

SOC & Automation Report

Prepared by: Isabelle Jaber

Date: August 26th, 2024

Table of Contents:

OVERVIEW OF THE SELECTED IOC.....	3
Introduction.....	3
Key Characteristics.....	3
SETTING UP WAZUH AND CREATING A RULE.....	4
Installation and Configuration.....	4
Rule Creation.....	4
SHUFFLER.IO CONFIGURATION FOR AUTOMATED RESPONSE.....	6
Integration with Wazuh.....	6
Automated Response Setup.....	7
RESULTS AND OBSERVATIONS FROM THREAT SIMULATION.....	10
Threat Simulation Scenario.....	10
Detection and Response.....	12
Observations and Improvements.....	12
CONCLUSION.....	14

OVERVIEW OF THE SELECTED IOC

Introduction

The IP address “112.248.107.4” was the selected Indicator of Compromise (IOC). As such, any network traffic going to or coming from this IP address should be flagged immediately and investigated to prevent unauthorized access to and/or exfiltration of company data.

Key Characteristics

According to AlienVault, this IP address is located in China. On VirusTotal 11 out of 94 security vendors flagged this IP address as malicious. According to a source on VirusTotal (Cluster25), “this IPV4 is used by MOZI. MOZI is a peer-to-peer (P2P) botnet network that utilizes the distributed hash table (DHT) system, implementing its own extended DHT. The distributed and decentralized lookup mechanism provided by DHT enables Mozi to hide C2 communication behind a large amount of legitimate DHT traffic. The use of DHT allows Mozi to quickly grow as P2P network and often it’s difficult to detect its C2 communication. Mozi infects systems usually by brute-forcing SSH and Telnet ports. It then blocks those ports so that it is not overwritten by other malicious actors or malware.”

SETTING UP WAZUH AND CREATING A RULE

Installation and Configuration

To install and configure the Wazuh server in VirtualBox on a MacOS machine, the team followed these steps:

- Download the Wazuh .ova file from the documentation [here](#).
- Import it into Virtual Box.
- Go to “File” > “Import Appliance” and select the .ova file.
- Once it is finished importing, set the VMSVGA graphic controller.
 - Select the imported VM.
 - Click Settings > Display
 - In “Graphics Controller”, select the VMSVGA option.
- Open the VM and, when prompted, access it using the following user and password:
 - user: wazuh-user
 - password: wazuh
- Shortly after starting the VM, the Wazuh dashboard can be accessed from the web interface by using the following credentials:
 - URL: `https://<wazuh_server_ip>`
 - user: admin
 - password: admin
- You can find `<wazuh_server_ip>` by typing the following command in the VM:
 - `ip a`

Rule Creation

Initially, an attempt was made to create a CDB list to hold the key : value pair of “112.248.107.4” : “Found” and create a rule that would test the incoming and outgoing IP address against the list and, if the associated value was “Found,” it would trigger the alert associated with the rule (this condition was contained within the `<list>` tag). It would have appeared as it does in *Image 1*.

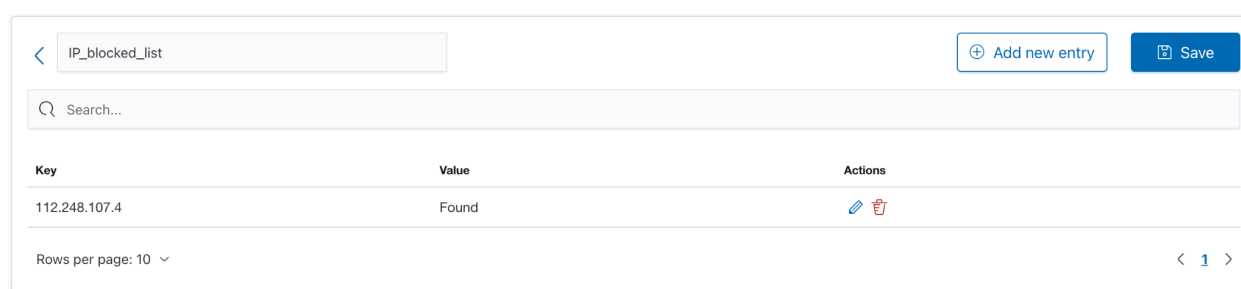


```

1 <!-- Local rules -->
2
3 <!-- Modify it at your will. -->
4 <!-- Copyright (C) 2015, Wazuh Inc. -->
5
6
7 <group name="id">
8   <rule id="100001" level="16">
9     <list field="malicious.id" lookup="match_key_value" check_value="Found">etc/lists/IP_blocked_list</list>
10    <description>Suspicious IP Address Detected</description>
11    <group>malicious_ip</group>
12  </rule>
13 </group>
14
15
16
17
18

```

Image 1: Custom Rule using CDB List



Key	Value	Actions
112.248.107.4	Found	 

Rows per page: 10

Image 2: CDB List for Malicious IP Addresses

However, in an attempt to save and create this rule, it was discovered that Wazuh seemed to be unable to find the IP_blocked_list CDB list. As a result, a different method was used to create the rule. Instead of linking the rule to a CDB list to check if the IP address should be allowed to interact with the system, the `<match>` tag explicitly checked for the malicious IP address within the rule as seen in *Image 3*.



```

1 <!-- Local rules -->
2
3 <!-- Modify it at your will. -->
4 <!-- Copyright (C) 2015, Wazuh Inc. -->
5
6
7 <group name="syslog, local">
8   <rule id="100001" level="16">
9     <description>Traffic from suspicious IP address</description>
10    <match>112.248.107.4</match>
11  </rule>
12 </group>

```

Image 3: Rule using <match> Tag to check for the Malicious IP Address

SHUFFLER.IO CONFIGURATION FOR AUTOMATED RESPONSE

Integration with Wazuh

1. Create a workflow in Shuffle and add a webhook trigger.
2. Copy the webhook URI generated and start the webhook.

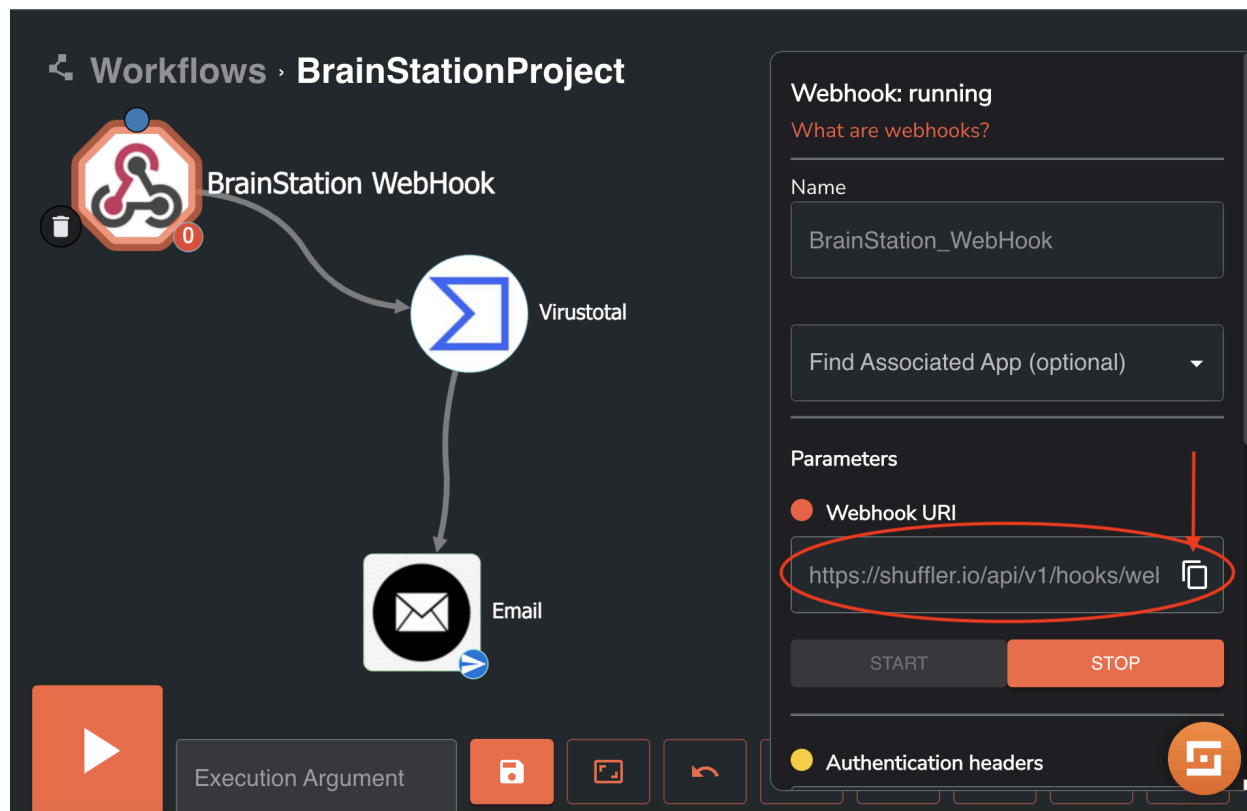


Image 4: Webhook URI

3. Edit the Wazuh server configuration file (`ossec.conf`) and add the Shuffle integration settings.
4. Specify the Shuffle webhook URI, rule ID, rule group, or alert level for events to be forwarded.



Image 5: Shuffler.io Integration in Wazuh `ossec.conf` file

Automated Response Setup

1. Then activate the Virustotal v3 App and add it to the workflow, so it occurs after the email is sent.
 - a. Configure it to get a report on the IP address found by the Wazuh rule and write the result to a JSON file.
 - b. Authentication using Isabelle Jaber's VirusTotal API key was required.

Workflows › BrainStationProject

BrainStation WebHook

Virustotal

Email

Name: Virustotal Delay: 0

Authentication: Latest 1.1.0 Auth for Vir... +

Find Actions: Get an ip address report x

Parameters: Authentication fields are hidden

Ip: 112.248.107.4 +

Headers

Execution Argument

Image 6a: Shuffler Workflow VirusTotal Configurations

Workflows › BrainStationProject

BrainStation WebHook

Virustotal

Email

Headers: Content-Type=application/json Accept=application/json +

Queries: view=basic&redirect=test +

Ssl verify: False

To file: True

Execution Argument

Image 6b: Shuffler Workflow VirusTotal Configurations

2. Activate the Email App and add it to the workflow so it occurs after the WebHook is triggered.
 - a. Configure this action to send an email notifying Isabelle Jaber that the malicious IP was detected.

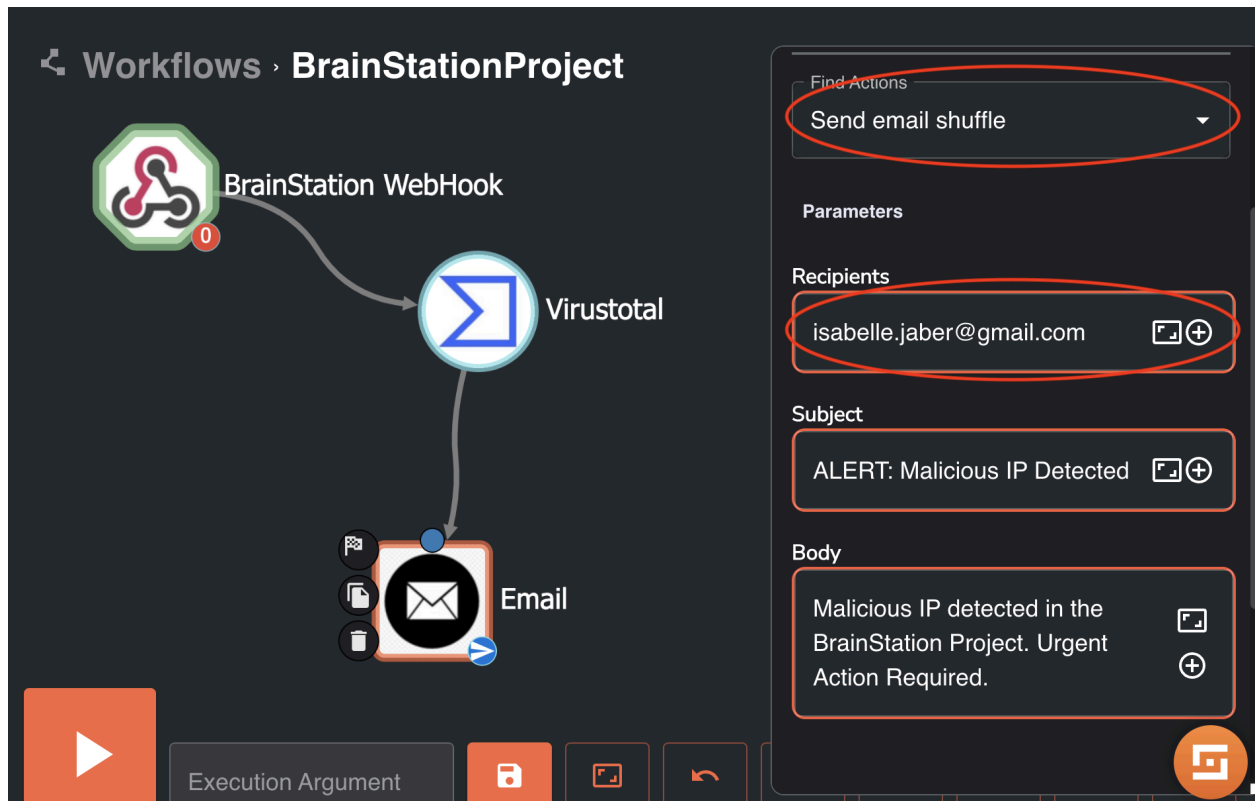


Image 7: Shuffler Workflow Email Configurations

RESULTS AND OBSERVATIONS FROM THREAT SIMULATION

Threat Simulation Scenario

A complete threat simulation scenario could not occur, as the agent was unable to read/access the logs on the Kali Linux machine used to test the rule. To test this scenario, the `logger <112.248.107.4> command` was run, with no alerts appearing in Wazuh. To ensure that this log was being processed by the agent, this code was added to the `/var/ossec/etc/ossec.conf` file on the Kali machine.

```
<localfile>

<log_format>syslog</log_format>

**This tells Wazuh to interpret the incoming logs in a syslog format.**

<command>journalctl -f -n 0</command>

**This command continuously streams logs from the systemd journal.**

</localfile>
```

Code Snippet 1: Configurations Added to the ossec.conf File on the Kali Machine

The agent was then restarted and the `logger <112.248.107.4> command` was run again with no result. The next step to resolving this issue was to verify that the `logger` command was being logged correctly by `systemd`. To do this `sudo journalctl -f` was run to view the activities in the `systemd` journal and, before quitting out of the command, `logger "Test log entry from 192.168.1.100"` was run to see if it would appear in the logs. As there was no log printed, it was determined that there may be an issue with how `logger` or `systemd-journald` is functioning on the Kali machine. This was an issue that was beyond our team's capabilities to fix.

To test the rule without using the agent on the Kali machine, the "Ruleset Test" function in the `local_rules.xml` file was used to run a log with the malicious IP address listed as the source IP, along with other data gathered when the log was generated.

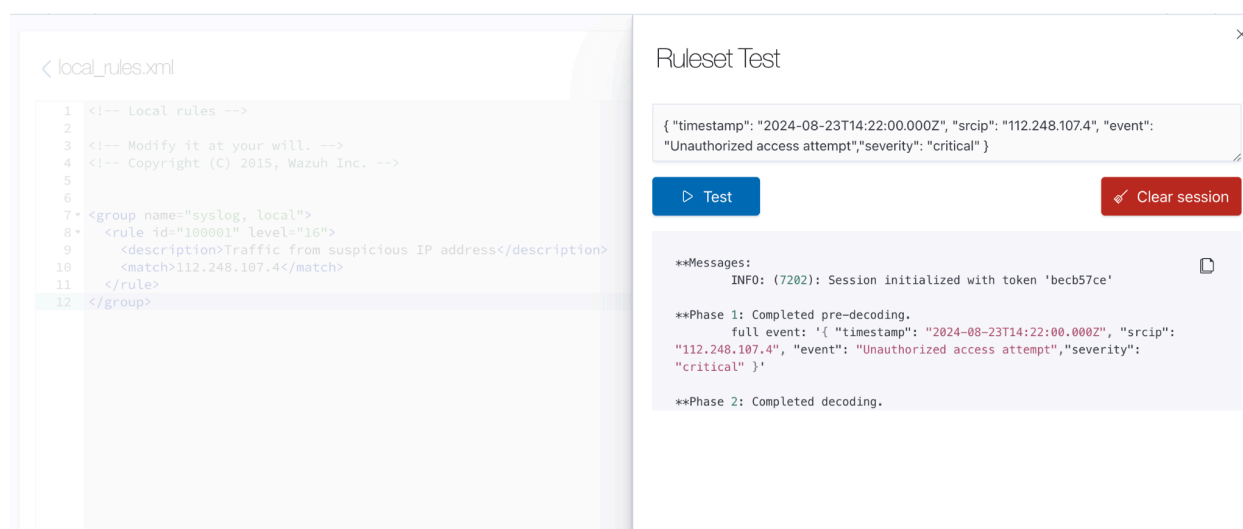


Image 8: Ruleset Test Log and Output

The complete result of the test shows that it generates the appropriate response within Wazuh.

****Messages:**

INFO: (7202): Session initialized with token 'becb57ce'

****Phase 1: Completed pre-decoding.**

full event: '{ "timestamp":
"2024-08-23T14:22:00.000Z", "srcip": "112.248.107.4",
"event": "Unauthorized access attempt", "severity":
"critical" }'

****Phase 2: Completed decoding.**

name: 'json'
event: 'Unauthorized access attempt'
severity: 'critical'
srcip: '112.248.107.4'
timestamp: '2024-08-23T14:22:00.000Z'

****Phase 3: Completed filtering (rules).**

id: '100001'
level: '16'

```

description: 'Traffic from suspicious IP address'
groups: '["syslog"," local"]'
firedtimes: '1'
mail: 'true'
**Alert to be generated.

```

Code Snippet 2: Full Ruleset Output

Unfortunately, this only tested the rule locally and did not generate an alert or trigger the Shuffler workflow as indicated by the last line of *Code Snippet 2*.

Detection and Response

As the team was unable to mimic an event that would trigger the rule, they were also unable to see how successful the integration between Wazuh and Shuffler.io was. As there were no errors when saving the integration configurations within the Wazuh `ossec.conf` file, there is currently no reason to believe that the integration was unsuccessful.

The way the rules and configurations were designed and implemented, should have resulted in Wazuh detecting any network logs that recorded any source or destination IP addresses defined by the rule. When this IP address is detected in the logs, Wazuh should generate an alert in its system, and then start the workflow linked to it in Shuffler.io. This workflow should have generated a JSON file with a VirusTotal report on the IP address for the SOC analyst to view when they begin investigating the alert. After the file is generated, it sends an email to isabelle.jaber@gmail.com (acting as a SOC analyst) to notify them of the alert.

Observations and Improvements

The biggest challenge in this simulation was trying to generate an alert for the created rule. It took a lot of time and energy on our team's part to try and discern what was preventing the alert from being generated, and, by the end of our investigation, we concluded that at least one problem contributing to this issue, which was that `logger` and `systemd-journald` were not communicating with each other as they should be which meant that the logs were not being stored or generated in the proper way or place for the Wazuh Agent to be able to detect an interaction with the malicious IP address.

Some other obstacles we encountered included connecting the Kali Agent and the Wazuh dashboard and linking the contents of a rule to a custom CDB list. Eventually, the Kali Agent was registered by and connected to the Wazuh dashboard. However, for future situations where this process will need to take place to launch a new agent on a new machine, the team must investigate why it took so long for the agent to connect and if any network or configuration settings hindered the process. Most importantly, the team should determine why the link to the CDB list in the rule wasn't able to connect to the target file. For future incident response and threat detection, it would be best if the team could create a running CDB list of multiple

malicious IP addresses and not have to create individual rules and shuffler workflows to detect and manage each one. It follows from this that research should be done on how to pass the IP address detected in Wazuh to the Shuffler workflow. This would eliminate the hard coding of the specific IP address in the VirusTotal tool used to get the report on the IP address, and the same workflow could be used for a multitude of IP addresses.

CONCLUSION

Once the issues related to the Kali machine, Wazuh agent, and CDB lists are resolved, it is the conclusion of this team that, with properly designed workflows and rules, the integration of Wazuh and Shuffler.io will be very effective in threat detection, analysis, and intelligence. In the case of this scenario and attempted simulation, the team was able to set up a basic rule to test the integration of Wazuh and Shuffler.io. However, due to unforeseen configuration issues on the Kali Linux machine, the team was unable to launch a successful threat simulation. After investigating the problem, the team found that the source of the issue may have stemmed from a miscommunication between `logger` and `systemd-journald`, which would be necessary for the logs to be stored or generated properly for the Wazuh Agent to be able to detect an endpoint event involving the malicious IP address. Since the team was unable to resolve this issue they were forced to only test the rule itself, instead of the Wazuh-Shuffler.io integration. However, since the team followed all of the integration steps according to [Wazuh documentation](#), the team has yet to identify any reason why this workflow would not be triggered if the rule generated an alert. In the future, more preparation should take place to make sure that all of the necessary tools and configurations are working on all necessary machines. This could decrease the amount of time debugging, and lead to a more efficient process for testing new software integrations.