HACKTHEBOX

LOGIN BRUTE FORCING
CHEAT SHEET

Login Brute Forcing Cheat Sheet

What is Brute Forcing?

A trial-and-error method used to crack passwords, login credentials, or encryption keys by systematically trying every possible combination of characters.

Factors Influencing Brute Force Attacks

- Complexity of the password or key
- Computational power available to the attacker
- Security measures in place

How Brute Forcing Works

1. Start: The attacker initiates the brute force process.
2. Generate Possible Combination: The software generates a potential password or key combination.
3. Apply Combination: The generated combination is attempted against the target system.
4. Check if Successful: The system evaluates the attempted combination.
5. Access Granted (if successful): The attacker gains unauthorized access.
6. End (if unsuccessful): The process repeats until the correct combination is found or the attacker gives up.

Types of Brute Forcing

Attack Type	Description	Best Used When
Simple Brute Force	Tries every possible character combination in a set (e.g., lowercase, uppercase, numbers, symbols).	When there is no prior information about the password.

Attack Type	Description	Best Used When
Dictionary Attack	Uses a pre-compiled list of common passwords.	When the password is likely weak or follows common patterns.
Hybrid Attack	Combines brute force and dictionary attacks, adding numbers or symbols to dictionary words.	When the target uses slightly modified versions of common passwords.
Credential Stuffing	Uses leaked credentials from other breaches to access different services where users may have reused passwords.	When you have a set of leaked credentials, and the target may reuse passwords.
Password Spraying	Attempts common passwords across many accounts to avoid detection.	When account lockout policies are in place.
Rainbow Table Attack	Uses precomputed tables of password hashes to reverse them into plaintext passwords.	When a large number of password hashes need cracking, and storage for tables is available.
Reverse Brute Force	Targets a known password against multiple usernames.	When there's a suspicion of password reuse across multiple accounts.
Distributed Brute Force	Distributes brute force attempts across multiple machines to speed up the process.	When the password is highly complex, and a single machine isn't powerful enough.

Default Credentials

- Default Usernames: Pre-set usernames that are widely known
- Default Passwords: Pre-set, easily guessable passwords that come with devices and software

Device	Username	Password
Linksys Router	admin	admin
Netgear Router	admin	password
TP-Link Router	admin	admin
Cisco Router	cisco	cisco

Device	Username	Password
Ubiquiti UniFi AP	ubnt	ubnt

Brute-Forcing Tools

Hydra

- Fast network login cracker
- Supports numerous protocols
- Uses parallel connections for speed
- Flexible and adaptable
- Relatively easy to use

```
hydra [-l LOGIN|-L FILE] [-p PASS|-P FILE] [-C FILE] -m MODULE [service://server[:PORT][/OPT]]
```

Hydra Service	Service/Protocol	Description	Example Command
ftp	File Transfer Protocol (FTP)	Used to brute-force login credentials for FTP services, commonly used to transfer files over a network.	`hydra -l admin -P /path/to/password_list.txt ftp://192.168.1.100`
ssh	Secure Shell (SSH)	Targets SSH services to brute-force credentials, commonly used for secure remote login to systems.	`hydra -l root -P /path/to/password_list.txt ssh://192.168.1.100`

Hydra Service	Service/Protocol	Description	Example Command
http-get/post	HTTP Web Services	Used to brute-force login credentials for HTTP web login forms using either GET or POST requests.	`hydra -l admin -P /path/to/password_list.txt 127.0.0.1 http-post-form "/login.php:user=^USER^&pass=^PASS^:F=incorrect"`

Medusa

- Fast, massively parallel, modular login brute-forcer
- Supports a wide array of services

```
medusa [-h host|-H file] [-u username|-U file] [-p password|-P file] [-C file] -M module [OPT]
```

Medusa Module	Service/Protocol	Description	Example Command
ssh	Secure Shell (SSH)	Brute force SSH login for the `admin` user.	`medusa -h 192.168.1.100 -u admin -P passwords.txt -M ssh`
ftp	File Transfer Protocol (FTP)	Brute force FTP with multiple usernames and passwords using 5 parallel threads.	`medusa -h 192.168.1.100 -U users.txt -P passwords.txt -M ftp -t 5`
rdp	Remote Desktop Protocol (RDP)	Brute force RDP login.	`medusa -h 192.168.1.100 -u admin -P passwords.txt -M rdp`
http-get	HTTP Web Services	Brute force HTTP Basic Authentication.	`medusa -h www.example.com -U users.txt -P passwords.txt -M http -m GET`
ssh	Secure Shell (SSH)	Stop after the first valid SSH login is found.	`medusa -h 192.168.1.100 -u admin -P passwords.txt -M ssh -f`

Custom Wordlists

Username Anarchy generates potential usernames based on a target's name.

Command	Description
`username-anarchy Jane Smith`	Generate possible usernames for "Jane Smith"
`username-anarchy -i names.txt`	Use a file (`names.txt`) with names for input. Can handle space, CSV, or TAB delimited names.
`username-anarchy -a --country us`	Automatically generate usernames using common names from the US dataset.
`username-anarchy -l`	List available username format plugins.
`username-anarchy -f format1,format2`	Use specific format plugins for username generation (comma-separated).
`username-anarchy -@ example.com`	Append `@example.com` as a suffix to each username.
`username-anarchy --case-insensitive`	Generate usernames in case-insensitive (lowercase) format.

CUPP (Common User Passwords Profiler) creates personalized password wordlists based on gathered intelligence.

Command	Description
`cupp -i`	Generate wordlist based on personal information (interactive mode).
`cupp -w profiles.txt`	Generate a wordlist from a predefined profile file.
`cupp -l`	Download popular password lists like `rockyou.txt`.

Password Policy Filtering

Password policies often dictate specific requirements for password strength, such as minimum length, inclusion of certain character types, or exclusion of common patterns. `grep` combined with regular expressions can be a powerful tool for filtering wordlists to identify passwords that adhere to a given policy. Below is a table summarizing common password policy requirements and the corresponding `grep` regex patterns to apply:

Policy Requirement	Grep Regex Pattern	Explanation
Minimum Length (e.g., 8 characters)	`grep -E '^.{8,}$' wordlist.txt`	`^` matches the start of the line, `.` matches any character, `{8,}` matches 8 or more occurrences, `$` matches the end of the line.
At Least One Uppercase Letter	`grep -E '[A-Z]' wordlist.txt`	`[A-Z]` matches any uppercase letter.
At Least One Lowercase Letter	`grep -E '[a-z]' wordlist.txt`	`[a-z]` matches any lowercase letter.
At Least One Digit	`grep -E '[0-9]' wordlist.txt`	`[0-9]` matches any digit.
At Least One Special Character	`grep -E '[!@#$%^&*()_+-=[]{};':"\,.<>/?]' wordlist.txt`	`[!@#$%^&*()_+-=[]{};':"\,.<>/?]` matches any special character (symbol).
No Consecutive Repeated Characters	`grep -E '(.)\1' wordlist.txt`	`(.)` captures any character, `\1` matches the previously captured character. This pattern will match any line with consecutive repeated characters. Use `grep -v` to invert the match.
Exclude Common Patterns (e.g., "password")	`grep -v -i 'password' wordlist.txt`	`-v` inverts the match, `-i` makes the search case-insensitive. This pattern will exclude any line containing "password" (or "Password", "PASSWORD", etc.).
Exclude Dictionary Words	`grep -v -f dictionary.txt wordlist.txt`	`-f` reads patterns from a file. `dictionary.txt` should contain a list of common dictionary words, one per line.
Combination of Requirements	`grep -E '^.{8,}$' wordlist.txt | grep -E '[A-Z]'`	This command filters a wordlist to meet multiple password policy requirements. It first ensures that each word has a minimum length of 8 characters (`grep -E '^.{8,}$'`), and then it pipes the result into a second `grep` command to match only words that contain at least one uppercase letter (`grep -E '[A-Z]'`). This approach ensures the filtered passwords meet both the length and uppercase letter criteria.