLoaded" type="pcre2">(?i)\\\\Local\\\\Temp\\\\onefile_\d+_\d+\\\\.+(dll|pyd)</field> <description>Possible Mint Stealer malware detected. $(win.eventdata.imageLoaded) file loaded by $(win.eventdata.image).</description> <mitre> <id>T1574.002</id> </mitre> </rule> <!-- Mint Stealer gathers victim host information using wmic command --> <rule id="100192" level="10"> <if_sid>92032</if_sid> <match type="pcre2">(?i)\\\\cmd.exe /c \\"wmic (os|csproduct|cpu|computersystem) get </match> <description>Possible Mint Stealer malware detected. Malware steals system information using wmic command.</description> <mitre> <id>T1592</id> </mitre> </rule> <!-- Mint Stealer attempts to steal clipboard data using PowerShell --> <rule id="100193" level="8" ignore="1200"> <if_sid>92021</if_sid> <field name="win.eventdata.parentImage" type="pcre2">(?i)\\\\.+exe</field> <match type="pcre2">(?i)powershell get-clipboard</match> <description>Possible Mint Stealer malware detected. Malware attempts to steal clipboard data using PowerShell.</description> <mitre> <id>T1115</id> </mitre> </rule> </group>

```
<group name="windows,sysmon,mint_stealer,">

<!-- Mint Stealer creates a malicious executable file -->
  <rule id="100190" level="8">
    <if_sid>92213</if_sid>
    <field name="win.system.eventID">11</field>
    <field name="win.eventdata.image" type="pcre2">(?i)\\\\.+exe</field>
    <field name="win.eventdata.targetFilename" type="pcre2">(?i)\\\\Local\\\\Temp\\\\onefile_\d+_\d+
    <description>Possible Mint Stealer malware detected. Malware creates a malicious executable $(wi
  </rule>

<!-- Mint Stealer loads DLL or Pyd files -->
  <rule id="100191" level="8">
    <if_sid>61609</if_sid>
    <field name="win.eventdata.image" type="pcre2">(?i)\\\\.+exe</field>
    <field name="win.eventdata.imageLoaded" type="pcre2">(?i)\\\\Local\\\\Temp\\\\onefile_\d+_\d+\\\
    <description>Possible Mint Stealer malware detected. $(win.eventdata.imageLoaded) file loaded by
    <mitre>
      <id>T1574.002</id>
    </mitre>
  </rule>

<!-- Mint Stealer gathers victim host information using wmic command -->

  <rule id="100192" level="10">
    <if_sid>92032</if_sid>
    <match type="pcre2">(?i)\\\\cmd.exe /c \\"wmic (os|csproduct|cpu|computersystem) get </match>
    <description>Possible Mint Stealer malware detected. Malware steals system information using wmi
    <mitre>
      <id>T1592</id>
    </mitre>
```

```
  </rule>

<!-- Mint Stealer attempts to steal clipboard data using PowerShell -->

  <rule id="100193" level="8" ignore="1200">
    <if_sid>92021</if_sid>
    <field name="win.eventdata.parentImage" type="pcre2">(?i)\\\\.+exe</field>
    <match type="pcre2">(?i)powershell get-clipboard</match>
    <description>Possible Mint Stealer malware detected. Malware attempts to steal clipboard data us
    <mitre>
      <id>T1115</id>
    </mitre>
  </rule>
</group>
```

The following rule IDs are triggered when Wazuh detects the malicious activities of the Mint Stealer malware:

- Rule ID `100190` is triggered when Mint Stealer creates a malicious executable file.
- Rule ID `100191` is triggered when Mint Stealer loads DLL or Pyd files.
- Rule ID `100192` is triggered when the malware gathers information like CPU, OS name, system name from the victim endpoint using wmic.
- Rule ID `100193` is triggered when the malware attempts to steal clipboard information from the victim endpoint using PowerShell.

3. Restart the Wazuh manager for the changes to take effect:

sudo systemctl restart wazuh-manager

sudo systemctl restart wazuh-manager

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

Visualizing alerts on the Wazuh dashboard

The alerts below are generated on the Wazuh dashboard when the Mint Stealer malware infects the victim endpoint. Perform the following steps to view the alerts on the Wazuh dashboard.

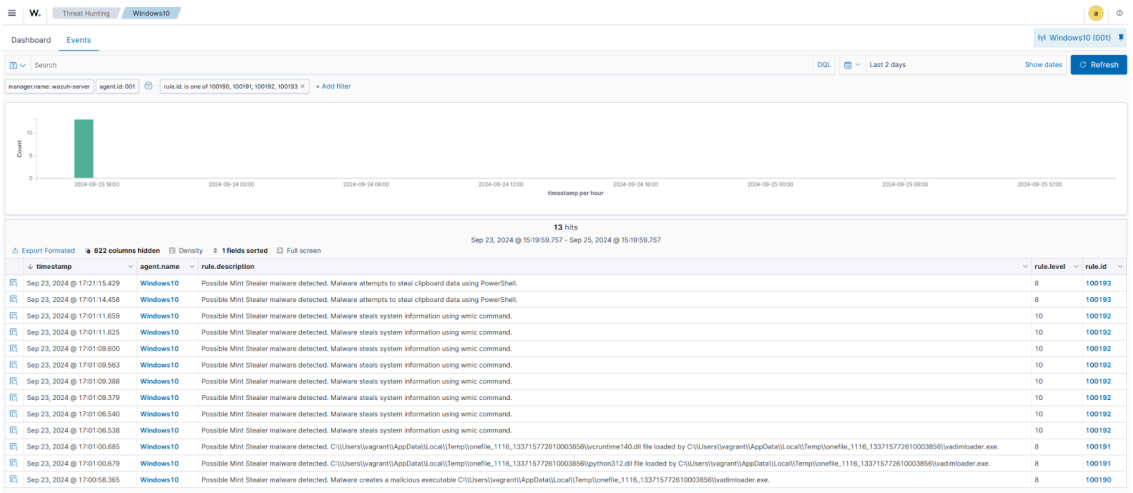
1. Navigate to **Threat intelligence** > **Threat Hunting**.

2. Click **+ Add filter**. Then, filter by `rule.id` in the **Field** field.

3. Filter for `is one of` in the **Operator** field.

4. Filter for `100190`, `100191`, `100192` and `100193` in the **Values** field.

5. Click **Save**.



YARA integration with Wazuh

YARA is an open source and multi-platform tool that identifies and classifies malware samples based on their textual or binary patterns. In this blog post, we use the Wazuh Active Response module to automatically execute a YARA scan on files added or modified in the `Downloads` folder of the monitored Windows endpoint.

Windows endpoint

To download and install YARA, we require the following packages installed on the victim endpoint:

- Python v 3.8.7 or later (with pip pre-installed).
- Microsoft Visual C++ 2015 Redistributable.

After installing the above packages, perform the steps below to download the YARA executable:

1. Launch PowerShell with administrator privileges and download YARA:

> Invoke-WebRequest -Uri https://github.com/VirusTotal/yara/releases/download/v4.5.2/yara-v4.5.2-2326-win64.zip -OutFile v4.5.2-2326-win64.zip

> Invoke-WebRequest -Uri https://github.com/VirusTotal/yara/releases/download/v4.5.2/yara-v4.5.2-2326-win64.zip -OutFile v4.5.2-2326-win64.zip

```
> Invoke-WebRequest -Uri https://github.com/VirusTotal/yara/releases/download/v4.5.2/yara-v4.5.2-232
```

2. Extract the YARA executable:

> Expand-Archive v4.5.2-2326-win64.zip

> Expand-Archive v4.5.2-2326-win64.zip

```
> Expand-Archive v4.5.2-2326-win64.zip
```

3. Create a folder named `C:\Program Files (x86)\ossec-agent\active-response\bin\yara\` and copy the YARA binary into it:

> mkdir 'C:\Program Files (x86)\ossec-agent\active-response\bin\yara\'

> cp .\v4.5.2-2326-win64\yara64.exe 'C:\Program Files (x86)\ossec-agent\active-response\bin\yara\'

> mkdir 'C:\Program Files (x86)\ossec-agent\active-response\bin\yara\' > cp .\v4.5.2-2326-win64\yara64.exe 'C:\Program Files (x86)\ossec-agent\active-response\bin\yara\'

```
> mkdir 'C:\Program Files (x86)\ossec-agent\active-response\bin\yara\'
> cp .\v4.5.2-2326-win64\yara64.exe 'C:\Program Files (x86)\ossec-agent\active-response\bin\yara\'
```

Perform the steps below to download YARA rules:

1. Using the same PowerShell terminal launched earlier, install `valhallaAPI` using the pip utility:

> pip install valhallaAPI

> pip install valhallaAPI

```
> pip install valhallaAPI
```

2. Create the file `download_yara_rules.py` and copy the following script into it:

from valhallaAPI.valhalla import ValhallaAPI

v = ValhallaAPI(api_key="1111111111111111111111111111111111111111111111111111111111111111")

response = v.get_rules_text()

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

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)

```
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)
```

3. Download YARA rules and copy them to the `C:\Program Files (x86)\ossec-agent\active-response\bin\yara\rules\` folder:

> python 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\'

> python 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\'

```
> python 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\'
```

4. Edit the file `C:\Program Files (x86)\ossec-agent\active-response\bin\yara\rules\yara_rules.yar` and add the following YARA rule to detect the Mint Stealer malware:

Author = "Benjamin Nworah"

Description = "Detect Mint Stealer malware"

Hash1 = "1064ab9e734628e74c580c5aba71e4660ee3ed68db71f6aa81e30f148a5080fa" // SHA-256 Hash

Hash2 = "cc93a4627a459d505c46de6fac342f856fb8f95b6a4fdcbd5e48be59aa4cbb7b" // SHA-256 Hash

$a2 = "GetSystemTimeAsFileTime"

rule MintStealer { meta: Author = "Benjamin Nworah" Description = "Detect Mint Stealer malware" Date = "13-09-2024" Hash1 = "1064ab9e734628e74c580c5aba71e4660ee3ed68db71f6aa81e30f148a5080fa" // SHA-256 Hash Hash2 = "cc93a4627a459d505c46de6fac342f856fb8f95b6a4fdcbd5e48be59aa4cbb7b" // SHA-256 Hash strings: $a1 = "FindResource" $a2 = "GetSystemTimeAsFileTime" $a3 = /NUITKA.{1,15}/ condition: all of ($a*) }

```
rule MintStealer
{
meta:
        Author = "Benjamin Nworah"
        Description = "Detect Mint Stealer malware"
        Date = "13-09-2024"
        Hash1 = "1064ab9e734628e74c580c5aba71e4660ee3ed68db71f6aa81e30f148a5080fa" // SHA-256 Hash
        Hash2 = "cc93a4627a459d505c46de6fac342f856fb8f95b6a4fdcbd5e48be59aa4cbb7b" // SHA-256 Hash

    strings:
        $a1 = "FindResource"
        $a2 = "GetSystemTimeAsFileTime"
        $a3 = /NUITKA.{1,15}/

    condition:
```

```
        all of ($a*)
}
```

5. Edit the Wazuh agent file `C:\Program Files (x86)\ossec-agent\ossec.conf` and add the below configuration within the `<syscheck>` block to monitor the `Downloads` folders of all users in real-time:

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

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

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

Note: In this blog post, the `Downloads` folders of all users are monitored. However, you can configure other folders you intend to monitor.

6. Create a batch file yara.bat in the `C:\Program Files (x86)\ossec-agent\active-response\bin\` folder. The Wazuh active response module executes this file to initiate YARA scans for malware detection and removal:

:: This script deletes Mint Stealer malware and other malicious files matched by the YARA Rules

setlocal enableDelayedExpansion

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

SET log_file_path="%programfiles%\ossec-agent\active-response\active-responses.log"

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

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

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

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

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%

:: Deleting the scanned file.

del /f "%syscheck_file_path%"

echo wazuh-yara: INFO - Successfully deleted: %%a >> %log_file_path%

:: This script deletes Mint Stealer malware and other malicious files matched by the YARA Rules @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) 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% :: Deleting the scanned file. del /f "%syscheck_file_path%" echo wazuh-yara: INFO - Successfully deleted: %%a >> %log_file_path%) exit /b

```
:: This script deletes Mint Stealer malware and other malicious files matched by the YARA Rules
@echo off
setlocal enableDelayedExpansion
reg Query "HKLM\Hardware\Description\System\CentralProcessor\0" | find /i "x86" > NUL && SET OS=32BI
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-a
SET syscheck_file_path=%%F
)
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.ya
echo %syscheck_file_path% >> %log_file_path%
for /f "delims=" %%a in ('powershell -command "& \"%yara_exe_path%\" \"%yara_rules_path%\" \"%syschec
    echo wazuh-yara: INFO - Scan result: %%a >> %log_file_path%
    :: Deleting the scanned file.
    del /f "%syscheck_file_path%"
    echo wazuh-yara: INFO - Successfully deleted: %%a >> %log_file_path%
)
exit /b
```

7. Restart the Wazuh agent using PowerShell for the changes to take effect:

> Restart-Service -Name wazuh

> Restart-Service -Name wazuh

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

Wazuh server

Perform the following steps to configure custom decoders, rules, and the Active Response module on the Wazuh server.

1. Edit the file `/var/ossec/etc/decoders/local_decoder.xml` on the Wazuh server and include the following decoders:

<!-- The decoders parse logs from the YARA scans -->

<decoder name="yara_decoder">

<prematch>wazuh-yara:</prematch>

<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 name="yara_decoder1">

<parent>yara_decoder</parent>

<regex>wazuh-yara: (\S+) - Successfully deleted: (\S+) (\S+)</regex>

<order>log_type, yara_rule, yara_scanned_file</order>

<!-- The decoders parse logs from the YARA scans --> <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> <decoder name="yara_decoder1"> <parent>yara_decoder</parent> <regex>wazuh-yara: (\S+) - Successfully deleted: (\S+) (\S+)</regex> <order>log_type, yara_rule, yara_scanned_file</order> </decoder>

```
<!-- The decoders parse logs from the YARA scans -->
<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>

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

2. Edit the file `/var/ossec/etc/rules/local_rules.xml` on the Wazuh server and include the following rules:

<!-- File added to the Downloads folder -->

<group name= "syscheck,">

<rule id="100028" level="7">

<field name="file" type="pcre2">(?i)C:\\Users.+Downloads</field>

<description>File modified in the Downloads folder.</description>

<!-- File modified in the Downloads folder -->

<rule id="100029" level="7">

<field name="file" type="pcre2">(?i)C:\\Users.+Downloads</field>

<description>File added to the Downloads folder.</description>

<!-- Rule for the decoder (yara_decoder) -->

<rule id="100194" level="0">

<decoded_as>yara_decoder</decoded_as>

<description>Yara grouping rule</description>

<!-- YARA scan detects a positive match -->

<rule id="100195" level="12">

<match type="pcre2">wazuh-yara: INFO - Scan result: </match>

<description>File "$(yara_scanned_file)" is a positive match. Yara rule: $(yara_rule)</description>

<rule id="100196" level="12">

<match type="pcre2">wazuh-yara: INFO - Successfully deleted: </match>

<description>Successfully removed "$(yara_scanned_file)". YARA rule: $(yara_rule)</description>

<!-- File added to the Downloads folder --> <group name= "syscheck,"> <rule id="100028" level="7"> <if_sid>550</if_sid> <field name="file" type="pcre2">(?i)C:\\Users.+Downloads</field> <description>File modified in the Downloads folder.</description> </rule> <!-- File modified in the Downloads folder --> <rule id="100029" level="7"> <if_sid>554</if_sid> <field name="file" type="pcre2">(?i)C:\\Users.+Downloads</field> <description>File added to the Downloads folder.</description> </rule> </group> <!-- Rule for the decoder (yara_decoder) --> <group name="yara,"> <rule id="100194" level="0"> <decoded_as>yara_decoder</decoded_as> <description>Yara grouping rule</description> </rule> <!-- YARA scan detects a positive match --> <rule id="100195" level="12"> <if_sid>100194</if_sid> <match type="pcre2">wazuh-yara: INFO - Scan result: </match> <description>File "$(yara_scanned_file)" is a positive match. Yara rule: $(yara_rule)</description> </rule> <rule id="100196" level="12"> <if_sid>100194</if_sid> <match type="pcre2">wazuh-yara: INFO - Successfully deleted: </match> <description>Successfully removed "$(yara_scanned_file)". YARA rule: $(yara_rule)</description> </rule> </group>

```
<!-- File added to the Downloads folder -->
<group name= "syscheck,">
  <rule id="100028" level="7">
    <if_sid>550</if_sid>
    <field name="file" type="pcre2">(?i)C:\\Users.+Downloads</field>
    <description>File modified in the Downloads folder.</description>
  </rule>
<!-- File modified in the Downloads folder -->
  <rule id="100029" level="7">
    <if_sid>554</if_sid>
    <field name="file" type="pcre2">(?i)C:\\Users.+Downloads</field>
    <description>File added to the Downloads folder.</description>
  </rule>
</group>
<!--  Rule for the decoder (yara_decoder) -->
<group name="yara,">
  <rule id="100194" level="0">
    <decoded_as>yara_decoder</decoded_as>
```

```
    <description>Yara grouping rule</description>
  </rule>
<!--  YARA scan detects a positive match -->
  <rule id="100195" level="12">
    <if_sid>100194</if_sid>
    <match type="pcre2">wazuh-yara: INFO - Scan result: </match>
    <description>File "$(yara_scanned_file)" is a positive match. Yara rule: $(yara_rule)</descripti
  </rule>
  <rule id="100196" level="12">
    <if_sid>100194</if_sid>
    <match type="pcre2">wazuh-yara: INFO - Successfully deleted: </match>
    <description>Successfully removed "$(yara_scanned_file)". YARA rule: $(yara_rule)</description>
  </rule>
</group>
```

3. Add the following configuration to the Wazuh server file `/var/ossec/etc/ossec.conf` within the `<ossec_config>` block:

<!-- The YARA batch script is executed when a file is added or modified in the Downloads folder monitored by Wazuh -->

<executable>yara.bat</executable>

<timeout_allowed>no</timeout_allowed>

<location>local</location>

<rules_id>100028,100029</rules_id>

<!-- The YARA batch script is executed when a file is added or modified in the Downloads folder monitored by Wazuh --> <command> <name>yara</name> <executable>yara.bat</executable> <timeout_allowed>no</timeout_allowed> </command> <active-response> <command>yara</command> <location>local</location> <rules_id>100028,100029</rules_id> </active-response>

```
<!-- The YARA batch script is executed when a file is added or modified in the Downloads folder moni
<command>
  <name>yara</name>
  <executable>yara.bat</executable>
  <timeout_allowed>no</timeout_allowed>
</command>
<active-response>
  <command>yara</command>
  <location>local</location>
  <rules_id>100028,100029</rules_id>
</active-response>
```

4. Restart the Wazuh manager for the changes to take effect:

systemctl restart wazuh-manager

systemctl restart wazuh-manager

```
# systemctl restart wazuh-manager
```

Visualizing alerts on the Wazuh dashboard

The Wazuh dashboard shows alerts when the Mint Stealer malware is downloaded to the `Downloads` folder of the victim endpoint. Perform the following steps to view the alerts on the Wazuh dashboard.

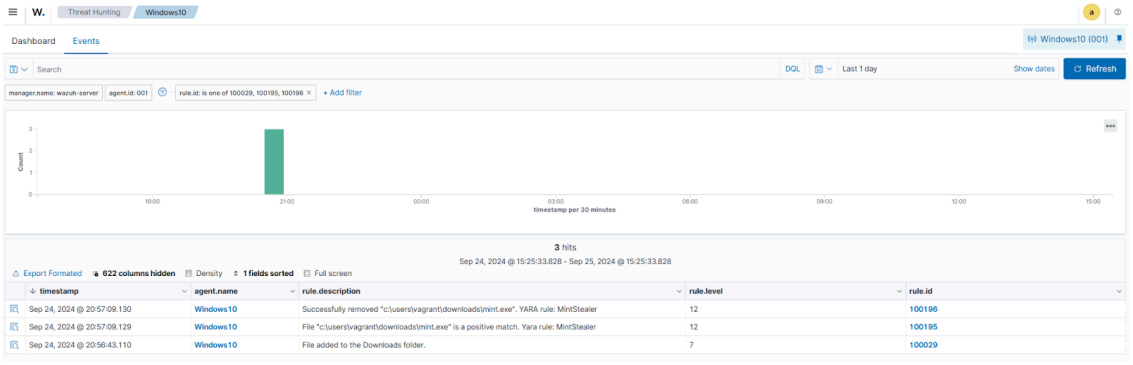
1. Navigate to **Threat intelligence** > **Threat Hunting**.

2. Click **+ Add filter**. Then filter by `rule.id` in the **Field** field.

3. Filter for `is one of` in the **Operator** field.

4. Filter for `100029`, `100195`, and `100196` in the **Values** field.

5. Click **Save**.



Conclusion

In this blog post, we used Sysmon integration with Wazuh to detect the malicious activities performed by Mint Stealer malware. We also used YARA integration with Wazuh to detect and remove this malware once it is downloaded to an endpoint.

Wazuh is a free and open source enterprise-ready security platform for uncovering security threats, incident response, and compliance. Wazuh integrates with third-party technologies. We also have a growing community where users are supported. To learn more about Wazuh, please check out our documentation and blog posts.

References

- Mint Stealer: A Comprehensive Study of a Python-Based Information Stealer
- Mint Stealer: New MaaS Malware Threatens Confidential Data

Source: https://wazuh.com/blog/how-wazuh-detects-and-responds-to-mint-stealer/