

DIGITAL ASSIGNMENT – 2

TOPIC : JOHN THE RIPPER

TEAM MEMBERS:

Reshma.P 17MIS1009

P.Nitya Sree 17MIS1007

Blessy Boban 17mis1014

Video Link:

https://drive.google.com/file/d/1ff8XPWK0EPP9w_FpW5-UHqUQnAP9DEJM/view?usp=sharing

John the ripper:

- John the Ripper is a free password cracking software tool.
- Originally developed for the Unix operating system, it can run on fifteen different platforms (eleven of which are architecture-specific versions of Unix, DOS, Win32, BeOS, and OpenVMS).
- It is among the most frequently used password testing and breaking programs as it combines a number of password crackers into one package, autodetects password hash types, and includes a customizable cracker.

- It can be run against various encrypted password formats including several crypt password hash types most commonly found on various Unix versions (based on DES, MD5, or Blowfish), Kerberos AFS, and Windows NT/2000/XP/2003 LM hash. Additional modules have extended its ability to include MD4-based password hashes and passwords stored in LDAP, MySQL, and others.

What is John the Ripper Used for?

John the ripper is primarily a password cracker used during pentesting exercises that can help IT staff spot weak passwords and poor password policies.

How Password cracking is done by john the ripper?

John The Ripper supports several common encryption technologies out-of-the-box for UNIX and Windows-based systems. (ed. Mac is UNIX based). John the ripper autodetects the encryption on the hashed data and compares it against a large plain-text file that contains popular passwords, hashing each password, and then stopping it when it finds a match

John the ripper also includes its own wordlists of common passwords for 20+ languages. These wordlists provide John the ripper with thousands of possible passwords from which it can generate the corresponding hash values to make a high-value guess of the target password. Since most people choose easy-to-remember passwords, John the ripper is often very effective even with its out-of-the-box wordlists of passwords.

John the ripper is included in the pentesting versions of Kali Linux.

Various modes in John the ripper :

Single crack mode:

This is the mode you should start cracking with. It will use the login names, "GECOS" / "Full Name" fields, and users' home directory names as candidate passwords, also with a large set of mangling rules applied. Since the information is only used against passwords for the accounts it was taken from (and against password hashes which happened to be assigned the same salt), "single crack" mode is much faster than wordlist mode. This permits for the use of a much larger set of word mangling rules with "single crack", and their use is always enabled with this mode. Successfully guessed passwords are also tried against all loaded password hashes just in case more users have the same password.

Incremental mode:

This is the most powerful cracking mode, it can try all possible character combinations as passwords. However, it is assumed that cracking with this mode will never terminate because of the number of combinations being too large (actually, it will terminate if you set a low password length limit or make it use a small charset), and you'll have to interrupt it earlier.

That's one reason why this mode deals with trigraph frequencies, separately for each character position and for each password length, to crack as many passwords as possible within a limited time.

To use the mode you need a specific definition for the mode's parameters, including password length limits and the charset to use. These parameters are defined in the configuration file sections called [Incremental:MODE], where MODE is any name that you assign to the mode (it's the name that you will need to specify on John's command line). You can either use a pre-defined incremental mode definition or define a custom one.

External mode:

You can define an external cracking mode for use with John. This is done with the configuration file sections called [List.External:MODE], where MODE is any name that you assign to the mode. The section should contain program code of some functions that John will use to generate the candidate passwords it tries. The functions are coded in a subset of C and are compiled by John at startup when you request the particular external mode on John's command line.

Execution:

1)raw-sha1 :

- the password is encrypted using sha1 and encrypted password is placed inside the file
- here, password is abc101
- sha1 encrypted password is bd9a944306b6a41cb61302f67ffac4f26ee38088
- The code is now typed in the terminal i.e sudo john –format=raw-sha1 cracked.txt
- John the ripper now cracks the encrypted password and displays it

File name :

Cracked.txt

content inside the file:

Pass:bd9a944306b6a41cb61302f67ffac4f26ee38088

```

kali@kali:~$ sudo john --format=raw-sha1 cracked.txt
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-SHA1 [SHA1 128/128 SSE2 4x])
Warning: no OpenMP support for this hash type, consider --fork=4
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 2 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 4 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 1 candidate buffered for the current salt, minimum 8 needed for performance.
Warning: Only 2 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 4 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 1 candidate buffered for the current salt, minimum 8 needed for performance.
Warning: Only 4 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 1 candidate buffered for the current salt, minimum 8 needed for performance.
Warning: Only 4 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 3 candidates buffered for the current salt, minimum 8 needed for performance.
Further messages of this type will be suppressed.
To see less of these warnings, enable 'RelaxKPCWarningCheck' in john.conf
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist
Warning: Only 2 candidates left, minimum 4 needed for performance.
Proceeding with incremental:ASCII
abc101 (pass)
1g 0:00:00:01 DONE 3/3 (2020-10-18 13:27) 0.6578g/s 112268p/s 112268c/s 112268C/s abc121..abc101
Use the "--show --format=Raw-SHA1" options to display all of the cracked passwords reliably
Session completed
kali@kali:~$

```

2)Ripemd:

- the password is encrypted using ripemd and encrypted password is placed inside the file
- here, password is jon101
- ripemd encrypted password is 7f69489884da244c4437db7133dce240
- The code is now typed in the terminal i.e sudo john --format=ripemd c3.txt
- John the ripper now cracks the encrypted password and displays it

File name :

C3.txt

content inside the file:

Pass: 7f69489884da244c4437db7133dce240

```
File Actions Edit View Help
kali@kali: ~ 1371085a 123cbf6f11 kali@kali: ~ 2115083607 e68199c5

kali@kali:~$ sudo john --format=raw-ripemd-128 c3.txt
[sudo] password for kali:
Unknown ciphertext format name requested
kali@kali:~$ sudo john --format=ripemd-128 c3.txt
Using default input encoding: UTF-8
Loaded 1 password hash (ripemd-128, RIPEMD 128 [32/64])
Warning: no OpenMP support for this hash type, consider --fork=4
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 2 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 5 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 2 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 4 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 5 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 4 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 5 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 7 candidates buffered for the current salt, minimum 8 needed for performance.
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist
Proceeding with incremental:ASCII
jon101 (pass)
1g 0:00:00:28 DONE 3/3 (2020-10-20 05:55) 0.03554g/s 1372Kp/s 1372Kc/s 1372KC/s joev6f..jon14u
Use the "--show" option to display all of the cracked passwords reliably
Session completed
kali@kali:~$
```

3)whirlpool:

- the password is encrypted using Whirlpool and encrypted password is placed inside the file
- here, password is jon101
- Whirlpool encrypted password is
2d1beab53b6c8f0c2688e135446aa04a06888c359a1f53b610b356a59af82c489edf054fdc9dc640d87997c57e64c9cb56e604d8dce78c4ed9d36a3236b528c8
- The code is now typed in the terminal i.e sudo john --format=whirlpool c4.txt
- John the ripper now cracks the encrypted password and displays it

File name :

C4.txt

content inside the file:

Pass:2d1beab53b6c8f0c2688e135446aa04a06888c359a1f53b610b356a59af82c489edf054fdc9dc640d87997c57e64c9cb56e604d8dce78c4ed9d36a3236b528c8

```

File Actions Edit View Help
kali@kali:~$ sudo john --format=whirlpool c4.txt
[sudo] password for kali:
Using default input encoding: UTF-8
Loaded 1 password hash (whirlpool [WHIRLPOOL 32/64])
Warning: poor OpenMP scalability for this hash type, consider --fork=4
Will run 4 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 2 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 5 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 2 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 4 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 5 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 4 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 5 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 7 candidates buffered for the current salt, minimum 8 needed for performance.
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist
Proceeding with incremental:ASCII
jon101 (pass)
1g 0:00:00:37 DONE 3/3 (2020-10-20 06:01) 0.02675g/s 1033Kp/s 1033Kc/s 1033KC/s jalfye..jehjug
Use the "--show" option to display all of the cracked passwords reliably
Session completed
kali@kali:~$

```

4)sha256:

- the password is encrypted using sha1 and encrypted password is placed inside the file
- here, password is jon101
- sha256 encrypted password is
8f938aded19a051371d85a2161c3cbf6f13eb15efa79d11508369984ae8199c5
- The code is now typed in the terminal i.e sudo john --format=raw-sha256 c5.txt
- John the ripper now cracks the encrypted password and displays it

File name :

C5.txt

content inside the file:

Pass: 8f938aded19a051371d85a2161c3cbf6f13eb15efa79d11508369984ae8199c5

```
File Actions Edit View Help
kali@kali: ~ john --format=raw-sha256 c5.txt kali@kali: ~
kali@kali:~$ sudo john --format=raw-sha256 c5.txt
[sudo] password for kali:
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-SHA256 [SHA256 128/128 SSE2 4x])
Warning: poor OpenMP scalability for this hash type, consider --fork=4
Will run 4 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 2 candidates buffered for the current salt, minimum 16 needed for performance.
Warning: Only 15 candidates buffered for the current salt, minimum 16 needed for performance.
Warning: Only 7 candidates buffered for the current salt, minimum 16 needed for performance.
Warning: Only 11 candidates buffered for the current salt, minimum 16 needed for performance.
Warning: Only 7 candidates buffered for the current salt, minimum 16 needed for performance.
Warning: Only 6 candidates buffered for the current salt, minimum 16 needed for performance.
Almost done: Processing the remaining buffered candidate passwords, if any.
Warning: Only 2 candidates buffered for the current salt, minimum 16 needed for performance.
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist
Proceeding with incremental:ASCII
jon101 (pass)
1g 0:00:00:17 DONE 3/3 (2020-10-20 06:11) 0.05567g/s 2150Kp/s 2150Kc/s 2150KC/s byacue..jehjug
Use the "--show --format=Raw-SHA256" options to display all of the cracked passwords reliably
Session completed
kali@kali:~$
```

5)md4:

- the password is encrypted using sha1 and encrypted password is placed inside the file
- here, password is jon101
- md4 encrypted password is 5a79144692d0081dba7d235b1d24d8e1
- The code is now typed in the terminal i.e sudo john --format=raw-md4 c6.txt
- John the ripper now cracks the encrypted password and displays it

File name :

C6.txt

content inside the file:

Pass: 5a79144692d0081dba7d235b1d24d8e1

```
kali@kali: ~
kali@kali: ~
kali@kali: ~

kali@kali:~$ sudo john --format=raw-md4 c6.txt
[sudo] password for kali:
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD4 [MD4 128/128 SSE2 4x3])
Warning: no OpenMP support for this hash type, consider --fork=4
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 2 candidates buffered for the current salt, minimum 12 needed for performance.
Warning: Only 4 candidates buffered for the current salt, minimum 12 needed for performance.
Warning: Only 7 candidates buffered for the current salt, minimum 12 needed for performance.
Warning: Only 9 candidates buffered for the current salt, minimum 12 needed for performance.
Warning: Only 8 candidates buffered for the current salt, minimum 12 needed for performance.
Warning: Only 4 candidates buffered for the current salt, minimum 12 needed for performance.
Almost done: Processing the remaining buffered candidate passwords, if any.
Warning: Only 8 candidates buffered for the current salt, minimum 12 needed for performance.
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist
Proceeding with incremental:ASCII
jon101 (pass)
ig 0:00:00:12 DONE 3/3 (2020-10-20 06:19) 0.08298g/s 3205Kp/s 3205Kc/s 3205KC/s joev6f..joaak1
Use the "--show" option to display all of the cracked passwords reliably
Session completed
kali@kali:~$
```

6)cracking 2 files at a time:

- the password is encrypted using md5 and encrypted password is placed inside the file
- here, password is one101 in c71.txt and two101 in c72.txt
- md5 encrypted password is d67eb21af516685c45e3dd2553398ffb in c71.txt and b8c38b218d0ca490dd2b1497183735e7 in c72.txt
- The code is now typed in the terminal i.e sudo john --form=raw-md5 c71.txt c72.txt
- John the ripper now cracks the encrypted password and displays it

File name :

C71.txt

C72.txt

content inside the file:

In C71.txt:

Pass: d67eb21af516685c45e3dd2553398ffb

In C72.txt:

Pass: b8c38b218d0ca490dd2b1497183735e7


```
File Actions Edit View Help
kali@kali: ~ kali@kali: ~ kali@kali: ~ kali@kali: ~
kali@kali:~$ sudo john --form=raw-md5 c71.txt c72.txt
[sudo] password for kali:
Using default input encoding: UTF-8
Loaded 2 password hashes with no different salts (Raw-MD5 [MD5 128/128 SSE2 4x3])
Warning: no OpenMP support for this hash type, consider --fork=4
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 2 candidates buffered for the current salt, minimum 12 needed for performance.
Warning: Only 4 candidates buffered for the current salt, minimum 12 needed for performance.
Warning: Only 7 candidates buffered for the current salt, minimum 12 needed for performance.
Warning: Only 9 candidates buffered for the current salt, minimum 12 needed for performance.
Warning: Only 8 candidates buffered for the current salt, minimum 12 needed for performance.
Warning: Only 4 candidates buffered for the current salt, minimum 12 needed for performance.
Almost done: Processing the remaining buffered candidate passwords, if any.
Warning: Only 8 candidates buffered for the current salt, minimum 12 needed for performance.
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist
Proceeding with incremental:ASCII
two101 (pass)
one101 (pass)
2g 0:00:02:44 DONE 3/3 (2020-10-20 06:29) 0.01216g/s 8754Kp/s 8754Kc/s 8851KC/s oneoy5..one156
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed
kali@kali:~$
```

7)View all formats:

John the Ripper supports much encryption some of which we showed above. To view all the formats it supports use the code `sudo john --list=formats`

```
kali@kali:~$ sudo john --list=formats
[sudo] password for kali:
descript, bsdictcrypt, md5crypt, md5crypt-long, bccrypt, sccrypt, LM, AFS,
tripcode, AndroidBackup, adxcrypt, agilekeychain, aix-ssh1, aix-ssh256,
aix-ssh512, andOTP, ansible, argon2, as400-des, as400-ssh1, asa-md5,
AxCrypt, AzureAD, BestCrypt, bfeegg, Bitcoin, BitLocker, bitshares, Bitwarden,
BKS, Blackberry-ES10, WoWSRP, Blockchain, chap, Clipperz, cloudkeychain,
dynamic_n, cq, CRC32, sha1crypt, sha256crypt, sha512crypt, Citrix_NS10,
dahua, dashlane, diskcryptor, Django, django-sccrypt, dmd5, dmg, dominosec,
dominosec8, DPAPImk, dragonfly3-32, dragonfly3-64, dragonfly4-32,
dragonfly4-64, Drupal7, eCryptfs, eigrp, electrum, EncFS, enpass, EPI,
EPiServer, ethereum, fde, Fortigate256, Fortigate, FormSpring, FVDE, geli,
gost, gpg, HAVAL-128-4, HAVAL-256-3, hdaa, hMailServer, hsrp, IKE, ipb2,
itunes-backup, iwork, KeePass, keychain, keyring, keystore, known_hosts,
krb4, krb5, krb5asrep, krb5pa-sha1, krb5tgs, krb5-17, krb5-18, krb5-3,
kwallet, lp, lpcli, leet, lotus5, lotus85, LUKS, MD2, mdc2, MediaWiki,
monero, money, MongoDB, scram, Mozilla, mscash, mscash2, MSCHAPv2,
mschapv2-naive, krb5pa-md5, mssql, mssql05, mssql12, multibit, mysqlna,
mysql-sha1, mysql, net-ah, nethalflm, netlm, netlmv2, net-md5, netntlmv2,
netntlm, netntlm-naive, net-sha1, nk, notes, md5ns, nsec3, NT, o10glogon,
o3logon, o5logon, ODF, Office, oldoffice, OpenBSD-SoftRAID, openssl-enc,
oracle, oracle11, Oracle12C, osc, ospf, Padlock, Palshop, Panama,
PBKDF2-HMAC-MD4, PBKDF2-HMAC-MD5, PBKDF2-HMAC-SHA1, PBKDF2-HMAC-SHA256,
PBKDF2-HMAC-SHA512, PDF, PEM, pfx, pgpdisk, pgpsda, pgpwde, phpass, PHPS,
PHPS2, pix-md5, PKZIP, po, postgres, PST, PuTTY, pwsafe, qnx, RACF,
RACF-KDAES, radius, RAdmin, RAKP, rar, RAR5, Raw-SHA512, Raw-Blake2,
Raw-Keccak, Raw-Keccak-256, Raw-MD4, Raw-MD5, Raw-MD5u, Raw-SHA1,
Raw-SHA1-AxCrypt, Raw-SHA1-Linkedin, Raw-SHA224, Raw-SHA256, Raw-SHA3,
```

```
Raw-SHA384, ripemd-128, ripemd-160, rsvp, Siemens-S7, Salted-SHA1, SSHA512,
sapb, sapg, saph, sappse, securezip, 7z, Signal, SIP, skein-256, skein-512,
skey, SL3, Snefru-128, Snefru-256, LastPass, SNMP, solarwinds, SSH, sspr,
STRIP, SunMD5, SybaseASE, Sybase-PROP, tacacs-plus, tcp-md5, telegram, tezos,
Tiger, tc_aes_xts, tc_ripemd160, tc_ripemd160boot, tc_sha512, tc_whirlpool,
vdi, OpenVMS, vmx, VNC, vtp, wbb3, whirlpool, whirlpool0, whirlpool1, wpapsk,
wpapsk-pmk, xmpp-scam, xsha, xsha512, ZIP, ZipMonster, plaintext, has-160,
HMAC-MD5, HMAC-SHA1, HMAC-SHA224, HMAC-SHA256, HMAC-SHA384, HMAC-SHA512,
dummy, crypt
kali@kali:~$
```

8) Checking Password Complexity with John the Ripper

Before you begin attempting password cracks, you should check the efficiency of John on your system by running it in test mode. The report tells you how many username/password combinations per second (c/s) your system will theoretically run for each password hash encryption type.

Use the code `sudo john --test`

```
kali@kali:~$ sudo john --test
Will run 4 OpenMP threads
Benchmarking: descript, traditional crypt(3) [DES 128/128 SSE2]... (4xOMP) DONE
Warning: "Many salts" test limited: 82/256
Many salts: 1330K c/s real, 486771 c/s virtual
Only one salt: 1114K c/s real, 357087 c/s virtual

Benchmarking: bsdictcrypt, BSDI crypt(3) ("_J9..", 725 iterations) [DES 128/128 SSE2]... (4xOMP) DONE
Speed for cost 1 (iteration count) of 725
Warning: "Many salts" test limited: 62/256
Many salts: 62859 c/s real, 22433 c/s virtual
Only one salt: 58368 c/s real, 21147 c/s virtual

Benchmarking: md5crypt, crypt(3) $1$ (and variants) [MD5 128/128 SSE2 4x3]... (4xOMP) DONE
Warning: "Many salts" test limited: 57/256
Many salts: 10729 c/s real, 4696 c/s virtual
Only one salt: 11976 c/s real, 4957 c/s virtual

Benchmarking: md5crypt-long, crypt(3) $1$ (and variants) [MD5 32/64]... (4xOMP) DONE
```

9)find the encryption algorithm automatically and Crack password if encryption algorithm is not mentioned by user

Even if you don't mention the encryption type , john the ripper can automatically find the encryption algorithm and crack the password

- the password is encrypted using any encryption type and encrypted password is placed inside the file
- here, password is zab in new.txt
- encrypted password is bdd5a5f42152ba789e24c1ef8132eb16704ed8a4
- The code is now typed in the terminal i.e `sudo john new.txt`

- John the ripper now cracks finds the encryption algorithm by its own, cracks the encrypted password and displays it

File name: New.txt

Content inside file:

Pass: bdd5a5f42152ba789e24c1ef8132eb16704ed8a4

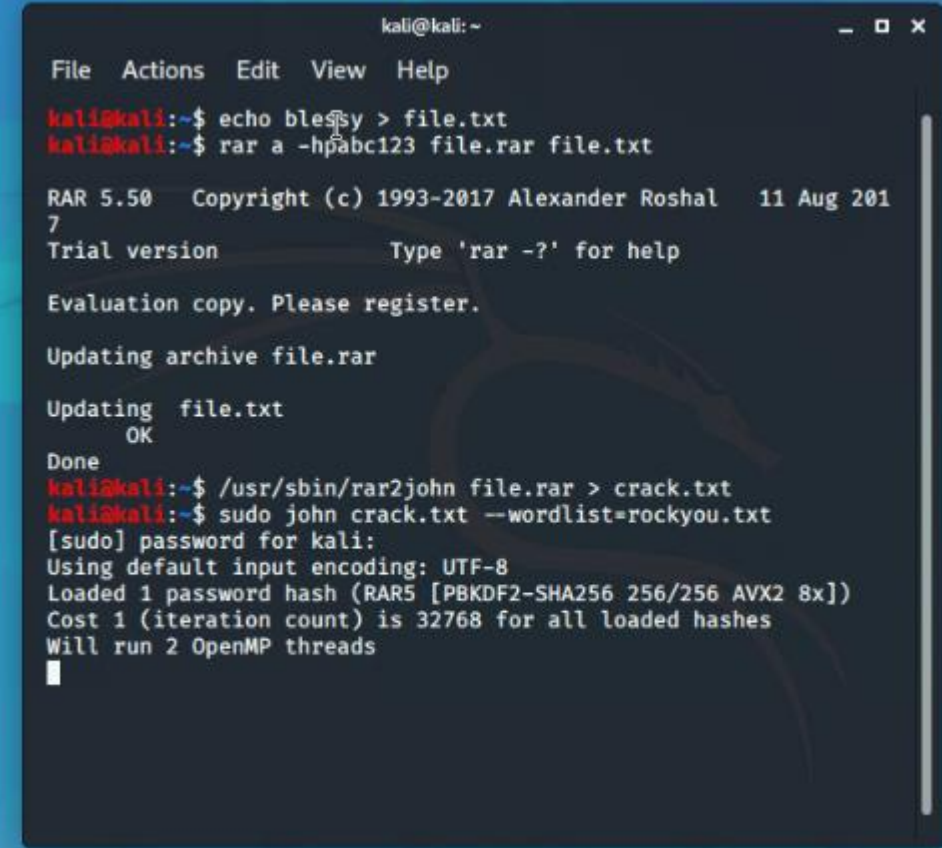
```
kali@kali:~$ sudo john new.txt
Warning: detected hash type "Raw-SHA1", but the string is also recognized as "Raw-SHA1-AxCrypt"
Use the "--format=Raw-SHA1-AxCrypt" option to force loading these as that type instead
Warning: detected hash type "Raw-SHA1", but the string is also recognized as "Raw-SHA1-Linkedin"
Use the "--format=Raw-SHA1-Linkedin" option to force loading these as that type instead
Warning: detected hash type "Raw-SHA1", but the string is also recognized as "ripemd-160"
Use the "--format=ripemd-160" option to force loading these as that type instead
Warning: detected hash type "Raw-SHA1", but the string is also recognized as "has-160"
Use the "--format=has-160" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-SHA1 [SHA1 128/128 SSE2 4x])
Warning: no OpenMP support for this hash type, consider --fork=4
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 2 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 4 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 1 candidate buffered for the current salt, minimum 8 needed for performance.
Warning: Only 2 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 4 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 1 candidate buffered for the current salt, minimum 8 needed for performance.
Warning: Only 4 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 3 candidates buffered for the current salt, minimum 8 needed for performance.
Further messages of this type will be suppressed.
```

```
To see less of these warnings, enable 'RelaxKPCWarningCheck' in john.conf
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist
Warning: Only 2 candidates left, minimum 4 needed for performance.
Proceeding with incremental:ASCII
zab (pass)
1g 0:00:00:41 DONE 3/3 (2020-10-18 14:36) 0.02383g/s 2687Kp/s 2687Kc/s 2687KC/s zar..zah
Use the "--show --format=Raw-SHA1" options to display all of the cracked passwords reliably
Session completed
kali@kali:~$
```

10)Cracking the RAR Password Hash

- Write the contents into the file.txt with echo
 - a = Add files to archive
 - hp[password] = Encrypt both file data and headers

- After creating the password it will show a prompt to enter password to open the file.rar
- Now John cannot directly crack this key after creating password
- So we need to change its format, which can be done using a John utility called "rar2john".
- We can use John the Ripper to crack the hash after changing the format



```

kali@kali: ~
File Actions Edit View Help
kali@kali:~$ echo bles$y > file.txt
kali@kali:~$ rar a -hpabc123 file.rar file.txt

RAR 5.50 Copyright (c) 1993-2017 Alexander Roshal 11 Aug 201
7
Trial version Type 'rar -?' for help

Evaluation copy. Please register.

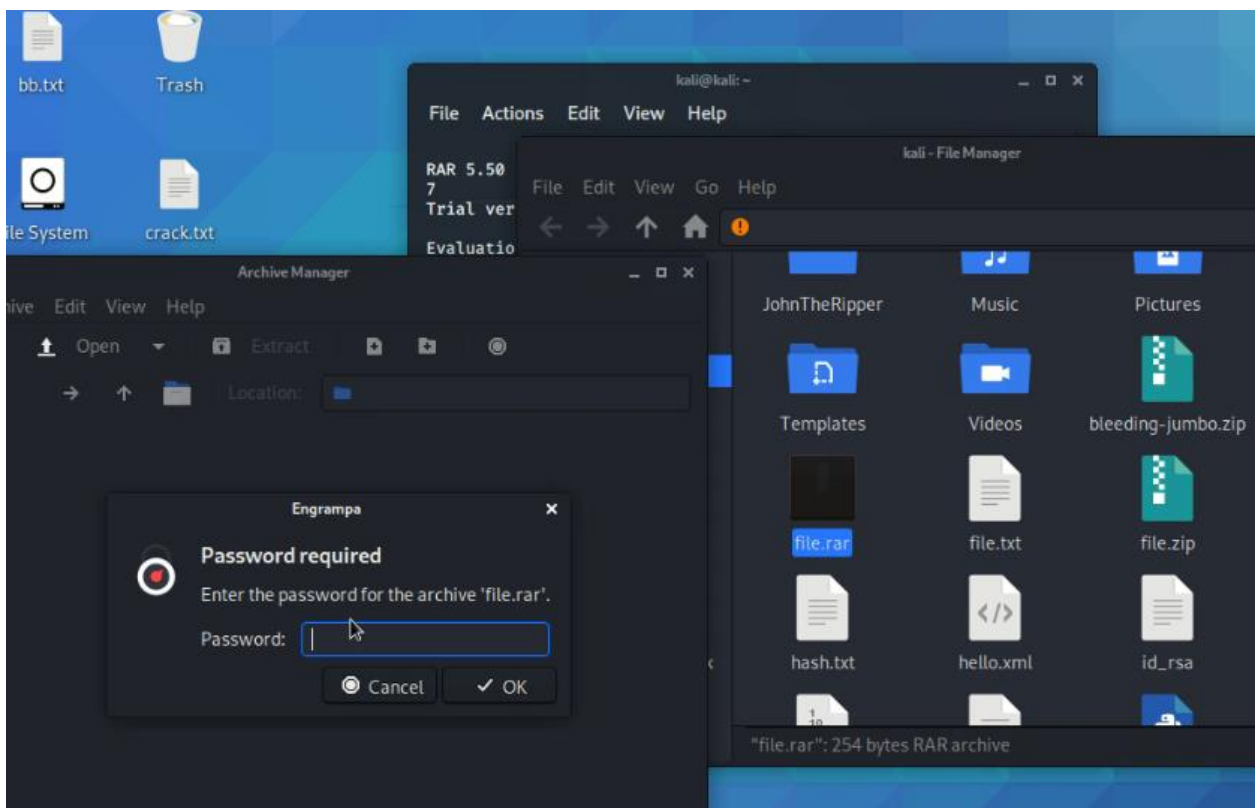
Updating archive file.rar

Updating file.txt
OK
Done
kali@kali:~$ /usr/sbin/rar2john file.rar > crack.txt
kali@kali:~$ sudo john crack.txt --wordlist=rockyou.txt
[sudo] password for kali:
Using default input encoding: UTF-8
Loaded 1 password hash (RAR5 [PBKDF2-SHA256 256/256 AVX2 8x])
Cost 1 (iteration count) is 32768 for all loaded hashes
Will run 2 OpenMP threads
█

```

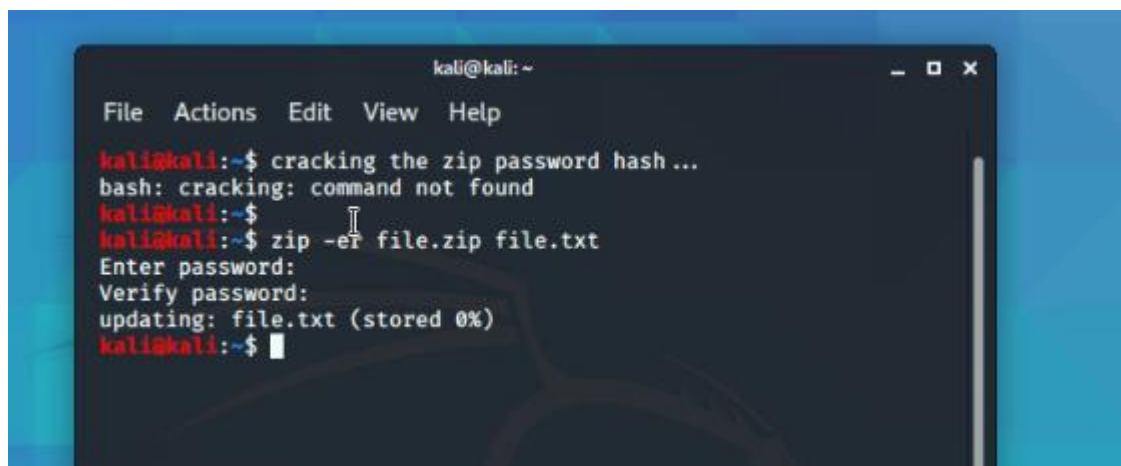


```
kali@kali: ~  
File Actions Edit View Help  
RAR 5.50 Copyright (c) 1993-2017 Alexander Roshal 11 Aug 2017  
Trial version Type 'rar -?' for help  
Evaluation copy. Please register.  
Updating archive file.rar  
Updating file.txt  
OK  
Done  
kali@kali:~$ /usr/sbin/rar2john file.rar > crack.txt  
kali@kali:~$ sudo john crack.txt --wordlist=rockyou.txt  
[sudo] password for kali:  
Using default input encoding: UTF-8  
Loaded 1 password hash (RAR5 [PBKDF2-SHA256 256/256 AVX2 8x])  
Cost 1 (iteration count) is 32768 for all loaded hashes  
Will run 2 OpenMP threads  
Press 'q' or Ctrl-C to abort, almost any other key for status  
abc123 (file.rar)  
1g 0:00:00:00 DONE (2020-10-18 07:17) 1.315g/s 84.21p/s 84.21c/s  
84.21C/s 123456..charlie  
Use the "--show" option to display all of the cracked passwords  
reliably  
Session completed  
kali@kali:~$
```

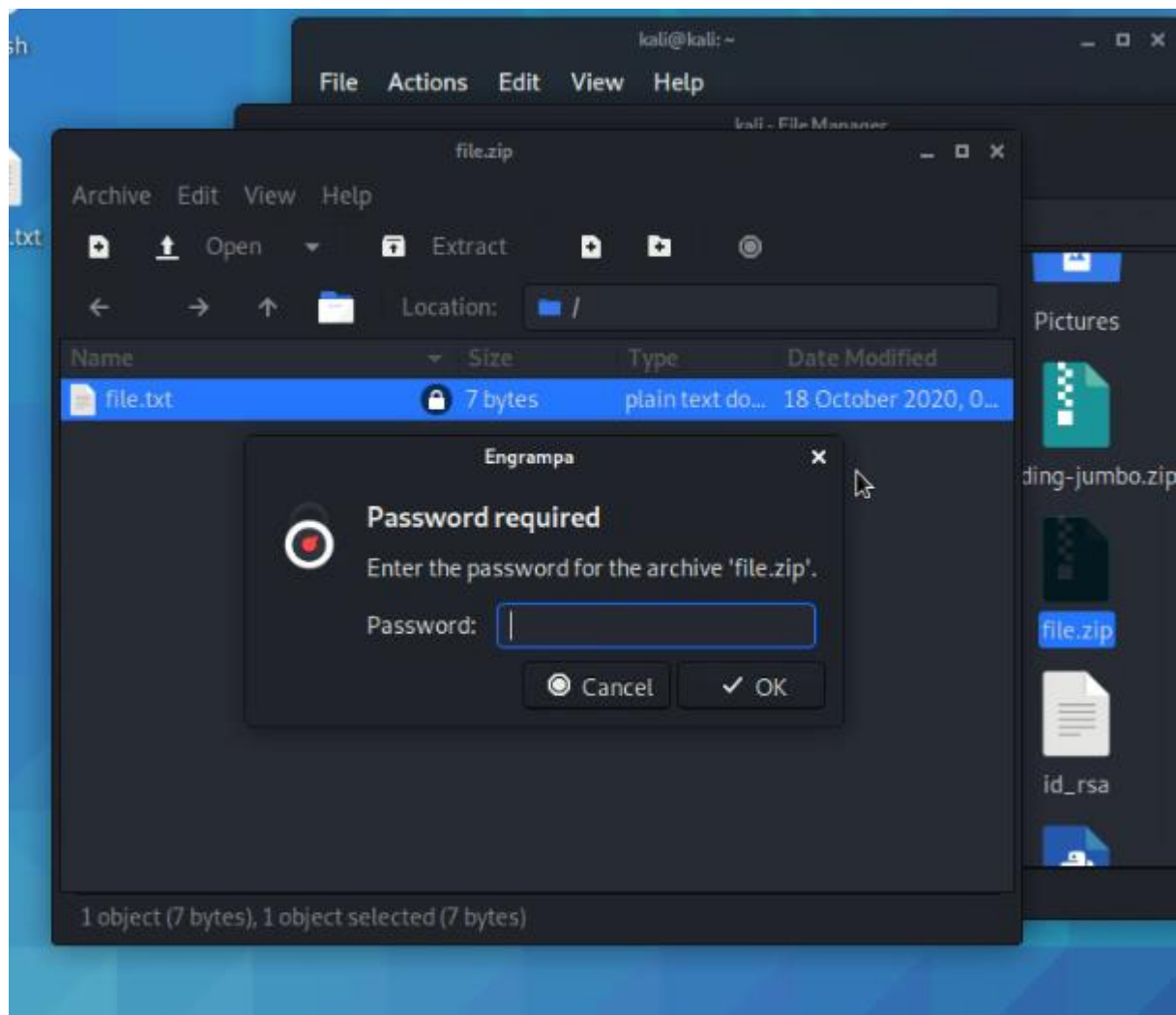


11)Cracking the ZIP Password Hash

- First we can create a encrypted zip file
- Enter and verify the password
- After that if we try to open the file.zip,it won't be open
- So we need to crack the password

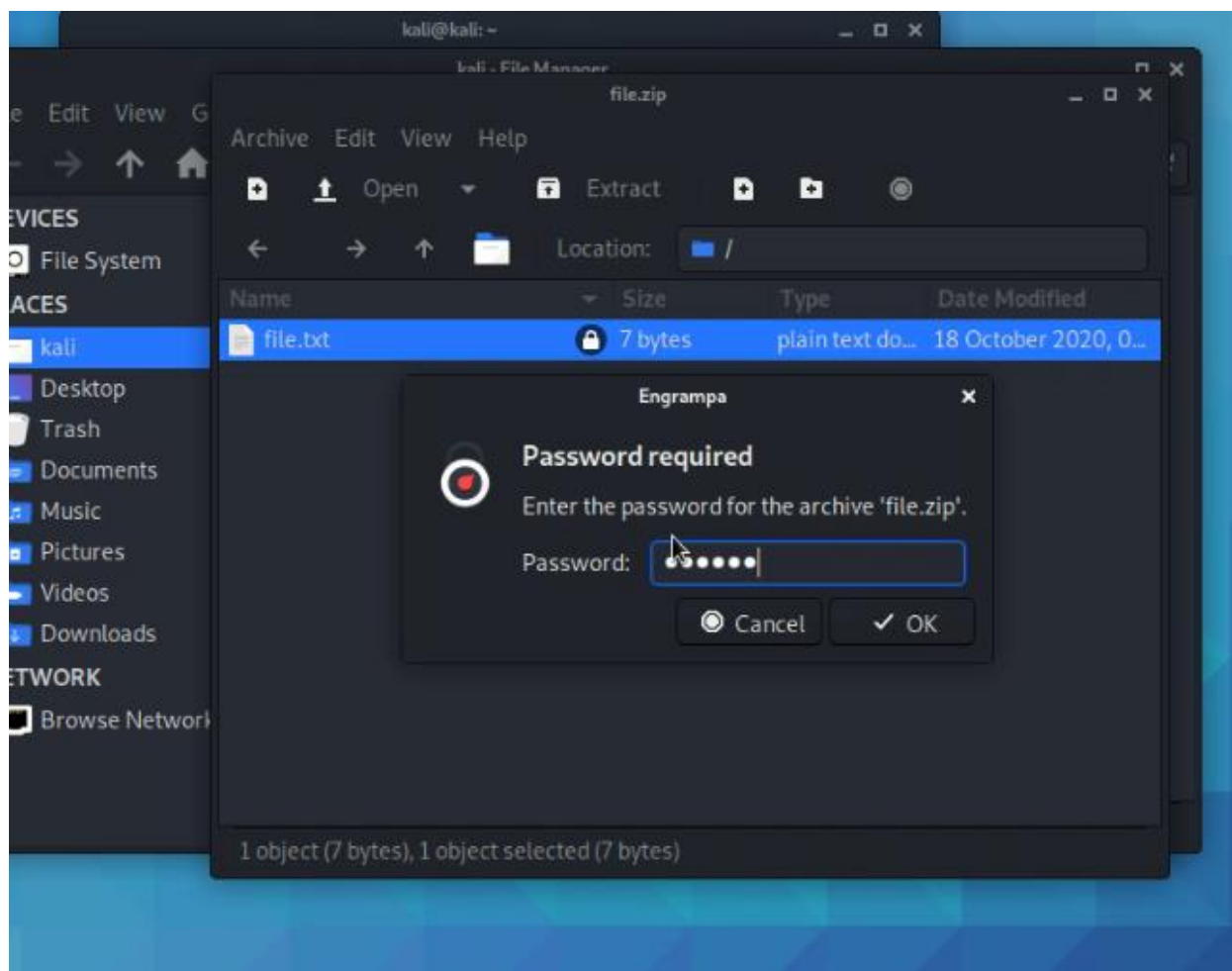
A screenshot of a Kali Linux terminal window. The window title is 'kali@kali: ~'. The menu bar shows 'File', 'Actions', 'Edit', 'View', and 'Help'. The terminal output shows the following sequence of commands and responses:

```
kali@kali:~$ cracking the zip password hash...
bash: cracking: command not found
kali@kali:~$ 
kali@kali:~$ zip -e file.zip file.txt
Enter password:
Verify password:
updating: file.txt (stored 0%)
kali@kali:~$
```



- then will locate where is the zip2john file is
 - e = Encrypt
 - r = Recurse into directories
- To crack the password we can use zip2john which changes its format
- Then the password will be cracked

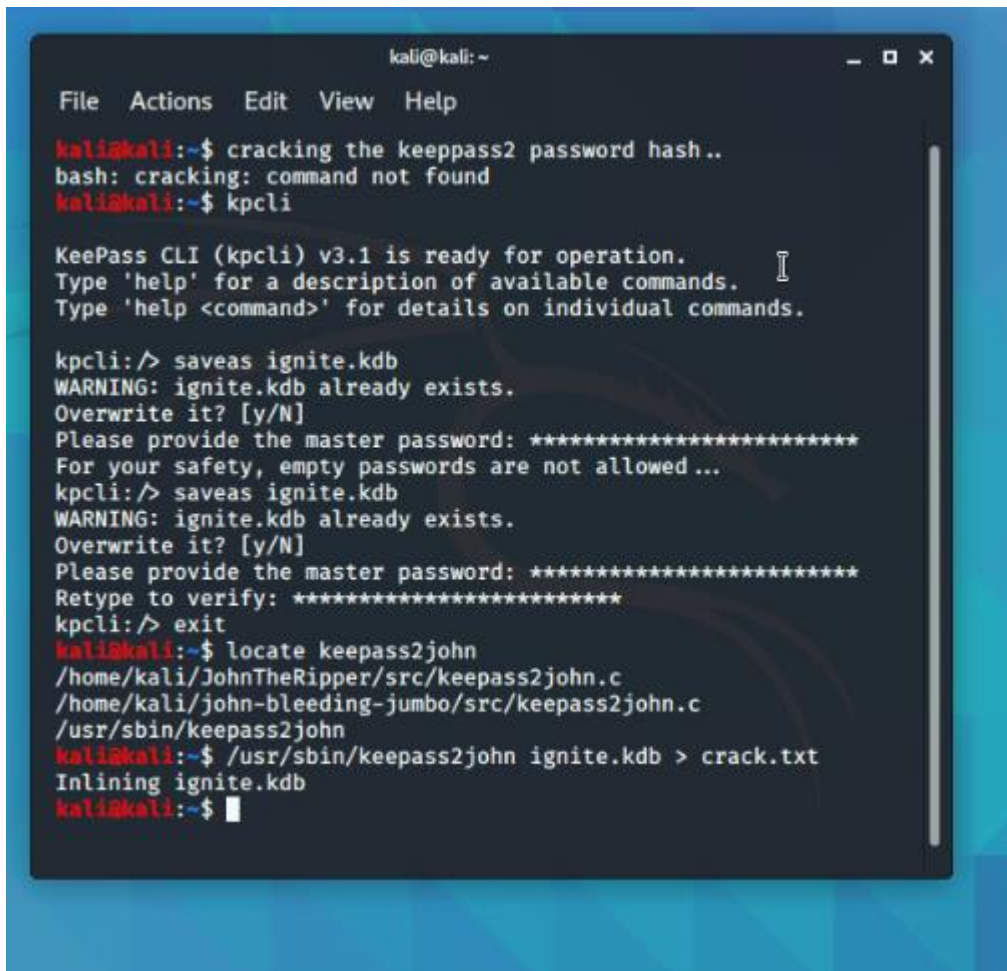
```
kali@kali: ~  
File Actions Edit View Help  
kali@kali:~$ cracking the zip password hash...  
bash: cracking: command not found  
kali@kali:~$  
kali@kali:~$ zip -er file.zip file.txt  
Enter password:  
Verify password:  
updating: file.txt (stored 0%)  
kali@kali:~$ locate zip2john  
/home/kali/JohnTheRipper/src/zip2john.c  
/home/kali/john-bleeding-jumbo/src/zip2john.c  
/usr/sbin/zip2john  
kali@kali:~$ /usr/sbin/zip2john file.zip > crack.txt  
ver 1.0 efh 5455 efh 7875 file.zip/file.txt PKZIP Encr: 2b chk,  
TS_chk, cmplen=19, decmplen=7, crc=9EB44774  
kali@kali:~$ sudo john crack.txt --wordlist=rockyou.txt  
Using default input encoding: UTF-8  
Loaded 1 password hash (PKZIP [32/64])  
Will run 2 OpenMP threads  
Press 'q' or Ctrl-C to abort, almost any other key for status  
abc123 (file.zip/file.txt)  
1g 0:00:00:00 DONE (2020-10-18 07:23) 33.33g/s 136533p/s 136533c  
/s 136533C/s 123456..000000  
Use the "--show" option to display all of the cracked passwords  
reliably  
Session completed  
kali@kali:~$
```

12)Cracking the KeepPass2 Password Hash

- Keepass2 can also be cracked by john the ripper
- To test the cracking of the key, first, we will have to create a set of new keys. To do this we will use a utility that is called "kpcli".

- Now we will create a database file
- By using the command “save as” and naming the database file as ignite.kdb
- enter a passcode to secure it.



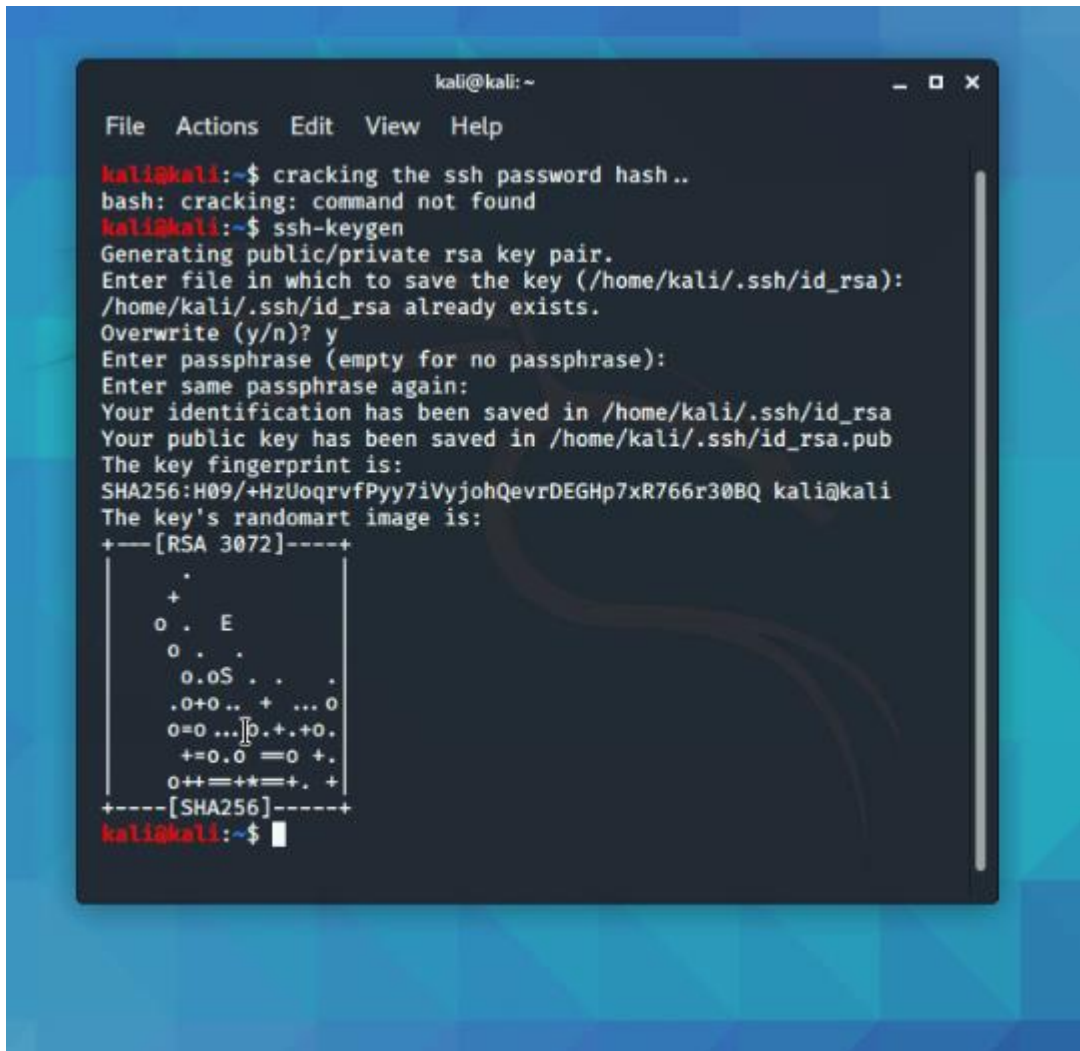
```
kali@kali: ~  
File Actions Edit View Help  
kali@kali:~$ cracking the keepass2 password hash..  
bash: cracking: command not found  
kali@kali:~$ kpcli  
  
KeePass CLI (kpcli) v3.1 is ready for operation.  
Type 'help' for a description of available commands.  
Type 'help <command>' for details on individual commands.  
  
kpcli:/> saveas ignite.kdb  
WARNING: ignite.kdb already exists.  
Overwrite it? [y/N]  
Please provide the master password: *****  
For your safety, empty passwords are not allowed...  
kpcli:/> saveas ignite.kdb  
WARNING: ignite.kdb already exists.  
Overwrite it? [y/N]  
Please provide the master password: *****  
Retype to verify: *****  
kpcli:/> exit  
kali@kali:~$ locate keepass2john  
/home/kali/JohnTheRipper/src/keepass2john.c  
/home/kali/john-bleeding-jumbo/src/keepass2john.c  
/usr/sbin/keepass2john  
kali@kali:~$ /usr/sbin/keepass2john ignite.kdb > crack.txt  
Inlining ignite.kdb  
kali@kali:~$
```

```
kali@kali: ~  
File Actions Edit View Help  
WARNING: ignite.kdb already exists.  
Overwrite it? [y/N]  
Please provide the master password: *****  
Retype to verify: *****  
kpcli:/> exit  
kali@kali:~$ locate keepass2john  
/home/kali/JohnTheRipper/src/keepass2john.c  
/home/kali/john-bleeding-jumbo/src/keepass2john.c  
/usr/sbin/keepass2john  
kali@kali:~$ /usr/sbin/keepass2john ignite.kdb > crack.txt  
Inlining ignite.kdb  
kali@kali:~$ sudo john crack.txt --wordlist=rockyou.txt  
Using default input encoding: UTF-8  
Loaded 1 password hash (KeePass [SHA256 AES 32/64])  
Cost 1 (iteration count) is 50000 for all loaded hashes  
Cost 2 (version) is 1 for all loaded hashes  
Cost 3 (algorithm [0=AES, 1=TwoFish, 2=ChaCha]) is 0 for all loaded hashes  
Will run 2 OpenMP threads  
Press 'q' or Ctrl-C to abort, almost any other key for status  
1abc123 (ignite.kdb)  
1g 0:00:00:00 DONE (2020-10-18 07:28) 1.639g/s 26.22p/s 26.22c/s  
26.22C/s 12345678..jessica  
Use the "--show" option to display all of the cracked passwords reliably  
Session completed  
kali@kali:~$
```

13)Cracking the SSH Password Hash

- John the Ripper can crack the SSH private key which is created in RSA Encryption. first, we will have to create a set of new private keys , to test the cracking of the private key. To do this we will use a utility that comes with ssh, called “ssh-keygen”.

- When we try to open the id_rsa file then there will be prompt opening asking us to type the password.
- Password can be cracked by ssh2john which will change its format.



```

kali@kali: ~
File Actions Edit View Help

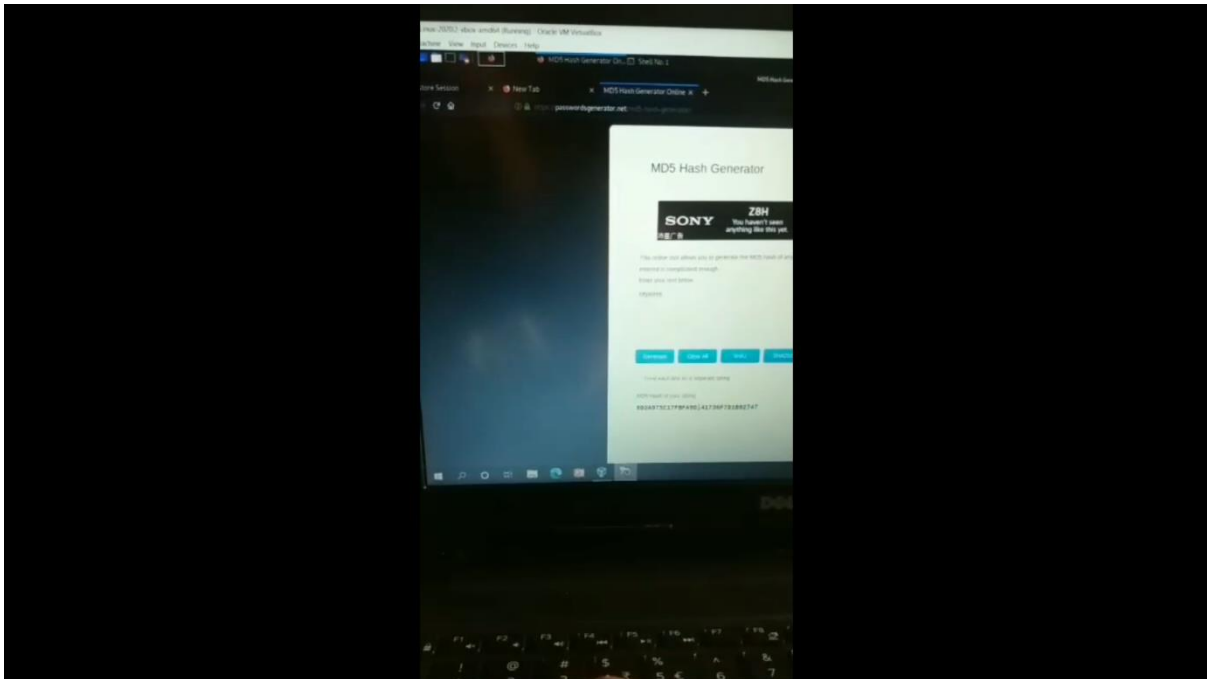
kali@kali:~$ cracking the ssh password hash..
bash: cracking: command not found
kali@kali:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/kali/.ssh/id_rsa):
/home/kali/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/kali/.ssh/id_rsa
Your public key has been saved in /home/kali/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:H09/+HzUoqrvfPyy7iVyjohQevrDEGHp7xR766r30BQ kali@kali
The key's randomart image is:
+---[RSA 3072]-----+
|
|  .
| +
| o . E
| o . .
| o.oS . . .
| .o+o.. + ...o
| o=o ...|o|.+.+o.
| +=o.o =o +.
| o++=+*+=+. +
+----[SHA256]-----+
kali@kali:~$

```

```
kali@kali: ~  
File Actions Edit View Help  
o . E  
o . .  
o.oS . . .  
.o+o.. + ... o  
o=o ... o.+.+o.  
+=o.o =o +.  
o++=+*=+. +  
+-----[SHA256]-----+  
kali@kali:~$ locate ssh2john  
/home/kali/ssh2john.py  
/home/kali/JohnTheRipper/run/ssh2john.py  
/home/kali/john-bleeding-jumbo/run/ssh2john.py  
/usr/share/john/ssh2john.py  
kali@kali:~$ /usr/share/john/ssh2john.py id_rsa > crack.txt  
kali@kali:~$ sudo john crack.txt --wordlist=rockyou.txt  
Using default input encoding: UTF-8  
Loaded 1 password hash (SSH [RSA/DSA/EC/OPENSSH (SSH private key  
s) 32/64])  
Cost 1 (KDF/cipher [0=MD5/AES 1=MD5/3DES 2=Bcrypt/AES]) is 2 for  
all loaded hashes  
Cost 2 (iteration count) is 16 for all loaded hashes  
Will run 2 OpenMP threads  
Note: This format may emit false positives, so it will keep tryi  
ng even after  
finding a possible candidate.  
Press 'q' or Ctrl-C to abort, almost any other key for status  
█
```

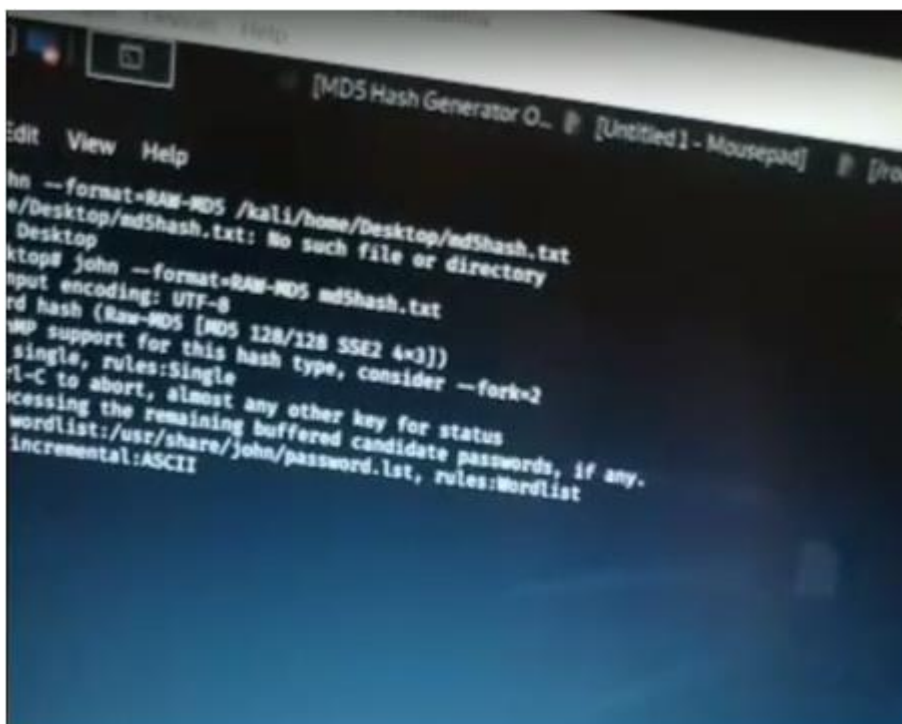
14)MD5

First create a MD5 hash code for the given word and then put the created md5 hash code in a file called md5hash.txt



To decrypt MD5 encryption and crack the password use the code as shown below

john --format=raw-md5 mdhash.txt



We can see that password is being cracked

