

CYBERSECURITY & SECURITY OPERATION CENTER

SOC Case

Daan Van Winkel

Inhoud

1. THE TOOLS	3
1.1. Wazuh (The SIEM/XDR): Detection & Monitoring	3
1.2. VirusTotal (The Threat Intel): Enrichment & Validation	3
1.3. Shuffle (The SOAR): Orchestration & Automation	3
1.4. IRIS (The DFIR Platform): Investigation & Case Management	3
1.5. Infection Monkey (The BAS): Adversary Emulation	3
1.6. Summary of the Defense Loop	4
2. THE ATTACKS	4
2.1. SSH Brute Force (using Infection Monkey)	4
2.2. Malware Download	4
2.3. Integration with your other tools	4
3. CONFIGURATION/DOWNLOAD GUIDE	5
3.1. Preperation	5
3.1.1. Wazuh	5
3.1.2. Shuffle	5
3.1.3. Iris	5
3.1.4. Infection Monkey	5
3.1.5. Ubuntu Server	5
3.2. Configuration	5
3.2.1. Installing the agent on our target	5
3.2.2. Integration of Shuffle in Wazuh	5
3.2.2.1. Step by Step	6
3.2.3. Shuffle to Discord	6
3.2.4. Wazuh → Shuffle → Iris	7
3.2.5. Shuffle → VirusTotal (IP Check)	7
3.2.6. VirusTotal malware check	8
3.2.7. IP Ban SSH Bruteforce	9
3.2.8. Infection Monkey	10
3.3. Testing	13
4. SUMMARY/REFLECTION	13
4.1. What I Learned	13
5. APPENDIX	14
5.1. Appendix A: Shuffle to Discord JSON (Wazuh Alerts)	14
5.2. Appendix B: Shuffle to Discord JSON (VirusTotal Analysis)	15
5.3. Appendix C: Shuffle to IRIS Alert JSON	16
5.4. Appendix D: Active Response Script (remove-threat.sh)	17

1. The tools

In a modern Security Operations Center (SOC), defense is no longer just about watching logs and alerting, it is about automated visibility and response. By integrating a SIEM, SOAR, and Threat Intelligence, you transform a passive monitoring system into an active defense architecture.

Here is how the four core tools in my setup work together to defend against the specific scenarios:

1.1. Wazuh (The SIEM/XDR): Detection & Monitoring

Wazuh acts as the "eyes" of your SOC. It collects and correlates logs from my endpoints in real time.

- **SSH Brute Force:** Wazuh monitors the agent/target checking their logs. If it detects a high frequency of failed login attempts from a single IP, it triggers an alert based on built-in rules.
- **Malware Download:** Using its **File Integrity Monitoring (FIM)** module, Wazuh detects when a new file is created in a sensitive directory (like Downloads).

1.2. VirusTotal (The Threat Intel): Enrichment & Validation

VirusTotal acts as the "brain" that validates if an alert is a real threat.

- **Validation:** When Wazuh detects a new file, it extracts the file's hash and **sends it to VirusTotal** via API.
- **Decision Making:** VirusTotal compares the hash against **68 antivirus engines**. If multiple engines flag it as malicious, it confirms the threat, allowing the SOC to proceed with high confidence.

1.3. Shuffle (The SOAR): Orchestration & Automation

Shuffle is the "connective tissue" that automates the workflow between tools.

- **Workflow Automation:** Once Wazuh generates an alert, Shuffle receives it via a webhook.
- **Execution:** With this it will send a webhook to Discord to send a message to a dedicated channel for Wazuh alerts and a separate one for IP verification through VirusTotal.
- **Alert Creation:** After it has done all of that it will send a webhook to Iris.

1.4. IRIS (The DFIR Platform): Investigation & Case Management

IRIS is the "workspace" where the **Digital Forensics and Incident Response (DFIR)** happen.

- **Case Tracking:** While Shuffle handles the automatic response, it also pushes all technical details (IPs, hashes, and VirusTotal reports) into IRIS which I can create a structured case with.
- **Final Review:** This ensures that even though the attack was blocked automatically, a human analyst can later review the full timeline of the incident for post-mortem analysis or deeper hunting.

1.5. Infection Monkey (The BAS): Adversary Emulation

Infection Monkey acts as the "test pilot" or **Adversary Emulator** for your SOC. It provides a safe way to simulate real-world attacks to ensure my defenses are working.

- **Attack Simulation:** For the **SSH Brute Force** scenario, Infection Monkey is configured to act as the attacker. It attempts to spread through the network by testing common credentials against my target machines, mimicking the behavior of a live worm or a malicious actor.
- **Defense Validation:** By launching a controlled attack, Infection Monkey allows you to verify that **Wazuh** is correctly logging the failed attempts and that your **Active Response** scripts are successfully blocking the simulated "infected" machine.

1.6. Summary of the Defense Loop

1. **Detect:** Wazuh identifies suspicious behavior (Brute force or file creation).
2. **Enrich:** VirusTotal provides the reputation of the file or IP.
3. **Respond:** Shuffle automates the alerting, Wazuh has built in rules for Active Response.
4. **Manage:** IRIS logs every step for a complete investigation record.

2. The attacks

2.1. SSH Brute Force (using Infection Monkey)

When a brute force attack occurs, Wazuh doesn't just alert, it takes immediate action to cut off the attacker.

- **The Trigger:** Wazuh monitors failed login attempts. When a specific rule (e.g., Rule ID 5763 for multiple failed logins) is triggered, it fires an Active Response.
- **The Action:** It automatically runs a script which adds the attacker's IP address to the local firewall.
- **Stateful Defense:** This is a **stateful** response, meaning the IP is blocked for a set period (3 minutes in my case) before the block is automatically lifted, preventing permanent accidental lockouts while stopping the immediate attack.

2.2. Malware Download

For malware, Active Response works in a "Check-then-Kill" loop with VirusTotal.

- **The Trigger:** The **File Integrity Monitoring (FIM)** module detects a new file in a monitored directory (I used /home/target).
- **The Verification:** Wazuh sends the file hash to **VirusTotal**. If VirusTotal returns a malicious verdict, Wazuh immediately fires the Active Response.
- **The Action:** A custom script (remove-threat.sh) is executed on the endpoint to the malicious file before it can be executed by a user.

2.3. Integration with other tools

- **Shuffle (SOAR):** While Active Response handles the *immediate* block, it simultaneously sends a webhook to **Shuffle**. Shuffle then carries out secondary tasks, like the alerting in **Discord**.
- **IRIS (DFIR):** Every action taken by the Active Response script including the timestamp, the blocked IP, and the deleted file path is logged and pushed to **IRIS**. This ensures that when you create the case in IRIS, you aren't just seeing a "threat," but a documented "threat that was already mitigated".

3. Configuration/Download guide

Now we will be heading into the boring part of this document. All the configuration I have done to come to this point.

3.1. Preperation

We must start at the beginning of even being able to configure, installation. After installation I went to each dashboard and did the steps noted in their documentation to get it working (setting user and/or password). There is more than enough explanation in the documentation that you already will have opened to install the software.

3.1.1. Wazuh

For the **installation** of **Wazuh** I made it very easy for myself, I used the **prebuilt vm**:
<https://documentation.wazuh.com/current/deployment-options/virtual-machine/virtual-machine.html>

3.1.2. Shuffle

For the **installation** of **Shuffle** it's a bit harder but there is also a guide for this that I followed:
<https://github.com/shuffle/shuffle/blob/main/.github/install-guide.md>

3.1.3. Iris

For the **installation** of **Iris** it is same difficulty if not easier than Shuffle, again with a guide and docker:
https://docs.dfir-iris.org/latest/getting_started/

3.1.4. Infection Monkey

This one was probably second easiest, I installed it on a Kali VM (which I downloaded prebuilt for VMWare) with the Appliance on their GitHub: <https://github.com/guardicore/monkey/releases/tag/v2.3.0>

3.1.5. Ubuntu Server

We will also need a **ubuntu server** as our **target** server. I trust you can download and install a ubuntu server VM on your own. I chose for 22.04 LTS Jammy as my version.

3.2. Configuration

3.2.1. Installing the agent on our target

There is a great tutorial on the Dashboard of your **Wazuh Server**. Its located at <https://<your-wazuh-ip>/app/endpoints-summary#/agents-preview/deploy> follow these and you will have an agent in no time.

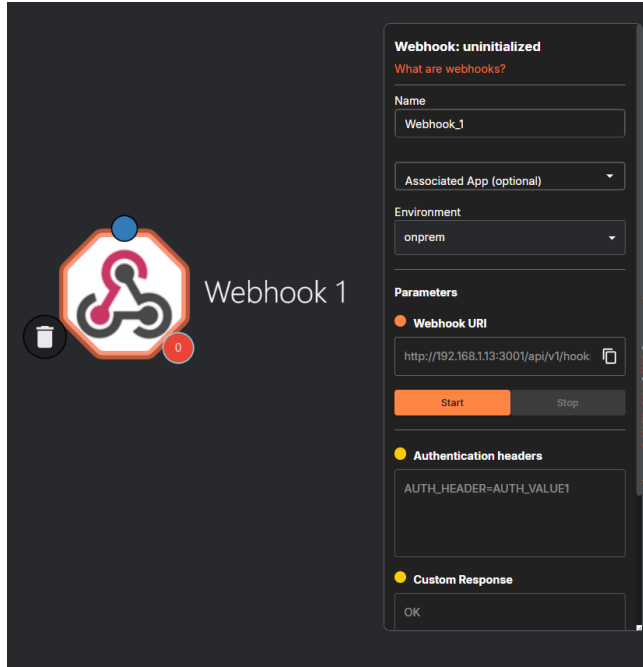
3.2.2. Integration of Shuffle in Wazuh

I struggled a bit with this one, mostly because I was too stubborn to ssh into the VM. I kept typing the **webhook** wrong that u get from **Shuffle**.

3.2.2.1. Step by Step

First, we need a workflow, creating a workflow can be done by going to the workflows dashboard and clicking "New workflow" as highlighted in the picture below. It will ask you for a name and description. These can be changed at any time.

Now, go to your shuffle dashboard and create a Webhook:



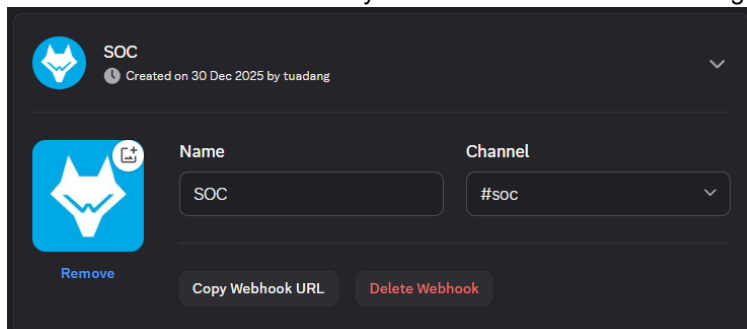
Copy the URL and go to your **Wazuh VM** and open `/var/ossec/etc/ossec.conf`. Place the following code block at the bottom still withing the `<ossec_config>`. **Change the hook_url** accoding to your own webhook url.

```
<integration>
  <name>shuffle</name>
  <hook_url>http://192.168.1.13:3001/api/v1/hooks/webhook_66a859dd-2274-4597-a707-6ff06305c235</hook_url>
  <alert_format>json</alert_format>
</integration>
```

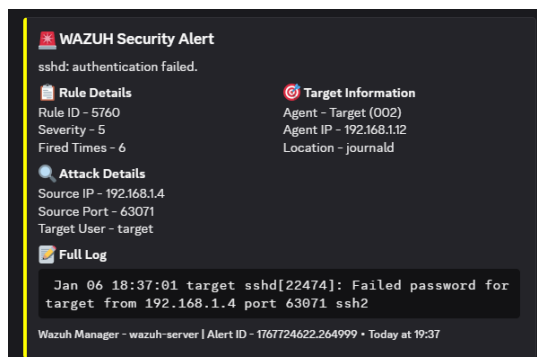
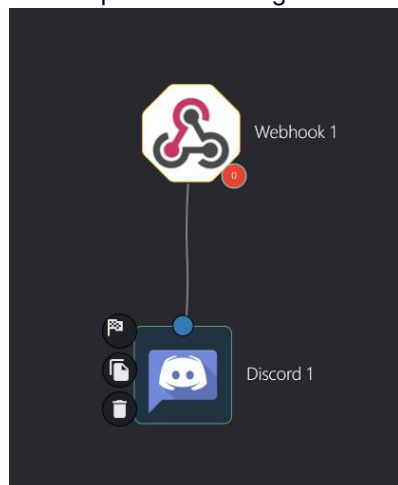
And now the webhook should be working.

3.2.3. Shuffle to Discord

Now I wanted to see my alerts come to live. So, I chose to do **Discord Webhooks** next, start with creating a webhook by going to Edit Channel → Integrations → Webhooks → New Webhook. Then you click the Webhook and name it however you want. I chose for the following



Now we first go to Shuffle → Apps → Discover Public Apps → Search Discord → Activate it.
In our workflow we can now drag the Discord app next to the webhook and connect them with the blue dot at the top when hovering over it. It should go from the Webhook to Discord.



Then we must configure our discord app. Configure everything along the lines of this screenshot to the right, for the body I created an embed you can see an example.

I will put the JSON that is in the body in the appendixes (Appendix A) since it would fill an entire paper. Now after all of this it should be working.

3.2.4. Wazuh → Shuffle → Iris

The alerts we get from Wazuh already go to Shuffle, now we will go from shuffle to Iris.

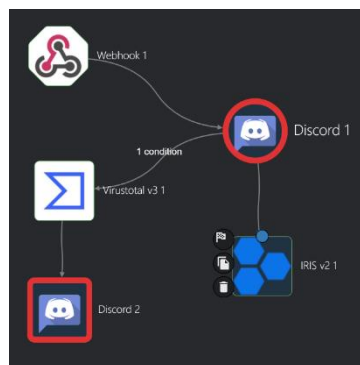
Same thing as with Discord install the app into Shuffle **make sure its Iris v2.**

Connect **Discord to Iris** since a Webhook can only go to 1 node. Now if we open Iris to configure you will see a lot of things. We need to find the action Add alert, then fill in the URL with the IP of your Iris server **MAKE SURE IT DOESN'T END IN A /**, otherwise it won't work. The body will be in the appendixes (Appendix C) like before. Now we see a field with ApiKey, this didn't work for me, I had to use the headers below it to authenticate the requests. So, I put "Authorization=Bearer <api-key-iris>", to get that ApiKey we go to the Iris Dashboard, click your user top left → My Settings → and copy the ApiKey.

3.2.5. Shuffle → VirusTotal (IP Check)

Same thing as with Iris and Discord download the VirusTotal v3.1 app.

Drag it next to the Discord and add a new Discord my layout is as follows



The condition that is on the arrow going from Discord 1 to VirusTotal is the following, you can get there by clicking the arrow and add condition then just copy what I put here. This checks if there is a source IP to check since we don't always have one. JSON for Discord Appendix B

3.2.6. VirusTotal malware check

We are done with Shuffle, finally, now back Wazuh. Well, more specifically the target VM
In the `/var/ossec/etc/ossec.conf` we must find the following block:

```
<!-- File integrity monitoring -->
<syscheck>
  <!-- Frequency that syscheck is executed default every 12 hours -->
  <frequency>43200</frequency>

  <scan_on_start>yes</scan_on_start>

  <!-- Directories to check (perform all possible verifications) -->
  <directories realtime="yes">/etc,/usr/bin,/usr/sbin,/home/target</directories>
  <directories realtime="yes">/bin,/sbin,/boot</directories>

  <!-- Files/directories to ignore -->
  <ignore>/etc/mtab</ignore>
  <ignore>/etc/hosts.deny</ignore>
  <ignore>/etc/mail/statistics</ignore>
  <ignore>/etc/random-seed</ignore>
  <ignore>/etc/random.seed</ignore>
  <ignore>/etc/adjtime</ignore>
  <ignore>/etc/httpd/logs</ignore>
  <ignore>/etc/utmpx</ignore>
  <ignore>/etc/wtmpx</ignore>
  <ignore>/etc/cups/certs</ignore>
  <ignore>/etc/dumpdates</ignore>
  <ignore>/etc/svc/volatile</ignore>
```

Here we only really added `/home/target` to the directories to check this makes it so that that directory also gets checked with the File Integrity Monitor.

Now then actually on our Wazuh VM:

Add the following rules to the Wazuh server `/var/ossec/etc/rules/local_rules.xml`

```
<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>
```

Now in `/var/ossec/etc/ossec.conf`

```
<integration>
  <name>virustotal</name>
  <api_key>[REDACTED]</api_key>
  <rule_id>100200,100201</rule_id>
  <alert_format>json</alert_format>
</integration>
```

All the way at the bottom add that block below the shuffle integration. Of course, with your own API key.

Then below that add this

```

<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>

```

Now our malware gets detected and it tries to execute a shell file which is not on the agent yet. We will now of course be adding it. Go back to the Target VM and make a file in “*/var/ossec/active-response/bin/*” named *remove-threat.sh* with the contents being in the appendix again. Appendix D

After adding the file run these commands so that Wazuh can run it:

```

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

```

3.2.7. IP Ban SSH Bruteforce

Back on the Wazuh VM add the following rules to the Wazuh server */var/ossec/etc/rules/local_rules.xml*

```

<group name="syscheck,pci_dss_11.5,nist_800_53_S1.7,">
  <!-- Rules for Linux systems -->
  <rule id="100200" level="7">
    <if_sid>550</if_sid>
    <field name="file">/home/target</field>
    <description>File modified in /home/target directory.</description>
  </rule>
  <rule id="100201" level="7">
    <if_sid>554</if_sid>
    <field name="file">/home/target</field>
    <description>File added to /home/target directory.</description>
  </rule>
</group>

```

And add this to */var/ossec/etc/ossec.conf*

```

<command>
  <name>firewall-drop</name>
  <executable>firewall-drop</executable>
  <timeout_allowed>yes</timeout_allowed>
</command>

<active-response>
  <disabled>no</disabled>
  <command>firewall-drop</command>
  <location>local</location>
  <rules_id>5763</rules_id>
  <timeout>180</timeout>
</active-response>

```

This uses a built-in function to IP Ban someone that has reached its max ssh attempts (8 default).

3.2.8. Infection Monkey

There isn't really much configuration here. I will put screenshots and you can find out where everything is (yes, it's that easy).

So go to the dashboard and follow along.

Only step ill give an explanation, you have to download the SSH Brute Force (not in my list since I already downloaded it before taking a screenshot rookie mistake I know).



Infection Monkey

- 1. Run Monkey ✓
- 2. Infection Map ✓
- 3. Security Reports ✓
- Reset

Configuration

Events

Plugins

Logout

Powered by 
Documentation
License
Infection Monkey Version: 2.3.0

Plugins

AVAILABLE PLUGINSINSTALLED PLUGINSUPLOAD NEW PLUGIN

SearchTypeAllALL SAFE PLUGINS

Name	Version	Type	Description	Safety
Chrome	1.0.0	Credentials Collector	Collects credentials from Chrome-based browsers.	
Cryptojacker	1.0.0	Payload	Simulates a cryptojacker running on your network using a set of configurable behaviors. To simulat...	
Hadoop	3.0.2	Exploiter	Remote code execution on Hadoop server with YARN and default settings. Logic based on [POC in...	
Log4Shell	2.0.1	Exploiter	Exploits a software vulnerability (CVE-2021-44228) in Apache Log4j, a Java logging framework. Ex...	
Mimikatz	1.0.2	Credentials Collector	Collects credentials from Windows Credential Manager using Mimikatz.	
MSSQL	2.0.1	Exploiter	Attempts a brute-force attack against Microsoft SQL using known credentials and takes advantage ...	
PowerShell	1.0.2	Exploiter	Exploits PowerShell remote execution setups. PowerShell Remoting uses Windows Remote Mana...	
Ransomware	1.0.0	Payload	Simulates a ransomware attack on your network using a set of configurable behaviors. To simulate...	
RDP	1.0.0	Exploiter	Attempts a brute-force attack over RDP using known credentials.	
SMB	1.1.1	Exploiter	Attempts a brute-force attack against SMB using known credentials.	
SNMP	2.0.0	Exploiter	Attempts remote command execution over SNMP using known credentials.	
SSH	1.0.1	Credentials Collector	Collects SSH keys from Linux users.	
WMI	1.1.0	Exploiter	Attempts a brute-force attack against WMI using known credentials.	
ZeroLogon	1.0.2	Exploiter	Exploits a privilege escalation vulnerability (CVE-2020-1472) in a Windows server domain controle...	

Rows per page: 1001-14 of 14



Infection Monkey

- 1. Run Monkey ✓
- 2. Infection Map
- 3. Security Reports
- Reset

Configuration

Events

Plugins

Logout

Powered by 
Documentation
License
Infection Monkey Version: 2.3.0

PropagationPayloadsCredentials collectorsMasqueradePolymorphismAdvanced

ExploitersNetwork analysisCredentialsGeneral

Exploiters

Configure the exploitation step of the attack

☒ Enabled exploiters

☒ SSH Exploiter

Enabled exploiters

Click on an exploiter to get more information about it.
⚠ Note that using unsafe exploits may cause crashes of the exploited machine/service.

Exploiters options

Configure exploitation options shared by all exploiters

HTTP ports

List of ports the Agent will check for using an HTTP protocol

80	↑	↓	+
443	↑	↓	+
7001	↑	↓	+
8008	↑	↓	+
8080	↑	↓	+
8983	↑	↓	+
9600	↑	↓	+

SubmitReset to defaults

Import configExport config

Monkey Configuration

Propagation

Payloads

Credentials collectors

Masquerade

Polymorphism

Advanced

Exploiters

Network analysis

Credentials

General

Network analysis

Configure the network analysis that the Agents will perform

Network

If "Scan Agent's networks" is checked, the Monkey scans for machines on each of the network interfaces of the machine it is running on. Additionally, the Monkey scans machines according to "Scan target list" and skips machines in "Blocked IPs".

Scan Agent's networks

☐ Scan Agent's networks

⚠ If the Agent runs on a machine that has a publicly-facing network interface, this setting could cause scanning and exploitation of systems outside your organization.

Scan target list

List of targets the Monkey will try to scan. Targets can be IPs, subnets or hosts. Examples:
Target a specific IP: "192.168.0.1"
Target a subnet using a network range: "192.168.0.5-192.168.0.20"
Target a subnet using an IP mask: "192.168.0.5/24"
Target a specific host: "printer.example"

192.168.1.12

-

+

Blocked IPs

List of IPs that the monkey will not scan.

+

Network segmentation testing

Test for network segmentation by providing a list of network segments that should not be accessible to each other.

For example, if you configured the following three segments: "10.0.0.0/24", "11.0.0.0/32" and "12.2.3.0/24", a Monkey running on 10.0.0.5 will try to access machines in the following

☒ Fingerprinters

☒ HTTP Fingerprinter

☒ MSSQL Fingerprinter

☒ SMB Fingerprinter

☒ SSH Fingerprinter

Fingerprinters

Fingerprint modules collect info about external services that Infection Monkey scans.

Fingerprint modules collect info about external services that Infection Monkey scans.

Submit

Reset to defaults

Import config

Export config

Only ssh needs to be on but it really doesn't impact performance

Monkey Configuration

Propagation

Payloads

Credentials collectors

Masquerade

Polymorphism

Advanced

Exploiters

Network analysis

Credentials

General

Credentials input

Identity	Password	LM	NTLM	SSH Public key	SSH Private key

SAVE

Saved Credentials

Identity ↑	Password	LM	NTLM	SSH Public key	SSH Private key	
target						
target						
target						
target						
target						
target						

Rows per page: 100 1-12 of 12 < >

Only problem I have with Infection Monkey is no built-in dictionaries to use for the attack.

And that's it only thing left is to run it to see if the ssh works or well doesn't work after 8 tries. We can test if we have been blocked by just running a ping (on kali this is endless until you stop it so it's a good indicator of connection to the VM we are attacking).

Infection Monkey

1. Run Monkey ✓

2. Infection Map ✓

3. Security Reports ✓

Reset

Configuration

Events

Plugins

Logout

Powered by Akamai

[Documentation](#)
[License](#)
 Infection Monkey Version: 2.3.0

1. Run Monkey

Go ahead and run the Monkey! (Or fine-tune its behavior by *adjusting the configuration*)

From Island
 Start on Monkey Island server.

Manual
 Run on a machine via command.

3.3. Testing

Configuration is done and we have tested the SSH bruteforce in there already.

So, I will guide you through the malware "attack", go to the target VM, we will type 1 command
wget <https://secure.eicar.org/eicar.com.txt>
if we do this in our home directory it should get flagged as malware by VirusTotal and be removed with the Active Response in Wazuh, the file we download is a test file which should get flagged by most malware detection. No actual malware.

4. Summary/Reflection

Building this SOC lab was a massive reality check. I realized that a professional defense isn't just about watching logs it's about **automation**.

4.1. What I Learned

- **API Integration is Key:** The hardest part wasn't the code; it was getting tools like Wazuh, Shuffle, and Iris to "talk" to each other. Solving the Iris API header issue taught me that security engineering is mostly about troubleshooting connectivity and data flow.
- **Fighting Alert Fatigue:** By using **VirusTotal** as a "brain," I learned how to filter noise. Sending every file to Discord is annoying; sending only *confirmed* threats is efficient.
- **Closing the Loop:** Seeing **Active Response** in action was the highlight. Watching my remove-threat.sh script delete malware before I could even click on it proved that automated defense is way faster than any human analyst.
- **Final Takeaway:** This project moved me from just "learning tools" to "building solutions." I now understand the full lifecycle of a threat and how to automate the boring stuff so I can focus on the real security issues.

5. Appendix

5.1. Appendix A: Shuffle to Discord JSON (Wazuh Alerts)

This JSON payload is used in the Shuffle Discord App to format the primary alert notification. It includes dynamic coloring based on the Wazuh rule level.

JSON

```
{
  "content": null,
  "embeds": [
    {
      "title": "🚨 WAZUH Security Alert",
      "description": "$exec.title",
      "color": {% if $exec.all_fields.rule.level <= 3 %}3447003{% elsif $exec.all_fields.rule.level <= 7 %}16776960{% elsif $exec.all_fields.rule.level <= 10 %}15158332{% else %}10038562{% endif %},
      "fields": [
        {
          "name": "📄 Rule Details",
          "value": "Rule ID - $exec.rule_id\nSeverity - $exec.all_fields.rule.level\nFired Times - $exec.all_fields.rule.firedtimes",
          "inline": true
        },
        {
          "name": "🎯 Target Information",
          "value": "Agent - $exec.all_fields.agent.name ($exec.all_fields.agent.id)\nAgent IP - $exec.all_fields.agent.ip\nLocation - $exec.all_fields.location",
          "inline": true
        },
        {
          "name": "🔍 Attack Details",
          "value": "Source IP - $exec.all_fields.data.srcip\nSource Port - $exec.all_fields.data.srcport\nTarget User - $exec.all_fields.data.dstuser",
          "inline": false
        },
        {
          "name": "📄 Full Log",
          "value": "```\n$exec.text\n```",
          "inline": false
        }
      ],
      "footer": {
        "text": "Wazuh Manager - $exec.all_fields.manager.name | Alert ID - $exec.id"
      },
      "timestamp": "$exec.timestamp"
    }
  ]
}
```

5.2. Appendix B: Shuffle to Discord JSON (VirusTotal Analysis)

This payload is triggered after the VirusTotal enrichment node to display the reputation of the source IP.
JSON

```
{
  "content": null,
  "embeds": [
    {
      "title": "🔍 VirusTotal IP Analysis",
      "description": "***IP Address** - $virustotal_v3_1.body.data.id",
      "color": {% if $virustotal_v3_1.body.data.attributes.last_analysis_stats.malicious > 5 %}10038562{% elsif
$virustotal_v3_1.body.data.attributes.last_analysis_stats.malicious > 0 %}15158332{% elsif
$virustotal_v3_1.body.data.attributes.last_analysis_stats.suspicious > 0 %}16776960{% else %}3066993{%
endif %},
      "fields": [
        {
          "name": "🚨 Detection Results",
          "value": "***Malicious** -
$virustotal_v3_1.body.data.attributes.last_analysis_stats.malicious\n**Suspicious** -
$virustotal_v3_1.body.data.attributes.last_analysis_stats.suspicious\n**Harmless** -
$virustotal_v3_1.body.data.attributes.last_analysis_stats.harmless\n**Undetected** -
$virustotal_v3_1.body.data.attributes.last_analysis_stats.undetected",
          "inline": true
        },
        {
          "name": "⭐ Community Votes",
          "value": "***Malicious** - $virustotal_v3_1.body.data.attributes.total_votes.malicious\n**Harmless** -
$virustotal_v3_1.body.data.attributes.total_votes.harmless",
          "inline": true
        },
        {
          "name": "📊 Reputation Score",
          "value": "$virustotal_v3_1.body.data.attributes.reputation",
          "inline": false
        },
        {
          "name": "🔗 VirusTotal Link",
          "value": "$virustotal_v3_1.body.data.links.self",
          "inline": false
        }
      ],
      "footer": {
        "text": "VirusTotal Analysis"
      },
      "timestamp": "$exec.timestamp"
    }
  ]
}
```

5.3. Appendix C: Shuffle to IRIS Alert JSON

This JSON configures how Shuffle pushes data into the IRIS case management system to create a new alert for investigation.

JSON

```
{
  "alert_title": "$exec.title",
  "alert_description": "$exec.all_fields.full_log",
  "alert_source": "Wazuh",
  "alert_source_ref": "$exec.id",
  "alert_source_link": "https://192.168.1.10",
  "alert_severity_id": 2,
  "alert_status_id": 2,
  "alert_source_event_time": "$exec.timestamp",
  "alert_note": "Rule ID: $exec.rule_id | Agent: $exec.all_fields.agent.name",
  "alert_tags": "wazuh, ssh_failure",
  "alert_customer_id": 1,
  "alert_classification_id": 1
}
```

5.4. Appendix D: Active Response Script (remove-threat.sh)

This script resides on the Ubuntu target agent. It is executed by Wazuh when a file is flagged as malicious by VirusTotal, effectively automating the containment phase.

Bash

```
#!/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;
```



CONTACT

Daan Van Winkel | Student
r0992869@thomasmore.be

THOMAS
MORE