

<b>Document Name</b>	<b>Brute-Force Protection (Ubuntu Web Server)</b>	<b>Version</b>	<b>1.3</b>
<b>Author</b>	<b>Anusha Ramu Chakravarthi</b>	<b>Date Created</b>	<b>24/04/2025</b>
<b>Protection Type</b>	<b>Brute-Force Defense (RDP and SMB Services)</b>	<b>Last Modified</b>	<b>27/04/2025</b>

## Document Description

This playbook outlines the implementation of brute-force protection for the Ubuntu 22.04 web server using **Fail2Ban**. It focuses on protecting SSH, Nginx, and WordPress (through PHP and web login attempts). This aligns with the **Protect** function of the NIST Cybersecurity Framework and ensures the security of publicly exposed services.

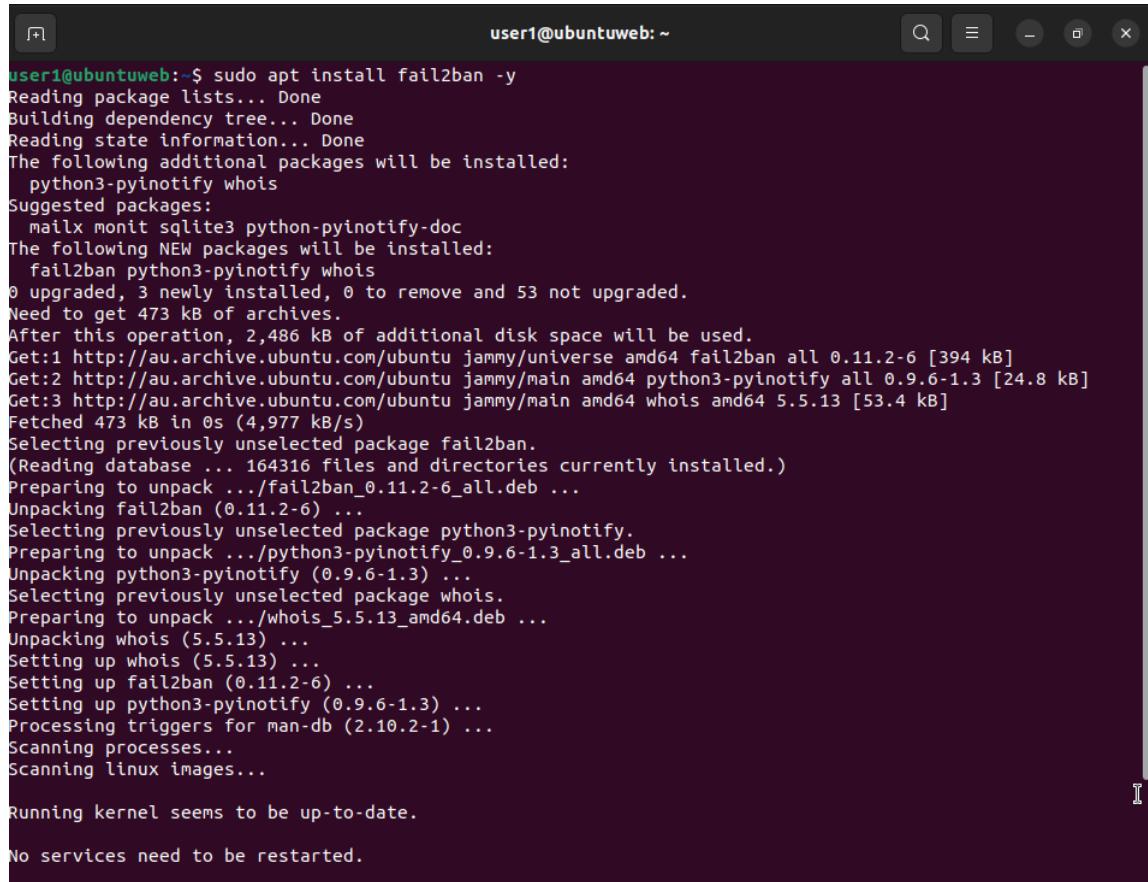
### Step 1

#### Install Fail2Ban on Ubuntu 22.04

Install the Fail2Ban package from the official repositories.

- Run:

```
sudo apt update && sudo apt install fail2ban -y
```



```
user1@ubuntuweb: $ sudo apt install fail2ban -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  python3-pyinotify whois
Suggested packages:
  mailx monit sqlite3 python-pyinotify-doc
The following NEW packages will be installed:
  fail2ban python3-pyinotify whois
0 upgraded, 3 newly installed, 0 to remove and 53 not upgraded.
Need to get 473 kB of archives.
After this operation, 2,486 kB of additional disk space will be used.
Get:1 http://au.archive.ubuntu.com/ubuntu jammy/universe amd64 fail2ban all 0.11.2-6 [394 kB]
Get:2 http://au.archive.ubuntu.com/ubuntu jammy/main amd64 python3-pyinotify all 0.9.6-1.3 [24.8 kB]
Get:3 http://au.archive.ubuntu.com/ubuntu jammy/main amd64 whois amd64 5.5.13 [53.4 kB]
Fetched 473 kB in 0s (4,977 kB/s)
Selecting previously unselected package fail2ban.
(Reading database ... 164316 files and directories currently installed.)
Preparing to unpack .../fail2ban_0.11.2-6_all.deb ...
Unpacking fail2ban (0.11.2-6) ...
Selecting previously unselected package python3-pyinotify.
Preparing to unpack .../python3-pyinotify_0.9.6-1.3_all.deb ...
Unpacking python3-pyinotify (0.9.6-1.3) ...
Selecting previously unselected package whois.
Preparing to unpack .../whois_5.5.13_amd64.deb ...
Unpacking whois (5.5.13) ...
Setting up whois (5.5.13) ...
Setting up fail2ban (0.11.2-6) ...
Setting up python3-pyinotify (0.9.6-1.3) ...
Processing triggers for man-db (2.10.2-1) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.
```

- Ensure the service is enabled and running:

```
sudo systemctl enable fail2ban && sudo systemctl start fail2ban
```



```
user1@ubuntuweb:~$ sudo systemctl enable fail2ban && sudo systemctl start fail2ban
Synchronizing state of fail2ban.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable fail2ban
Created symlink /etc/systemd/system/multi-user.target.wants/fail2ban.service → /lib/systemd/system/fail2ban.service.
user1@ubuntuweb:~$
```

## Step 2

### Protect SSH Service

Configure Fail2Ban to protect SSH against brute-force attacks.

- **Copy default config:**

```
sudo cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local
```

- **Edit the jail file:**

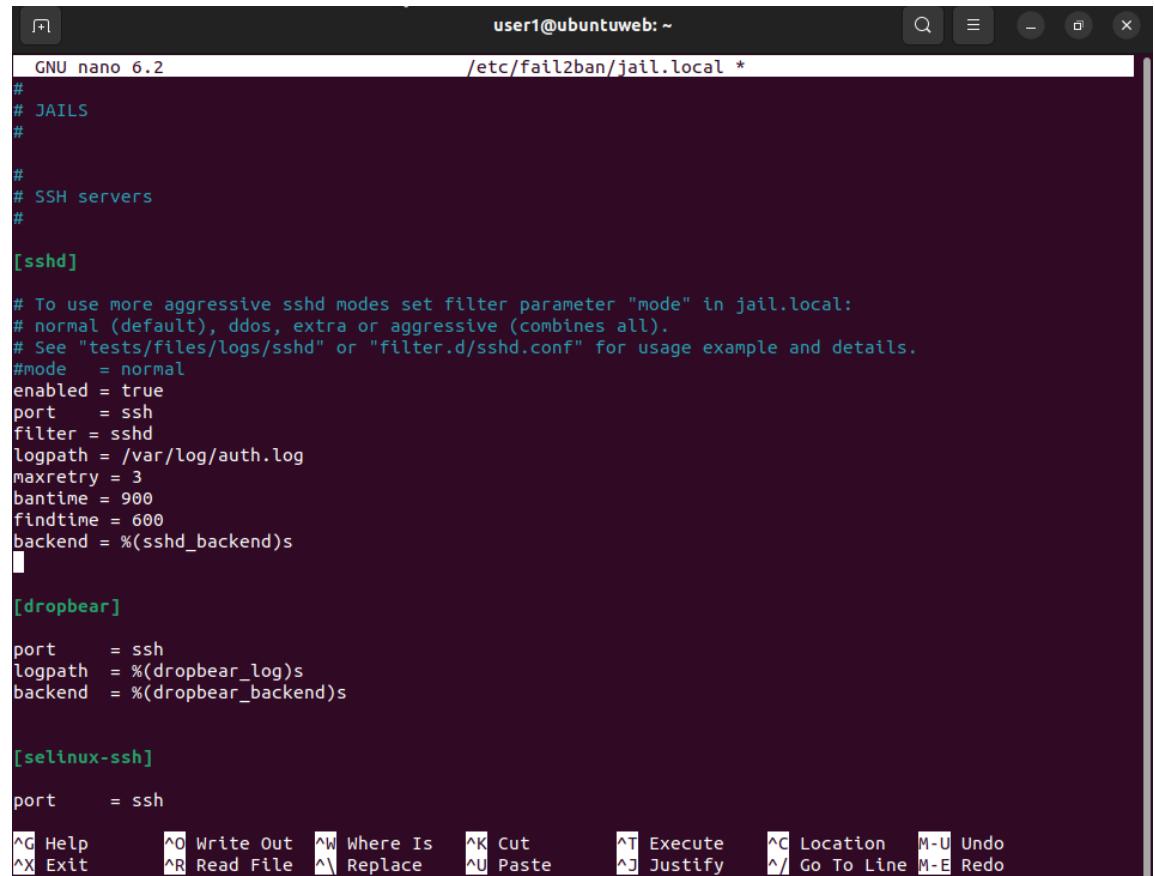
```
sudo nano /etc/fail2ban/jail.local
```



```
user1@ubuntuweb:~$ sudo cp /etc/f
fail2ban/   firefox/   fonts/   fprintd.conf  fstab      fuse.conf    fwupd/
user1@ubuntuweb:~$ sudo cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local
user1@ubuntuweb:~$ sudo nano /etc/fail2ban/jail.local
```

- Enable and configure the [sshd] jail:

```
[sshd]
enabled = true
port = ssh
filter = sshd
logpath = /var/log/auth.log
maxretry = 3
bantime = 900
findtime = 600
```



```
user1@ubuntuweb: ~
GNU nano 6.2          /etc/fail2ban/jail.local *

#
# JAILS
#
#
# SSH servers
#
[sshd]
# To use more aggressive sshd modes set filter parameter "mode" in jail.local:
# normal (default), ddos, extra or aggressive (combines all).
# See "tests/files/logs/sshd" or "filter.d/sshd.conf" for usage example and details.
#mode   = normal
enabled = true
port    = ssh
filter  = sshd
logpath = /var/log/auth.log
maxretry = 3
bantime = 900
findtime = 600
backend = %(sshd_backend)s

[dropbear]

port    = ssh
logpath = %(dropbear_log)s
backend = %(dropbear_backend)s

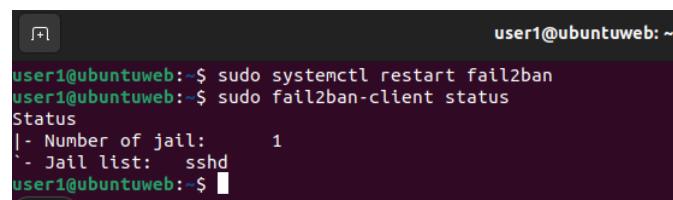
[selinux-ssh]

port    = ssh

^G Help      ^O Write Out  ^W Where Is  ^K Cut        ^T Execute  ^C Location  M-U Undo
^X Exit      ^R Read File  ^A Replace   ^U Paste     ^J Justify  ^L Go To Line M-E Redo
```

- Save and restart:

```
sudo systemctl restart fail2ban
```



```
user1@ubuntuweb:~$ sudo systemctl restart fail2ban
user1@ubuntuweb:~$ sudo fail2ban-client status
Status
|- Number of jail:      1
`- Jail list:  sshd
user1@ubuntuweb:~$
```

## Step 3

### Enable Nginx Brute-Force Protection

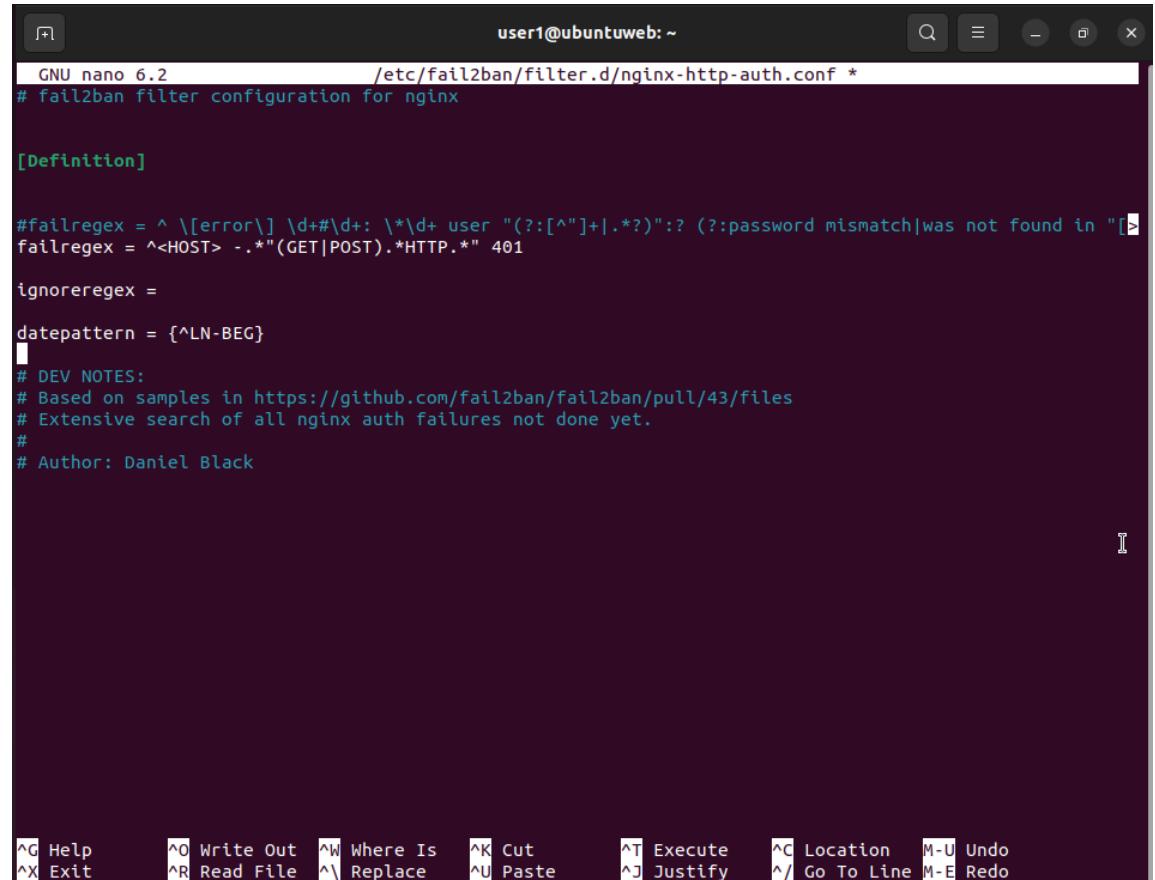
- Configure protection for Nginx HTTP/HTTPS services.
- Create **custom filter**: /etc/fail2ban/filter.d/nginx-http-auth.conf
- Add rules for detecting failed login attempts via Nginx logs.

```
sudo nano /etc/fail2ban/filter.d/nginx-http-auth.conf
```

**Write:**

```
failregex = ^<HOST> -.*"(GET|POST).*HTTP.*" 401
```

```
ignoreregex =
```



```
GNU nano 6.2 /etc/fail2ban/filter.d/nginx-http-auth.conf *
# fail2ban filter configuration for nginx

[Definition]

#failregex = ^ \[error\] \d+ user "(?:[^"]+|.*?)"(?: password mismatch|was not found in "[")
failregex = ^<HOST> -.*"(GET|POST).*HTTP.*" 401

ignoreregex =

datepattern = {^LN-BEG}

# DEV NOTES:
# Based on samples in https://github.com/fail2ban/fail2ban/pull/43/files
# Extensive search of all nginx auth failures not done yet.
#
# Author: Daniel Black


```

The terminal window has a dark theme. The nano editor's command bar at the bottom includes the following keys: ^G Help, ^O Write Out, ^W Where Is, ^K Cut, ^T Execute, ^C Location, M-U Undo, ^X Exit, ^R Read File, ^\ Replace, ^U Paste, ^J Justify, ^/ Go To Line, M-E Redo.

- In **jail.local**, add:

```
[nginx-http-auth]
enabled = true
port = http,https
filter = nginx-http-auth
logpath = /var/log/nginx/error.log
maxretry = 3
bantime = 900
findtime = 600
```

The screenshot shows a terminal window titled "user1@ubuntuweb: ~" with the command "/etc/fail2ban/jail.local \*" in the title bar. The window contains the fail2ban configuration file "jail.local". The file includes sections for [nginx-http-auth], [openhab-auth], [nginx-limit-req], and [nginx-botsearch]. It defines various parameters like enabled, port, filter, logpath, maxretry, bantime, and findtime. A note at the bottom of the configuration file suggests using the 'nginx-limit-req' jail with the 'ngx\_http\_limit\_req\_module'. The terminal window has a dark theme and includes a menu bar with options like Help, Exit, Write Out, Read File, Where Is, Replace, Cut, Paste, Execute, Justify, Location, Go To Line, Undo, and Redo.

```
GNU nano 6.2                               /etc/fail2ban/jail.local *
maxretry = 1

[openhab-auth]

filter = openhab
banaction = %(banaction_allports)s
logpath = /opt/openhab/logs/request.log

[nginx-http-auth]

enabled = true
port = http,https
filter = nginx-http-auth
logpath = /var/log/nginx/error.log
maxretry = 3
bantime = 900
findtime = 600
#
# To use 'nginx-limit-req' jail you should have 'ngx_http_limit_req_module'
# and define 'limit_req' and 'limit_req_zone' as described in nginx documentation
# http://nginx.org/en/docs/http/ngx_http_limit_req_module.html
# or for example see in 'config/filter.d/nginx-limit-req.conf'
[nginx-limit-req]
port = http,https
logpath = %(nginx_error_log)s

[nginx-botsearch]

port = http,https
logpath = %(nginx_error_log)s
maxretry = 2

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

- Restart Fail2Ban.

```
sudo systemctl restart fail2ban
```

## Step 4

### Protect WordPress via wp-login.php & XMLRPC

Use custom Fail2Ban filters for WordPress login abuse detection.

- Install and configure the fail2ban-wordpress filter (or create custom regex).

```
sudo nano /etc/fail2ban/filter.d/wordpress.conf
```

**Write:**

```
failregex = <HOST> -.*"(GET|POST) /wp-login.php HTTP.*" 200  
<HOST> -.*"(GET|POST) /xmlrpc.php HTTP.*" 200
```

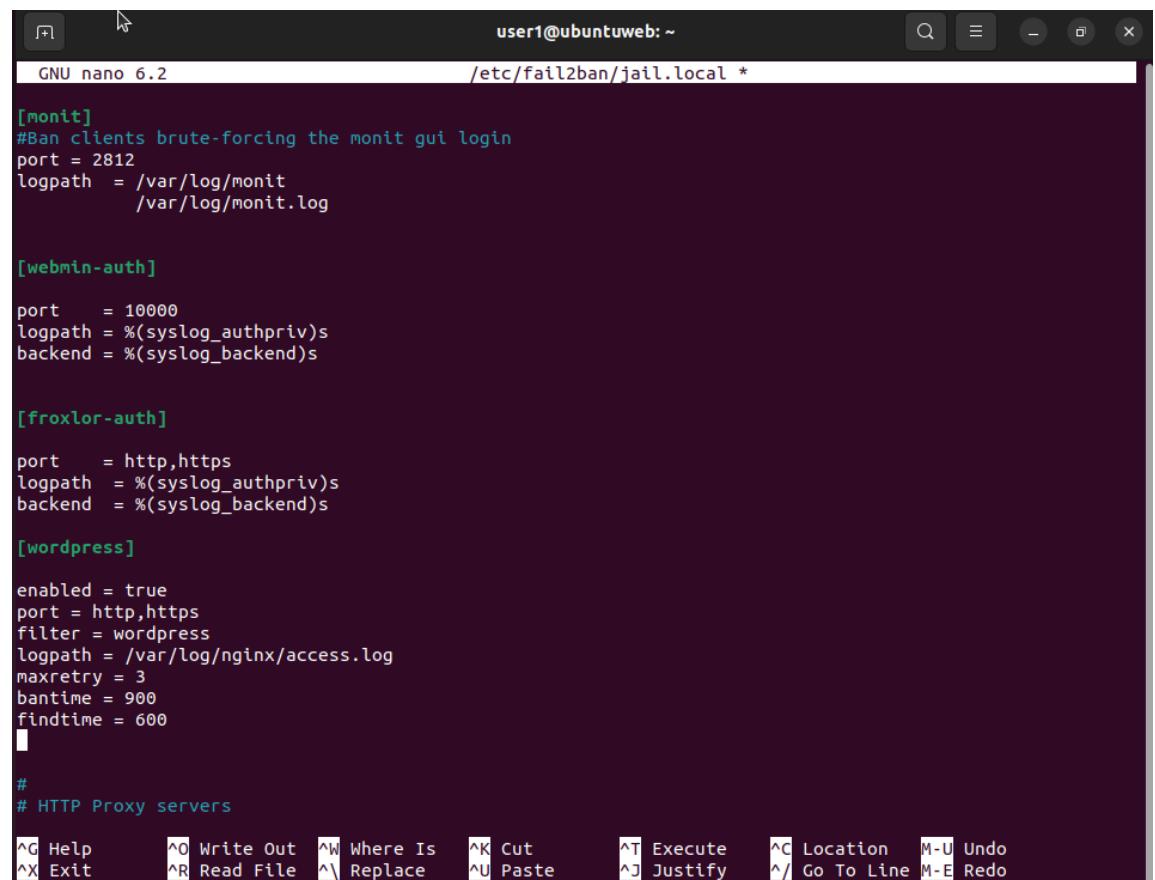
```
ignoreregex =
```

```
user1@ubuntuweb: ~  
GNU nano 6.2 /etc/fail2ban/filter.d/wordpress.conf  
[Definition]  
failregex = <HOST> -.*"(GET|POST) /wp-login.php HTTP.*" 200  
          <HOST> -.*"(GET|POST) /xmlrpc.php HTTP.*" 200  
  
ignoreregex =  
  
^G Help      ^O Write Out   ^W Where Is   [ Read 6 lines ]  
^X Exit      ^R Read File   ^\ Replace    ^K Cut        ^T Execute  
          ^U Paste     ^J Justify    ^C Location   M-U Undo  
          ^/ Go To Line M-E Redo
```

- Monitor logs: `/var/log/nginx/access.log` or `/var/log/auth.log`

- In **jail.local**, add:

```
[wordpress]
enabled = true
port = http,https
filter = wordpress
logpath = /var/log/nginx/access.log
maxretry = 3
bantime = 900
findtime = 600
```



```
user1@ubuntuweb: ~
/etc/fail2ban/jail.local *

[monit]
#Ban clients brute-forcing the monit gui login
port = 2812
logpath = /var/log/monit
          /var/log/monit.log

[webmin-auth]
port = 10000
logpath = %(syslog_authpriv)s
backend = %(syslog_backend)s

[froxlor-auth]
port = http,https
logpath = %(syslog_authpriv)s
backend = %(syslog_backend)s

[wordpress]
enabled = true
port = http,https
filter = wordpress
logpath = /var/log/nginx/access.log
maxretry = 3
bantime = 900
findtime = 600
#
# HTTP Proxy servers

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

- Save and restart Fail2Ban.

```
sudo systemctl restart fail2ban
```

## Step 5

### Enable UFW and Allow Only Essential Ports

1. Allow **SSH** (default port 22):

```
sudo ufw allow OpenSSH
```

2. Allow **HTTP** and **HTTPS**:

```
sudo ufw allow 80
```

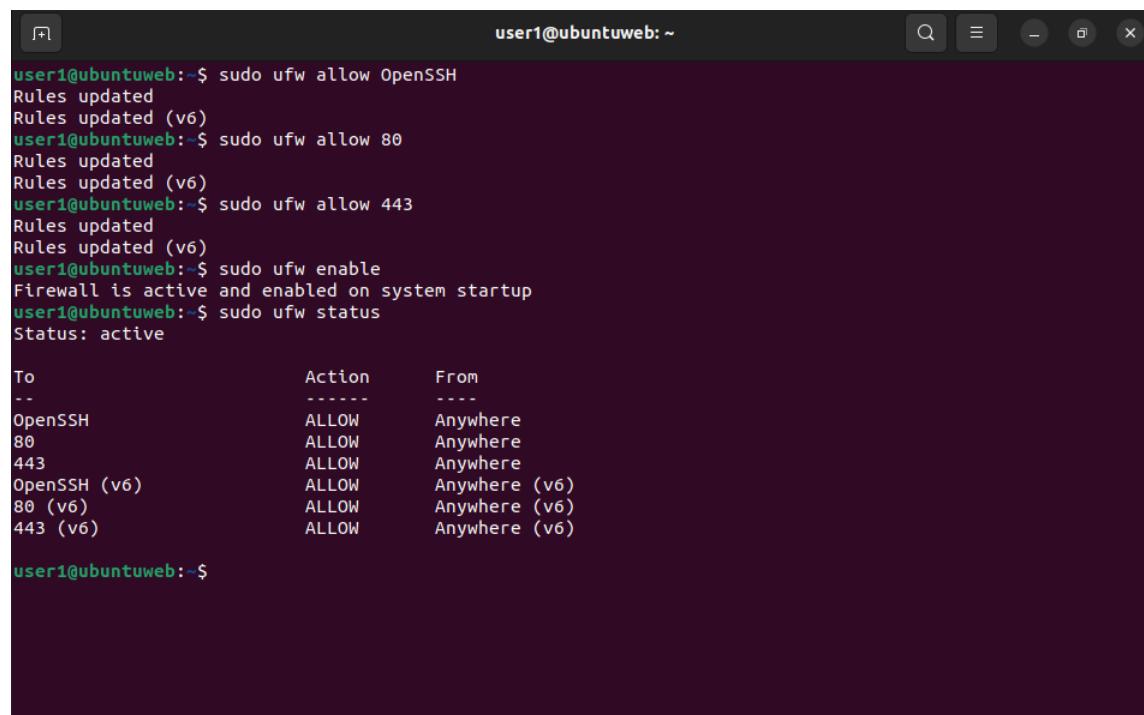
```
sudo ufw allow 443
```

3. Enable UFW:

```
sudo ufw enable
```

4. Check status:

```
sudo ufw status
```



The screenshot shows a terminal window titled "user1@ubuntuweb: ~". The user has run several commands to manage the firewall:

- `sudo ufw allow OpenSSH`: Rules updated (v6)
- `sudo ufw allow 80`: Rules updated (v6)
- `sudo ufw allow 443`: Rules updated (v6)
- `sudo ufw enable`: Firewall is active and enabled on system startup
- `sudo ufw status`: Status: active

The final output shows the current list of allowed ports and their actions:

To	Action	From
--	-----	-----
OpenSSH	ALLOW	Anywhere
80	ALLOW	Anywhere
443	ALLOW	Anywhere
OpenSSH (v6)	ALLOW	Anywhere (v6)
80 (v6)	ALLOW	Anywhere (v6)
443 (v6)	ALLOW	Anywhere (v6)

## Step 6

### Block Known Malicious User Agents in Nginx

1. Open your Nginx site config:

```
sudo nano /etc/nginx/sites-available/your-site.conf
```

2. Inside the server {} block, add:

```
if ($http_user_agent ~* (nikto|sqlmap|nmap|masscan)) {  
    return 403;  
}
```

3. Save and exit CTRL+O, ENTER, CTRL+X

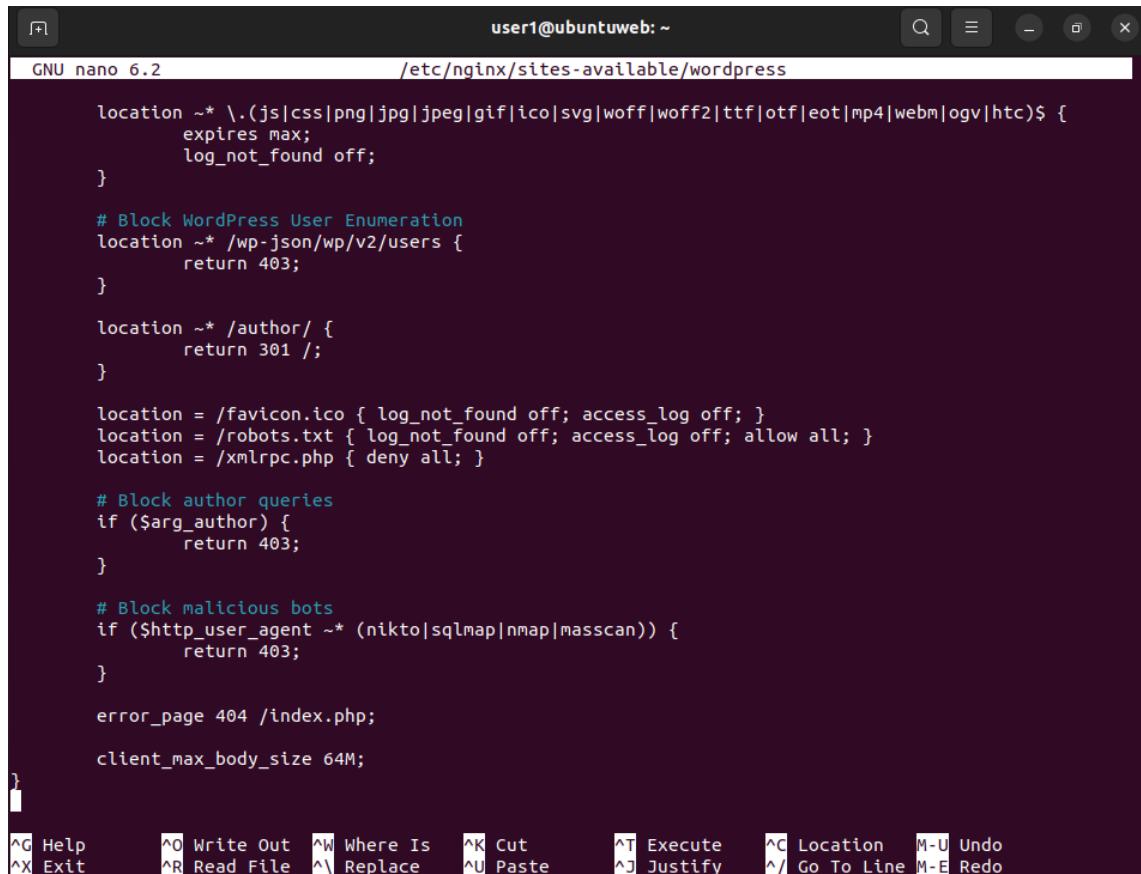
4. Test config:

```
sudo nginx -t
```

5. Reload Nginx:

```
sudo systemctl reload nginx
```

This blocks common vulnerability scanners from reaching your site.



The screenshot shows a terminal window titled "user1@ubuntuweb: ~". The command "GNU nano 6.2" is running in the background, editing the file "/etc/nginx/sites-available/wordpress". The terminal content displays an Nginx configuration file with several location blocks. One specific block is highlighted with a yellow background, containing the following code:

```
if ($http_user_agent ~* (nikto|sqlmap|nmap|masscan)) {  
    return 403;  
}
```

The configuration also includes blocks to block WordPress user enumeration, malicious bots, and specific file types like favicons and robots.txt. The terminal interface includes standard nano key bindings at the bottom.

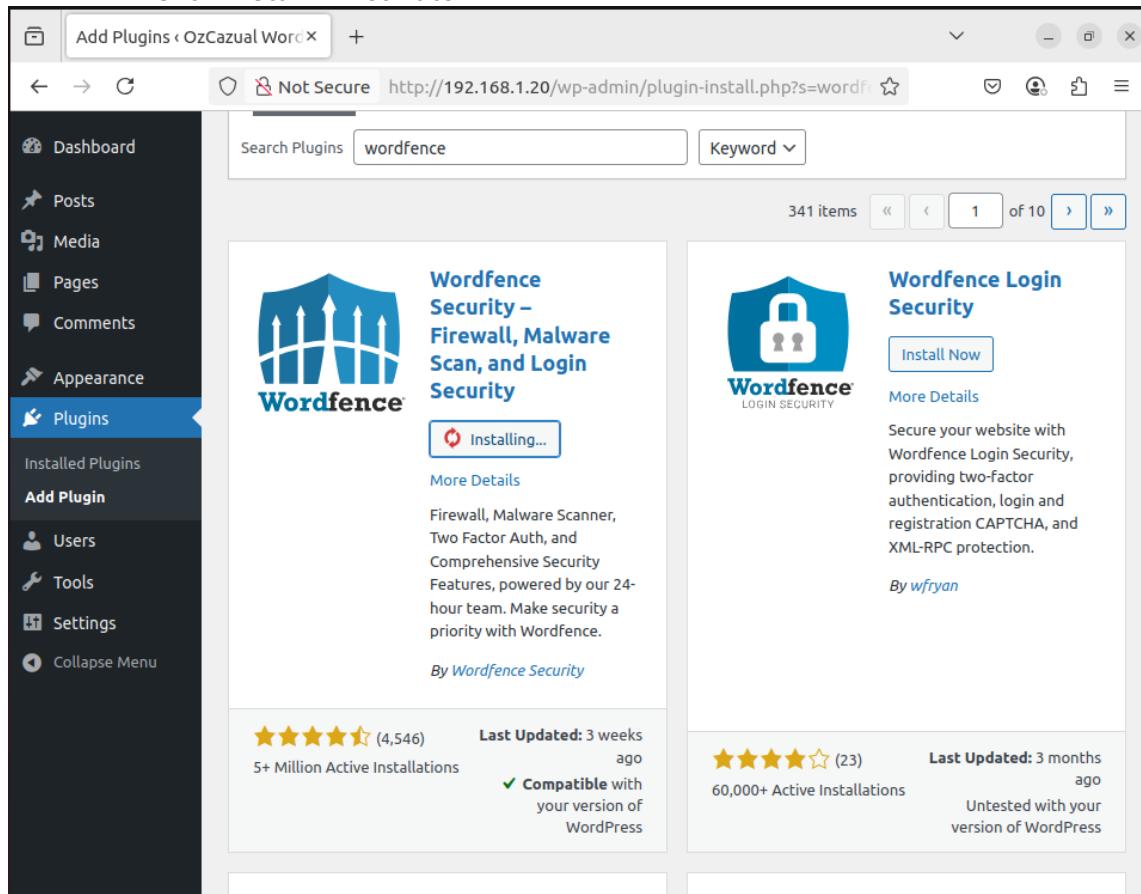
## Step 7

### Setup WordPress Plugins (like Wordfence)

Install and configure Wordfence to enable MFA, firewall and for monitoring logs.

- **Install and Configure Wordfence Login Security**

1. **Log in to WordPress Admin** (<https://your-ip/wp-admin>)
2. Go to:  
**Plugins > Add New**
3. Search for:  
**Wordfence Security – Firewall & Malware Scan**
4. Click **Install > Activate**



- **Configure 2FA (Wordfence)**
  1. Go to: **Wordfence > Login Security**
  2. Scan the QR code using **Google Authenticator**, **Authy**, or similar.
  3. Save your **recovery codes**.

**Two-Factor Authentication**

Editing User: user66 (you)

1. Scan Code or Enter Key

Scan the code below with your authenticator app to add this account. Some authenticator apps also allow you to type in the text version instead.

LJGOSFUDCOZUYZLTGY05KC3IWMRIPTWC

2. Enter Code from Authenticator App

Download Recovery Codes *Optional*

Use one of these 5 codes to log in if you lose access to your authenticator device. Codes are 16 characters long plus optional spaces. Each one may be used only once.

efd6 12ce 1548 51d9  
f459 7696 ff8f 4b09  
6f10 6d76 31ef dd56  
51fc 928c b01a 6090  
01ef 7771 48fa d6e5

**DOWNLOAD**

Enter the code from your authenticator app below to verify and activate two-factor authentication for this account.

4. Enable for:
  - **Your admin account**
  - Optionally: all other users (checkbox under “Settings” tab)

Role	Total Users	2FA Active	2FA Inactive
Administrator	2	1	1
Author	1	0	1
Subscriber	2	0	2
<b>Total</b>	<b>5</b>	<b>1</b>	<b>4</b>

**2FA Roles**

Administrator	Editor	Author	Contributor	Subscriber
Required	Required	Required	Required	Required

**Grace Period**

- **Configure Firewall and Lockout Settings**
  1. Go to: **Wordfence > Firewall**
  2. Click **Manage Firewall**
    - Set it to “Enabled and Protecting”

The screenshot shows the 'Basic Firewall Options' section of the Wordfence Firewall interface. On the left is a dark sidebar with navigation links for Dashboard, Posts, Media, Pages, Comments, Appearance, Plugins, Users, Tools, Settings, and Wordfence (which is currently selected). The main content area has a header with 'Firewall Options < Firewall' and a 'Not Secure' warning. It includes a 'Back to Firewall' link and buttons for 'RESTORE DEFAULTS', 'CANCEL', and 'SAVE'. Below this is a 'Basic Firewall Options' section with two columns: 'Web Application Firewall Status' (set to 'Enabled and Protecting') and 'Protection Level' (described as 'Extended Protection: All PHP requests will be processed by the firewall prior to running'). A 'REMOVE EXTENDED PROTECTION' button is also present. At the bottom is a 'Real-Time IP Blocklist' section with a 'Premium Feature' note, an 'UPGRADE TO PREMIUM' button, and a 'LEARN MORE' button.

3. Go to **Rate Limiting tab:**
  - Brute-force protection settings:
    - Lock out after 3–5 login failures
    - Lock out for 5–15 minutes
    - Choose to email on lockout

The screenshot shows the 'Brute Force Protection' section of the Wordfence Firewall interface. The sidebar and header are identical to the previous screenshot. The main content area has a 'Brute Force Protection' section with an 'Enable brute force protection' toggle switch (set to 'ON'). Below this are four configuration fields: 'Lock out after how many login failures' (set to 3), 'Lock out after how many forgot password attempts' (set to 3), 'Count failures over what time period' (set to 10 minutes), and 'Amount of time a user is locked out' (set to 30 minutes). There are also two checkboxes at the bottom: 'Immediately lock out invalid usernames' (unchecked) and 'Immediately block the IP of users who try to sign in as these usernames' (unchecked).

4. Go to: **Wordfence > Tools > Live Traffic**
  - o Monitor brute-force attempts (IP, username, action)
- **Avoid Lockout During Hydra or WPScan**
  - Go to: **Wordfence > All Options > Brute Force Protection**
  - Add **your IP address** to “Allowlisted IP addresses that bypass all rules”

Temporarily increase lockout thresholds during testing

**Advanced Firewall Options**

Allowlisted IP addresses that bypass all rules [?](#)

127.0.0.1/8  
192.168.1.20

Allowlisted services [?](#)

Sucuri  Facebook  Uptime Robot  StatusCake  ManageWP  Seznam Search Engine

Immediately block IPs that access these URLs [?](#)

Separate multiple URLs with commas or place them on separate lines. Asterisks are wildcards, but use with care. If you see an attacker repeatedly probing your site for a known vulnerability you can use this to immediately block them. All URLs must start with a "/" without quotes and must be relative. e.g. /badURLone/, /bannedPage.html, /dont-access/this/URL/, /starts/with-\*

## Step 8

### Enable Logging and Monitoring

Ensure logs are retained for analysis and alerts are configured.

- View Fail2Ban status:

```
sudo fail2ban-client status and sudo fail2ban-client status [jailname]
```

- Enable **email alerts** (optional) in **/etc/fail2ban/jail.local**
- Integrate with **log aggregation tools** if available (e.g., Graylog, ELK)

## Step 9

### Whitelist Internal IPs

Prevent banning of trusted internal sources.

- Edit ignoreip in /etc/fail2ban/jail.local:

```
ignoreip = 127.0.0.1/8 192.168.1.20/24
```

```
GNU nano 6.2 /etc/fail2ban/jail.local *
# more aggressive example of formula has the same values only for factor "2.0 / 2.885385" :
#bantime.formula = ban.Time * math.exp(float(ban.Count+1)*banFactor)/math.exp(1*banFactor)

# "bantime.multipliers" used to calculate next value of ban time instead of formula, corresponding
# previously ban count and given "bantime.factor" (for multipliers default is 1);
# following example grows ban time by 1, 2, 4, 8, 16 ... and if last ban count greater as multipliers c>
# always used last multiplier (64 in example), for factor '1' and original ban time 600 - 10.6 hours
#bantime.multipliers = 1 2 4 8 16 32 64
# following example can be used for small initial ban time (bantime=60) - it grows more aggressive at b>
# for bantime=60 the multipliers are minutes and equal: 1 min, 5 min, 30 min, 1 hour, 5 hour, 12 hour, >
#bantime.multipliers = 1 5 30 60 300 720 1440 2880

# "bantime.overalljails" (if true) specifies the search of IP in the database will be executed
# cross over all jails, if false (default), only current jail of the ban IP will be searched
#bantime.overalljails = false

# -----
#
# "ignoreself" specifies whether the local resp. own IP addresses should be ignored
# (default is true). Fail2ban will not ban a host which matches such addresses.
#ignoreself = true

# "ignoreip" can be a list of IP addresses, CIDR masks or DNS hosts. Fail2ban
# will not ban a host which matches an address in this list. Several addresses
# can be defined using space (and/or comma) separator.
ignoreip = 127.0.0.1/8 192.168.1.20 192.168.1.99

# External command that will take an tagged arguments to ignore, e.g. <ip>,
# and return true if the IP is to be ignored. False otherwise.
#
# ignorecommand = /path/to/command <ip>
ignorecommand =

# "bantime" is the number of seconds that a host is banned.

^G Help      ^O Write Out   ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo
^X Exit      ^R Read File   ^A Replace    ^U Paste      ^J Justify    ^I Go To Line M-E Redo
```

- Save and restart Fail2Ban
- Verify the setting is active:

```
sudo fail2ban-client get sshd ignoreip
```

```
user1@ubuntuweb:~$ sudo systemctl restart fail2ban
user1@ubuntuweb:~$ sudo fail2ban-client get sshd ignoreip
These IP addresses/networks are ignored:
|- 192.168.1.99
|- 192.168.1.20
`- 127.0.0.1
user1@ubuntuweb:~$
```

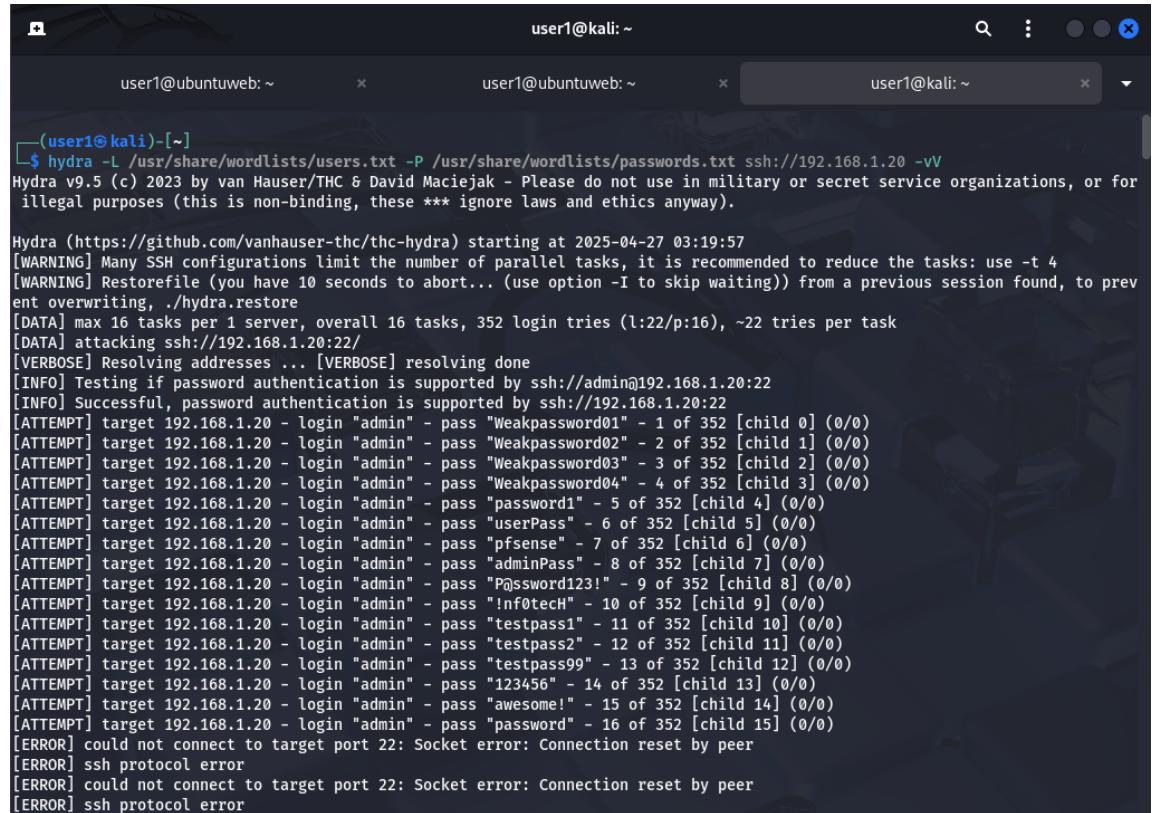
## Step 10

### Test Brute-Force Scenarios and Validate Protection

Simulate attacks using tools to confirm Fail2Ban is working.

- Use **Hydra** to simulate SSH and WordPress brute-force attacks

```
hydra -L users.txt -P passwords.txt ssh://<target_IP> -vv
```



The screenshot shows a terminal window with three tabs, all titled "user1@ubuntuweb: ~". The central tab displays the command being run:

```
$ hydra -L /usr/share/wordlists/users.txt -P /usr/share/wordlists/passwords.txt ssh://192.168.1.20 -vv
```

Hydra v9.5 (c) 2023 by van Hauser/TiHC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these \*\*\* ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-04-27 03:19:57

[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4

[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a previous session found, to prevent overwriting, ./hydra.restore

[DATA] max 16 tasks per 1 server, overall 16 tasks, 352 login tries (l:22/p:16), ~22 tries per task

[DATA] attacking ssh://192.168.1.20:22

[VERBOSE] Resolving addresses ... [VERBOSE] resolving done

[INFO] Testing if password authentication is supported by ssh://admin@192.168.1.20:22

[INFO] Successful, password authentication is supported by ssh://192.168.1.20:22

[ATTEMPT] target 192.168.1.20 - login "admin" - pass "Weakpassword01" - 1 of 352 [child 0] (0/0)

[ATTEMPT] target 192.168.1.20 - login "admin" - pass "Weakpassword02" - 2 of 352 [child 1] (0/0)

[ATTEMPT] target 192.168.1.20 - login "admin" - pass "Weakpassword03" - 3 of 352 [child 2] (0/0)

[ATTEMPT] target 192.168.1.20 - login "admin" - pass "Weakpassword04" - 4 of 352 [child 3] (0/0)

[ATTEMPT] target 192.168.1.20 - login "admin" - pass "password1" - 5 of 352 [child 4] (0/0)

[ATTEMPT] target 192.168.1.20 - login "admin" - pass "userPass" - 6 of 352 [child 5] (0/0)

[ATTEMPT] target 192.168.1.20 - login "admin" - pass "pfsense" - 7 of 352 [child 6] (0/0)

[ATTEMPT] target 192.168.1.20 - login "admin" - pass "adminPass" - 8 of 352 [child 7] (0/0)

[ATTEMPT] target 192.168.1.20 - login "admin" - pass "P@ssword123!" - 9 of 352 [child 8] (0/0)

[ATTEMPT] target 192.168.1.20 - login "admin" - pass "!nf0tecH" - 10 of 352 [child 9] (0/0)

[ATTEMPT] target 192.168.1.20 - login "admin" - pass "testpass1" - 11 of 352 [child 10] (0/0)

[ATTEMPT] target 192.168.1.20 - login "admin" - pass "testpass2" - 12 of 352 [child 11] (0/0)

[ATTEMPT] target 192.168.1.20 - login "admin" - pass "testpass99" - 13 of 352 [child 12] (0/0)

[ATTEMPT] target 192.168.1.20 - login "admin" - pass "123456" - 14 of 352 [child 13] (0/0)

[ATTEMPT] target 192.168.1.20 - login "admin" - pass "awesome!" - 15 of 352 [child 14] (0/0)

[ATTEMPT] target 192.168.1.20 - login "admin" - pass "password" - 16 of 352 [child 15] (0/0)

[ERROR] could not connect to target port 22: Socket error: Connection reset by peer

[ERROR] ssh protocol error

[ERROR] could not connect to target port 22: Socket error: Connection reset by peer

[ERROR] ssh protocol error

```
hydra -L users.txt -P passwords.txt <ubuntu_ip> http-post-form "/wp-login.php:log^USER^&pwd^PASS^&wp-submit=Log+In&redirect_to=/wp-admin&testcookie=1:S=wp-admin" -V
```

```
user1@kali:~  
_____(user1㉿kali)-[~]  
$ hydra -L /usr/share/wordlists/users.txt -P /usr/share/wordlists/passwords.txt 192.168.1.20 http-post-form "/wp-login.php:log=^USER^&pwd=^PASS^&wp-submit=Log+In&redirect_to=/wp-admin&testcookie=1:F-The username or password"-V  
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).  
  
hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-05-01 06:14:20  
[DATA] max 16 tasks per 1 server, overall 16 tasks, 352 login tries (l:22/p:16), ~22 tries per task  
[DATA] attacking http-post-form://192.168.1.20:80/wp-login.php:log=^USER^&pwd=^PASS^&wp-submit=Log+In&redirect_to=/wp-admin&testcookie=1:F-The username or password  
[ATTEMPT] target 192.168.1.20 - login "admin" - pass "Weakpassword01" - 1 of 352 [child 0] (0/0)  
[ATTEMPT] target 192.168.1.20 - login "admin" - pass "Weakpassword02" - 2 of 352 [child 1] (0/0)  
[ATTEMPT] target 192.168.1.20 - login "admin" - pass "Weakpassword03" - 3 of 352 [child 2] (0/0)  
[ATTEMPT] target 192.168.1.20 - login "admin" - pass "Weakpassword04" - 4 of 352 [child 3] (0/0)  
[ATTEMPT] target 192.168.1.20 - login "admin" - pass "password1" - 5 of 352 [child 4] (0/0)  
[ATTEMPT] target 192.168.1.20 - login "admin" - pass "userPass" - 6 of 352 [child 5] (0/0)  
[ATTEMPT] target 192.168.1.20 - login "admin" - pass "pfsense" - 7 of 352 [child 6] (0/0)  
[ATTEMPT] target 192.168.1.20 - login "admin" - pass "adminPass" - 8 of 352 [child 7] (0/0)  
[ATTEMPT] target 192.168.1.20 - login "admin" - pass "P@ssword123!" - 9 of 352 [child 8] (0/0)  
[ATTEMPT] target 192.168.1.20 - login "admin" - pass "!nf0t3ch" - 10 of 352 [child 9] (0/0)  
[ATTEMPT] target 192.168.1.20 - login "admin" - pass "testpass1" - 11 of 352 [child 10] (0/0)  
[ATTEMPT] target 192.168.1.20 - login "admin" - pass "testpass2" - 12 of 352 [child 11] (0/0)  
[ATTEMPT] target 192.168.1.20 - login "admin" - pass "testpass99" - 13 of 352 [child 12] (0/0)  
[ATTEMPT] target 192.168.1.20 - login "admin" - pass "123456" - 14 of 352 [child 13] (0/0)  
[ATTEMPT] target 192.168.1.20 - login "admin" - pass "awesome!" - 15 of 352 [child 14] (0/0)  
[ATTEMPT] target 192.168.1.20 - login "admin" - pass "password" - 16 of 352 [child 15] (0/0)  
[80][http-post-form] host: 192.168.1.20 login: admin password: Weakpassword03  
[80][http-post-form] host: 192.168.1.20 login: admin password: Weakpassword02  
[ATTEMPT] target 192.168.1.20 - login "ozcazual\administrator" - pass "Weakpassword01" - 17 of 352 [child 2] (0/0)  
[80][http-post-form] host: 192.168.1.20 login: admin password: Weakpassword04  
[80][http-post-form] host: 192.168.1.20 login: admin password: password1  
[80][http-post-form] host: 192.168.1.20 login: admin password: userPass  
[80][http-post-form] host: 192.168.1.20 login: admin password: pfsense  
[80][http-post-form] host: 192.168.1.20 login: admin password: adminPass  
[80][http-post-form] host: 192.168.1.20 login: admin password: P@ssword123!  
[80][http-post-form] host: 192.168.1.20 login: admin password: !nf0t3ch  
[80][http-post-form] host: 192.168.1.20 login: admin password: testpass1
```

In the above image, the hydra gets all the user-password combinations as valid for WordPress login, as it cannot detect the valid user credentials hence confusing the attacker.

- Use **WPScan** to simulate WordPress brute-force attacks

```
wpscan -v --url <ip> -U users.txt -P passwords.txt --force
```

- Monitor ban behavior:

```
fail2ban-client status
fail2ban-client status sshd
fail2ban-client status wordpress
```

- Review Nginx and auth logs for matching entries

```
sudo tail -f /var/log/auth.log
```

```

user1@ubuntuweb: ~
user1@ubuntuweb: ~
user1@kali: ~

Apr 26 17:21:00 ubuntuweb sshd[5684]: Received disconnect from 192.168.1.99 port 53222:11: Bye Bye [preauth]
Apr 26 17:21:00 ubuntuweb sshd[5684]: Disconnected from invalid user james 192.168.1.99 port 53222 [preauth]
Apr 26 17:21:00 ubuntuweb sshd[5686]: Received disconnect from 192.168.1.99 port 53234:11: Bye Bye [preauth]
Apr 26 17:21:00 ubuntuweb sshd[5686]: Disconnected from invalid user james 192.168.1.99 port 53234 [preauth]
Apr 26 17:21:00 ubuntuweb sshd[5708]: User testuser99 from 192.168.1.99 not allowed because not listed in AllowUsers
Apr 26 17:21:00 ubuntuweb sshd[5708]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.1.99 user=testuser99
Apr 26 17:21:00 ubuntuweb sshd[5688]: Received disconnect from 192.168.1.99 port 53250:11: Bye Bye [preauth]
Apr 26 17:21:00 ubuntuweb sshd[5688]: Disconnected from invalid user james 192.168.1.99 port 53250 [preauth]
Apr 26 17:21:00 ubuntuweb sshd[5698]: Failed password for invalid user james from 192.168.1.99 port 53330 ssh2
Apr 26 17:21:00 ubuntuweb sshd[5691]: Failed password for invalid user james from 192.168.1.99 port 53286 ssh2
Apr 26 17:21:00 ubuntuweb sshd[5690]: Failed password for invalid user james from 192.168.1.99 port 53282 ssh2
Apr 26 17:21:00 ubuntuweb sshd[5694]: Failed password for invalid user james from 192.168.1.99 port 53302 ssh2
Apr 26 17:21:00 ubuntuweb sshd[5696]: Failed password for invalid user james from 192.168.1.99 port 53316 ssh2
Apr 26 17:21:00 ubuntuweb sshd[5710]: User testuser99 from 192.168.1.99 not allowed because not listed in AllowUsers
Apr 26 17:21:00 ubuntuweb sshd[5712]: User testuser99 from 192.168.1.99 not allowed because not listed in AllowUsers
Apr 26 17:21:00 ubuntuweb sshd[5712]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.1.99 user=testuser99
Apr 26 17:21:00 ubuntuweb sshd[5710]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.1.99 user=testuser99
Apr 26 17:21:00 ubuntuweb sshd[5700]: Failed password for invalid user james from 192.168.1.99 port 53360 ssh2
Apr 26 17:21:00 ubuntuweb sshd[5701]: Failed password for invalid user james from 192.168.1.99 port 53366 ssh2
Apr 26 17:21:00 ubuntuweb sshd[5690]: Received disconnect from 192.168.1.99 port 53282:11: Bye Bye [preauth]
Apr 26 17:21:00 ubuntuweb sshd[5690]: Disconnected from invalid user james 192.168.1.99 port 53282 [preauth]
Apr 26 17:21:00 ubuntuweb sshd[5691]: Received disconnect from 192.168.1.99 port 53286:11: Bye Bye [preauth]
Apr 26 17:21:00 ubuntuweb sshd[5691]: Disconnected from invalid user james 192.168.1.99 port 53286 [preauth]
Apr 26 17:21:00 ubuntuweb sshd[5694]: Received disconnect from 192.168.1.99 port 53302:11: Bye Bye [preauth]
Apr 26 17:21:00 ubuntuweb sshd[5694]: Disconnected from invalid user james 192.168.1.99 port 53302 [preauth]
Apr 26 17:21:00 ubuntuweb sshd[5696]: Received disconnect from 192.168.1.99 port 53316:11: Bye Bye [preauth]
Apr 26 17:21:00 ubuntuweb sshd[5696]: Disconnected from invalid user james 192.168.1.99 port 53316 [preauth]
Apr 26 17:21:00 ubuntuweb sshd[5714]: User testuser99 from 192.168.1.99 not allowed because not listed in AllowUsers
Apr 26 17:21:00 ubuntuweb sshd[5715]: User testuser99 from 192.168.1.99 not allowed because not listed in AllowUsers
Apr 26 17:21:00 ubuntuweb sshd[5704]: Failed password for invalid user james from 192.168.1.99 port 53374 ssh2
Apr 26 17:21:00 ubuntuweb sshd[5718]: User testuser99 from 192.168.1.99 not allowed because not listed in AllowUsers
Apr 26 17:21:00 ubuntuweb sshd[5622]: Failed password for invalid user james from 192.168.1.99 port 53198 ssh2
Apr 26 17:21:00 ubuntuweb sshd[5715]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
```

```
sudo tail -f /var/log/fail2ban.log
```

For Brute-Force with **WPScan**:

```
sudo tail -f /var/log/nginx/access.log
```

```
sudo tail -f /var/log/nginx/error.log
```

## For brute-force **wordpress** with **hydra**:

```
sudo tail -f /var/log/nginx/access.log
```

## Step 11

### Maintain Configuration and Update Regularly

Keep Fail2Ban up to date and validate configurations monthly.

- Regularly update filters, jail configurations
  - Test protections every month
  - Document logs and include in audit reports
  - Apply updates with `sudo apt update && sudo apt upgrade fail2ban`