Practical's

Q1. Demonstrate the use of network tools:

1. Ping

```
PS C:\Users\thegr> ping 192.168.1.1

Pinging 192.168.1.1 with 32 bytes of data:
Reply from 192.168.1.1: bytes=32 time=3ms TTL=64
Reply from 192.168.1.1: bytes=32 time=3ms TTL=64
Reply from 192.168.1.1: bytes=32 time=1ms TTL=64
Reply from 192.168.1.1: bytes=32 time=3ms TTL=64

Ping statistics for 192.168.1.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 3ms, Average = 2ms
```

2. Ipconfig

```
PS C:\Users\thegr> ipconfig

Windows IP Configuration


Ethernet adapter VirtualBox Host-Only Network:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::dd82:edce:f25a:7ab2%21
   IPv4 Address. . . . . . . . . . . : 192.168.56.1
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :

Unknown adapter McAfee VPN:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Local Area Connection* 1:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Local Area Connection* 2:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::414a:8c87:37ef:5eb8%5
   IPv4 Address. . . . . . . . . . . : 192.168.1.11
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.1.1
```

3. tracert

```
PS C:\Users\thegr> tracert -4 192.168.1.1

Tracing route to 192.168.1.1 [192.168.1.1]
over a maximum of 30 hops:

  1     3 ms     4 ms     2 ms  192.168.1.1 [192.168.1.1]
```

4. arp

```
PS C:\Users\thegr> arp -a 192.168.1.1

Interface: 192.168.1.11 --- 0x5
  Internet Address      Physical Address      Type
  192.168.1.1           54-46-17-28-d3-72     dynamic
```

5. netstat

```
PS C:\Users\thegr> netstat -a

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:135            AdityaPC:0             LISTENING
  TCP    0.0.0.0:445            AdityaPC:0             LISTENING
  TCP    0.0.0.0:5040           AdityaPC:0             LISTENING
  TCP    0.0.0.0:6646           AdityaPC:0             LISTENING
  TCP    0.0.0.0:49664          AdityaPC:0             LISTENING
  TCP    0.0.0.0:49665          AdityaPC:0             LISTENING
  TCP    0.0.0.0:49666          AdityaPC:0             LISTENING
  TCP    0.0.0.0:49667          AdityaPC:0             LISTENING
  TCP    0.0.0.0:49668          AdityaPC:0             LISTENING
  TCP    0.0.0.0:49669          AdityaPC:0             LISTENING
  TCP    0.0.0.0:49670          AdityaPC:0             LISTENING
  TCP    0.0.0.0:49671          AdityaPC:0             LISTENING
  TCP    127.0.0.1:49698        AdityaPC:49699         ESTABLISHED
  TCP    127.0.0.1:49699        AdityaPC:49698         ESTABLISHED
  TCP    127.0.0.1:49703        AdityaPC:49704         ESTABLISHED
  TCP    127.0.0.1:49704        AdityaPC:49703         ESTABLISHED
  TCP    192.168.1.11:139       AdityaPC:0             LISTENING
  TCP    192.168.1.11:49444     20.198.162.76:https    ESTABLISHED
  TCP    192.168.1.11:54859     52.98.58.50:https      ESTABLISHED
  TCP    192.168.1.11:54860     52.98.57.162:https     ESTABLISHED
  TCP    192.168.1.11:54863     117.18.237.29:http     CLOSE_WAIT
  TCP    192.168.1.11:54891     117.18.232.200:https   CLOSE_WAIT
  TCP    192.168.1.11:54975     161.69.49.133:https    CLOSE_WAIT
```

6. whois

```
┌──(kali㉿kali)-[~]
└─$ whois facebook.com
   Domain Name: FACEBOOK.COM
   Registry Domain ID: 2320948_DOMAIN_COM-VRSN
   Registrar WHOIS Server: whois.registrarsafe.com
   Registrar URL: http://www.registrarsafe.com
   Updated Date: 2022-01-26T16:45:06Z
   Creation Date: 1997-03-29T05:00:00Z
   Registry Expiry Date: 2031-03-30T04:00:00Z
   Registrar: RegistrarSafe, LLC
   Registrar IANA ID: 3237
   Registrar Abuse Contact Email: abusecomplaints@registrarsafe.com
   Registrar Abuse Contact Phone: +1-650-308-7004
   Domain Status: clientDeleteProhibited https://icann.org/epp#clientDelet
eProhibited
   Domain Status: clientTransferProhibited https://icann.org/epp#clientTra
nsferProhibited
   Domain Status: clientUpdateProhibited https://icann.org/epp#clientUpdat
eProhibited
   Domain Status: serverDeleteProhibited https://icann.org/epp#serverDelet
eProhibited
   Domain Status: serverTransferProhibited https://icann.org/epp#serverTra
nsferProhibited
   Domain Status: serverUpdateProhibited https://icann.org/epp#serverUpdat
eProhibited
```

Q2. Use of password cracking tool:

John the Ripper

```
┌──(kali㉿kali)-[~/Desktop]
└─$ cd ..

┌──(kali㉿kali)-[~]
└─$ john /home/kali/Desktop/hash.txt
Using default input encoding: UTF-8
Loaded 1 password hash (Office, 2007/2010/2013 [SHA1 256/256 AVX2 8x / SHA
512 256/256 AVX2 4x AES])
Cost 1 (MS Office version) is 2013 for all loaded hashes
Cost 2 (iteration count) is 100000 for all loaded hashes
Will run 2 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:00:21 32.87% 1/3 (ETA: 18:01:34) 0g/s 66.16p/s 66.16c/s 66.16C/s r
docx..tdocx192.docx
0g 0:00:00:54 73.53% 1/3 (ETA: 18:01:44) 0g/s 65.18p/s 65.18c/s 65.18C/s d
ocx30..docx192.docx32
Almost done: Processing the remaining buffered candidate passwords, if any
.
Proceeding with wordlist:/usr/share/john/password.lst
12345            (192.docx)
1g 0:00:01:21 DONE 2/3 (2022-04-23 18:01) 0.01224g/s 65.10p/s 65.10c/s 65.
10C/s 123456..secret
Use the "--show" option to display all of the cracked passwords reliably
Session completed.

┌──(kali㉿kali)-[~]
└─$
```

Q3. Perform the encryption and decryption of:

1. Caesar Cipher

   Encryption code

```
#include <bits/stdc++.h>

using namespace std;

int main()
{
    char message[100], ch;
    int i, key;
    cout << "Enter a message to encrypt: ";
    cin.getline(message, 100);      //what is message and size of message
    cout << "Enter key: ";
    cin >> key;
    for (i = 0; message[i] != '\0'; ++i)
    {
        ch = message[i];
        if (ch >= 'a' && ch <= 'z')   //for small casing
        {
            ch = ch + key;
            if (ch > 'z')
            {
                ch = ch - 'z' + 'a' - 1;
            }
            message[i] = ch;
        }
        else if (ch >= 'A' && ch <= 'Z') // for capitall casing
        {
            ch = ch + key;
            if (ch > 'Z')
            {
                ch = ch - 'Z' + 'A' - 1;
            }
            message[i] = ch;
        }
    }
    cout << "Encrypted message: " << message;
    return 0;
}
```

Output

Enter a message to encrypt: hello world

Enter key: 4

Encrypted message: lipps asvph

Decryption code

```
#include<bits/stdc++.h>
```

```
using namespace std;

int main()
{
char message[100], ch;
int i, key;
cout << "Enter a message to decrypt: ";
cin.getline(message, 100);
cout << "Enter key: ";
cin >> key;
for(i = 0; message[i] != '\0'; ++i){
ch = message[i];
if(ch >= 'a' && ch <= 'z'){
ch = ch - key;
if(ch < 'a'){
ch = ch + 'z' - 'a' + 1;
}
message[i] = ch;
}
else if(ch >= 'A' && ch <= 'Z'){
ch = ch - key;
if(ch > 'a'){
ch = ch + 'Z' - 'A' + 1;
}
message[i] = ch;
}
}
cout << "Decrypted message: " << message;
return 0;
}
```

Output

Enter a message to decrypt: lipps asvph

Enter key: 4

Decrypted message: hello world

2. Rail Fence Cipher

Code

```
// Rail Fence Cipher
// Encryption and Decryption
#include <bits/stdc++.h>
using namespace std;
```

```
// function to encrypt a message
string encryptRailFence(string text, int key)
{
    // create the matrix to cipher plain text
    // key = rows , length(text) = columns
    char rail[key][(text.length())];

    // filling the rail matrix to distinguish filled
    // spaces from blank ones
    for (int i=0; i < key; i++)
        for (int j = 0; j < text.length(); j++)
            rail[i][j] = '\n';

    // to find the direction
    bool dir_down = false;
    int row = 0, col = 0;

    for (int i=0; i < text.length(); i++)
    {
        // check the direction of flow
        // reverse the direction if we've just
        // filled the top or bottom rail
        if (row == 0 || row == key-1)
            dir_down = !dir_down;

        // fill the corresponding alphabet
        rail[row][col++] = text[i];

        // find the next row using direction flag
        dir_down?row++ : row--;
    }

    //now we can construct the cipher using the rail matrix
    string result;
    for (int i=0; i < key; i++)
        for (int j=0; j < text.length(); j++)
            if (rail[i][j]!='\n')
                result.push_back(rail[i][j]);

    return result;
}

// This function receives cipher-text and key
// and returns the original text after decryption
string decryptRailFence(string cipher, int key)
{
    // create the matrix to cipher plain text
```

```
    // key = rows , length(text) = columns
    char rail[key][cipher.length()];

    // filling the rail matrix to distinguish filled
    // spaces from blank ones
    for (int i=0; i < key; i++)
        for (int j=0; j < cipher.length(); j++)
            rail[i][j] = '\n';

    // to find the direction
    bool dir_down;

    int row = 0, col = 0;

    // mark the places with '*'
    for (int i=0; i < cipher.length(); i++)
    {
        // check the direction of flow
        if (row == 0)
            dir_down = true;
        if (row == key-1)
            dir_down = false;

        // place the marker
        rail[row][col++] = '*';

        // find the next row using direction flag
        dir_down?row++ : row--;
    }

    // now we can construct the fill the rail matrix
    int index = 0;
    for (int i=0; i<key; i++)
        for (int j=0; j<cipher.length(); j++)
            if (rail[i][j] == '*' && index<cipher.length())
                rail[i][j] = cipher[index++];


    // now read the matrix in zig-zag manner to construct
    // the resultant text
    string result;

    row = 0, col = 0;
    for (int i=0; i< cipher.length(); i++)
    {
        // check the direction of flow
        if (row == 0)
            dir_down = true;
```

```
        if (row == key-1)
            dir_down = false;

        // place the marker
        if (rail[row][col] != '*')
            result.push_back(rail[row][col++]);

        // find the next row using direction flag
        dir_down?row++: row--;
    }
    return result;
}

//driver program to check the above functions
int main()
{
    cout << "----------Encryption of the following text-----------"<<endl;
    cout << "attack at once :-  " << encryptRailFence("attack at once", 2) <<
endl;
    cout << "GeeksforGeeks :-  "<< encryptRailFence("GeeksforGeeks ", 3) <<
endl;
    cout << "defend the east wall :-  " << encryptRailFence("defend the east
wall", 3) << endl;

    cout <<"-------now decryption of the same txt-----------" <<endl;
    //Now decryption of the same cipher-text
    cout << decryptRailFence("GsGsekfrek eoe",3) << endl;
    cout << decryptRailFence("atc toctaka ne",2) << endl;
    cout << decryptRailFence("dnhaweedtees alf  tl",3) << endl;

    return 0;
}
```

Output

```
----------Encryption of the following text-----------
attack at once :-  atc toctaka ne
GeeksforGeeks :-  GsGsekfrek eoe
defend the east wall :-  dnhaweedtees alf  tl
-------now decryption of the same txt-----------
GeeksforGeeks
attack at once
defend the east wall
```

Q4. Use nmap to analyse remote machine

Nmap flags detail

-sS = full TCP scan /stealth scan

-O = to identify operating system

-Pn = only for port scaning

-sX = only use against linux system

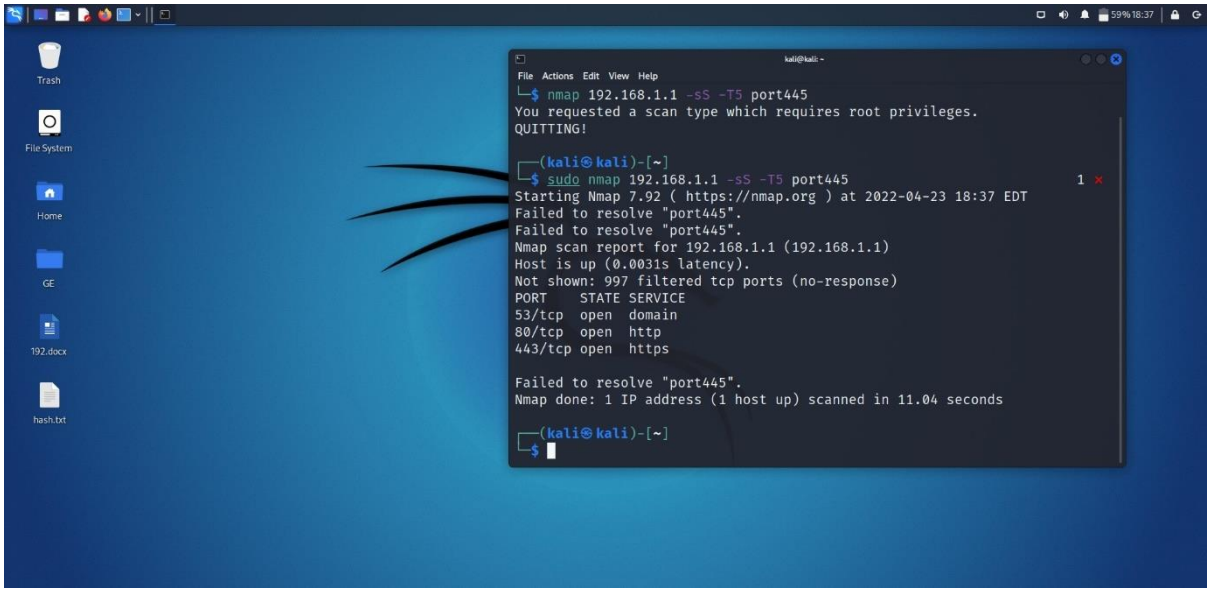
-A = acknowledgment flag

-2 = UDP scan

-1 =ICMP ping scan

-S = sync scan

-T5 = to speed up



Q6. Demonstrate sending a protected word document.

Follow the steps below to apply a password to a document:

1. Click the **File** tab.
2. Click **Info**.
3. Click **Protect Document**, and then click **Encrypt with Password**.
4. In the **Encrypt Document** box, type a password, and then click **OK**.
5. In the **Confirm Password** box, type the password again, and then click **OK**.
6. Now send it using any platform .

Q7. Demonstrate sending a digitally signed document.

1. Open your document and click the **File** tab.
2. Click **Info** and then click **Protect Document**.
3. From the Protect Document drop-down menu, click Add a Digital Signature.
4. Select a Commitment Type, such as created and approved this document, and then click Sign
5. Insert your PIV card into the card reader. Enter your smart card (PIV) PIN and click **OK**.
6. *The **Signature Confirmation** box tells you that Word saved your digital signature.* Click **OK**.

Q8. Demonstrate sending a protected worksheet.

1. Click **Review** > **Protect Workbook**.
2. Enter a password in the **Password** box.
3. Select OK, re-enter the password to confirm it, and then select OK again.

Q9. Demonstrate the use of steganography tools.

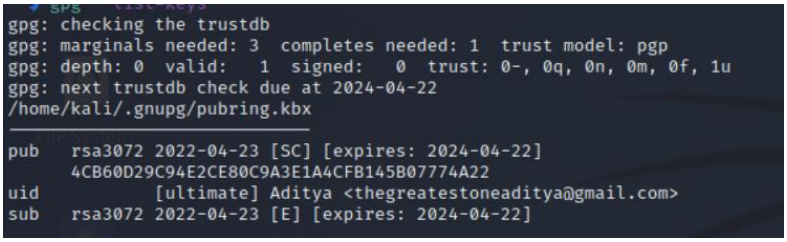
Here we will use “STEGHIDE” TOOL in kali linux and we will hide our secret tect inside image.

1. Download steghide .
2. Then create your secret .txt file and select the image you want to be embedded.
3. Then use the command “steghide embed -cf ‘image name ‘ -ef ‘txt file name’ “
4. Then give it the password and your file is ready to send message secretly .
5. To extract use the command “steghide extract -sf ‘file name’ ”.

Q10. Demonstrate the gpg utility for signing and encrypting purpose.

Steps

1. Installation of gpg :- sudo apt-get install gnupg
2. Generation of Key :- gpg –gen-key
3. Export the PUBLIC KEY :- gpg --export -a “user name” > public.key
4. Export the PRIVATE KEY :- gpg –export-secret-key -a “user name”>private.key
5. Import a public key :- gpg –import public.key
6. Import a private key :- gpg –allow-secret.key-import –import private.key
7. List the keys in your public key ring :- gpg –list-keys
8. List the keys in your secret key ring :- gpg –list-secret-keys
9. Encrypt the and decrypt the file



```
gpg: checking the trustdb
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: depth: 0  valid:   1  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 1u
gpg: next trustdb check due at 2024-04-22
/home/kali/.gnupg/pubring.kbx
---------------------------
pub   rsa3072 2022-04-23 [SC] [expires: 2024-04-22]
      4CB60D29C94E2CE80C9A3E1A4CFB145B07774A22
uid           [ultimate] Aditya <thegreatestoneaditya@gmail.com>
sub   rsa3072 2022-04-23 [E] [expires: 2024-04-22]
```