1. Perform an Experiment for port scanning with nmap.

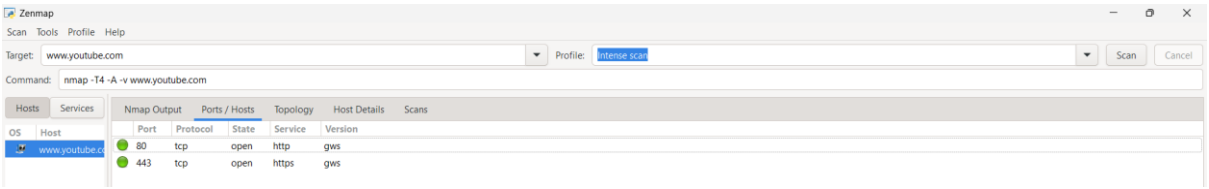
Nmap Output Tab:

The “Nmap Output” tab is displayed by default when a scan is run. It shows the familiar Nmap terminal output. The display highlights parts of the output according to their meaning; for example, open and closed ports are displayed in different colors.

```
Starting Nmap 7.95 ( https://nmap.org ) at 2024-04-28 12:46 India Standard Time
NSE: Loaded 157 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 12:46
Completed NSE at 12:46, 0.00s elapsed
Initiating NSE at 12:46
Completed NSE at 12:46, 0.00s elapsed
Initiating NSE at 12:46
Completed NSE at 12:46, 0.00s elapsed
Initiating Ping Scan at 12:46
Scanning www.youtube.com (142.250.196.174) [4 ports]
Completed Ping Scan at 12:46, 0.10s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 12:46
Completed Parallel DNS resolution of 1 host. at 12:46, 0.02s elapsed
Initiating SYN Stealth Scan at 12:46
Scanning www.youtube.com (142.250.196.174) [1000 ports]
Discovered open port 443/tcp on 142.250.196.174
Discovered open port 80/tcp on 142.250.196.174
Completed SYN Stealth Scan at 12:46, 5.21s elapsed (1000 total ports)
Initiating Service scan at 12:46
Scanning 2 services on www.youtube.com (142.250.196.174)
Service scan Timing: About 50.00% done; ETC: 12:48 (0:01:00 remaining)
Completed Service scan at 12:47, 68.73s elapsed (2 services on 1 host)
Initiating OS detection (try #1) against www.youtube.com (142.250.196.174)
Retrying OS detection (try #2) against www.youtube.com (142.250.196.174)
Initiating Traceroute at 12:47
Completed Traceroute at 12:47, 3.04s elapsed
Initiating Parallel DNS resolution of 12 hosts. at 12:47
Completed Parallel DNS resolution of 12 hosts. at 12:48, 13.47s elapsed
NSE: Script scanning 142.250.196.174.
Initiating NSE at 12:48
Completed NSE at 12:48, 7.29s elapsed
Initiating NSE at 12:48
Completed NSE at 12:48, 1.55s elapsed
Initiating NSE at 12:48
Completed NSE at 12:48, 0.01s elapsed
Nmap scan report for www.youtube.com (142.250.196.174)
Host is up (0.029s latency).
Other addresses for www.youtube.com (not scanned): 2404:6800:4007:827::200e 2404:68
172.217.166.110 142.250.77.142 142.250.77.174 142.250.195.46 142.250.195.78 142.250
142.250.196.78
rDNS record for 142.250.196.174: maa03s47-in-f14.1e100.net
Not shown: 998 filtered tcp ports (no-response)
```

Ports/Hosts Tab:

The “Ports / Hosts” tab's display differs depending on whether a host or a service is currently selected. When a host is selected, it shows all the interesting ports on that host, along with version information when available.

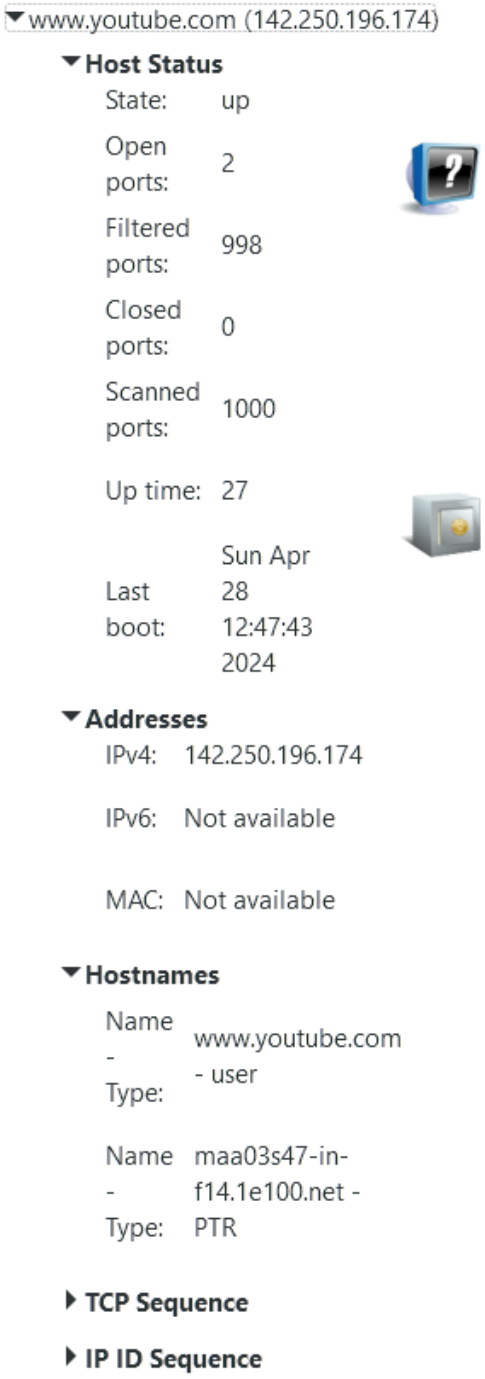


Topology Tab:

The “Topology” tab is an interactive view of the connections between hosts in a network. Hosts are arranged in concentric rings. Each ring represents an additional network hop from the center node. Clicking on a node brings it to the center. Because it shows a representation of the network paths between hosts, the “Topology” tab benefits from the use of the `--traceroute` option.

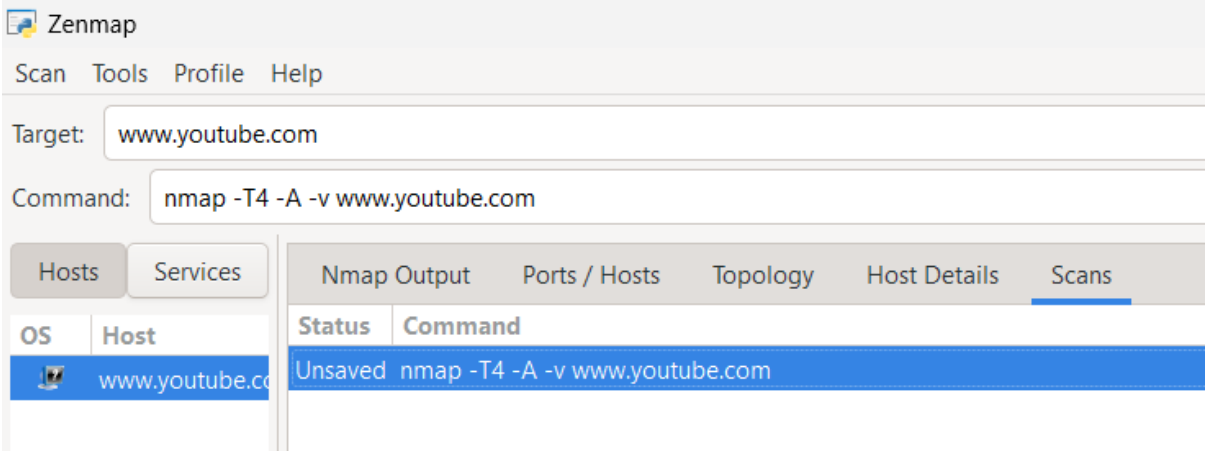
Host Details Tab:

The “Host Details” tab breaks all the information about a single host into a hierarchical display. Shown are the host's names and addresses, its state (up or down), and the number and status of scanned ports. The host's uptime, operating system, OS icon and other associated details are shown when available. When no exact OS match is found, the closest matches are displayed. There is also a collapsible text field for storing a comment about the host which will be saved when the scan is saved to a file.



Scans Tab:

The “Scans” tab shows all the scans that are aggregated to make up the network inventory. From this tab you can add scans (from a file or directory) and remove scans. While a scan is executing and not yet complete, its status is “Running”. You may cancel a running scan by clicking the “Cancel Scan” button.



2. Setup a Honeypot and Monitor the Honeypot on your Network.

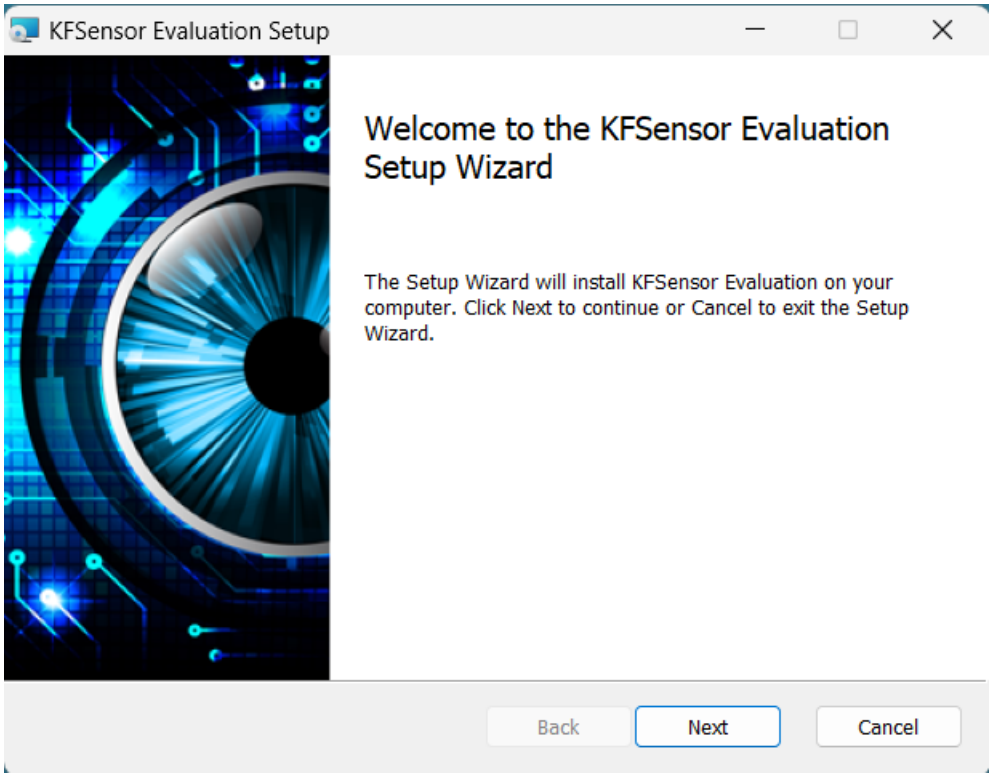
Pre-requisites:

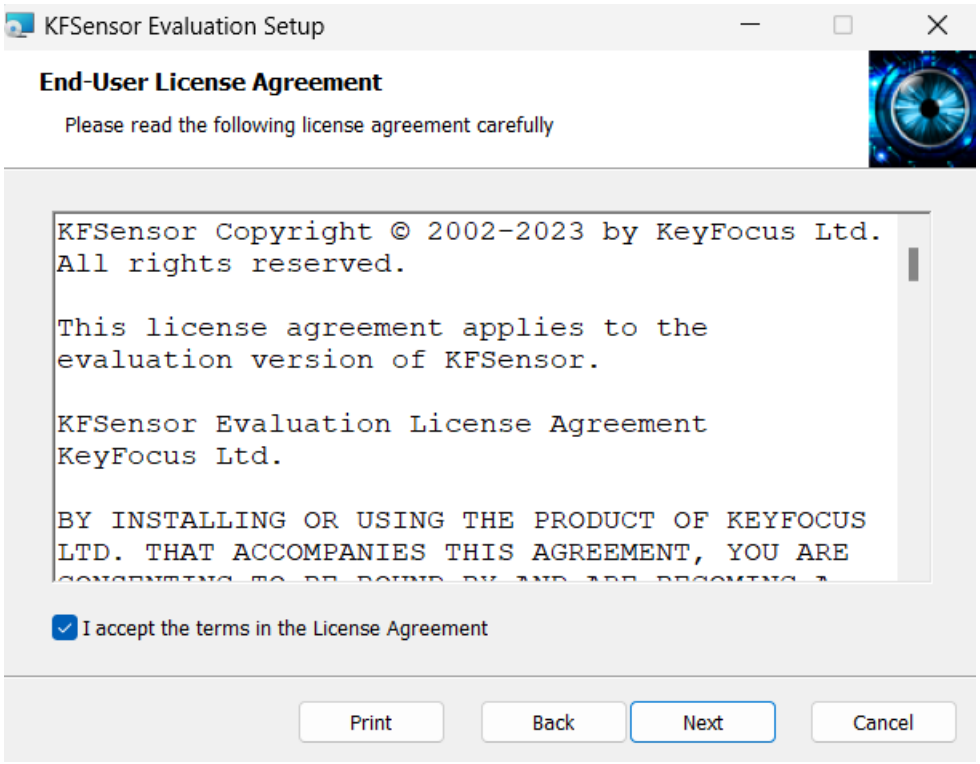
- **KFSensor Tool:**

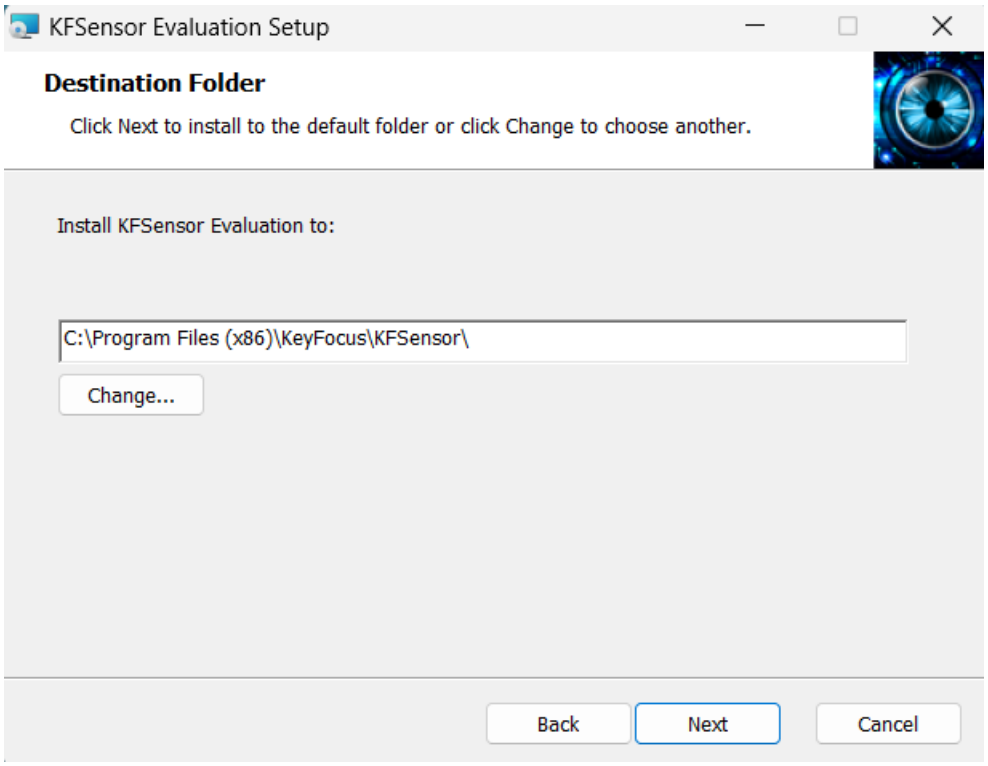
KF Sensor is a Windows-based honeypot system designed to simulate various services and detect malicious activities. It provides an easy-to-use interface for configuring and monitoring honeypot services, making it suitable for both novice and experienced users.

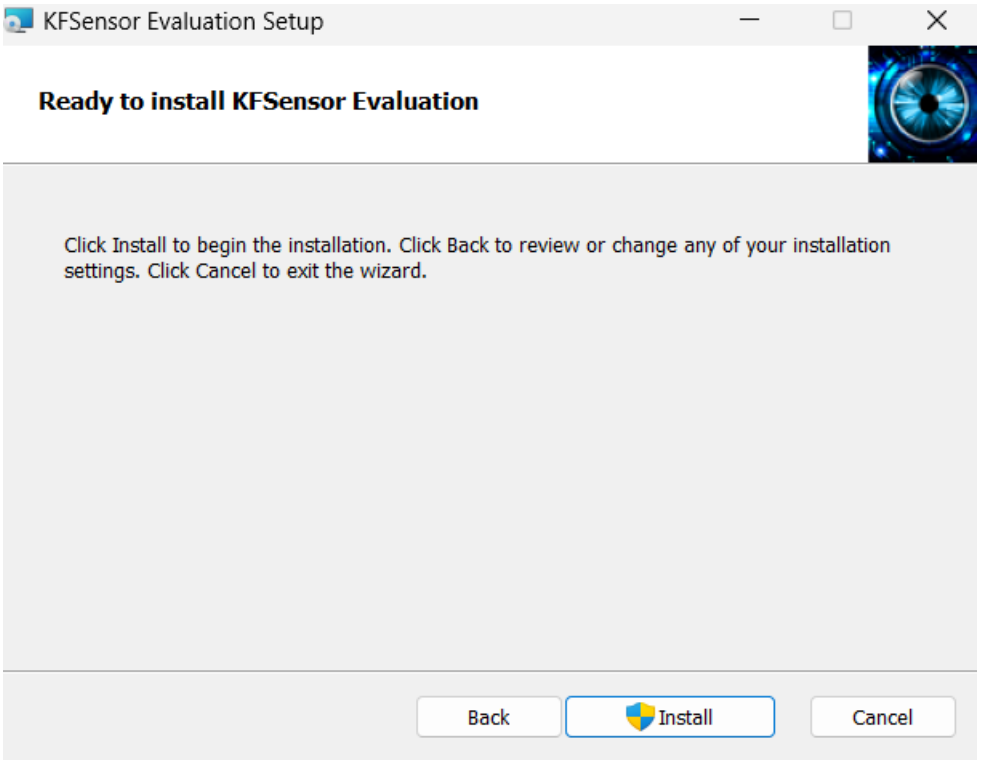
Installation:

1. Open the Browser and navigate to the following link: https://www.kfsensor.net/kfsensor/free-trial/
2. Download the .msi file for the KFSensor Professional for free trial.
3. Follow the given sequence of steps for the installation;

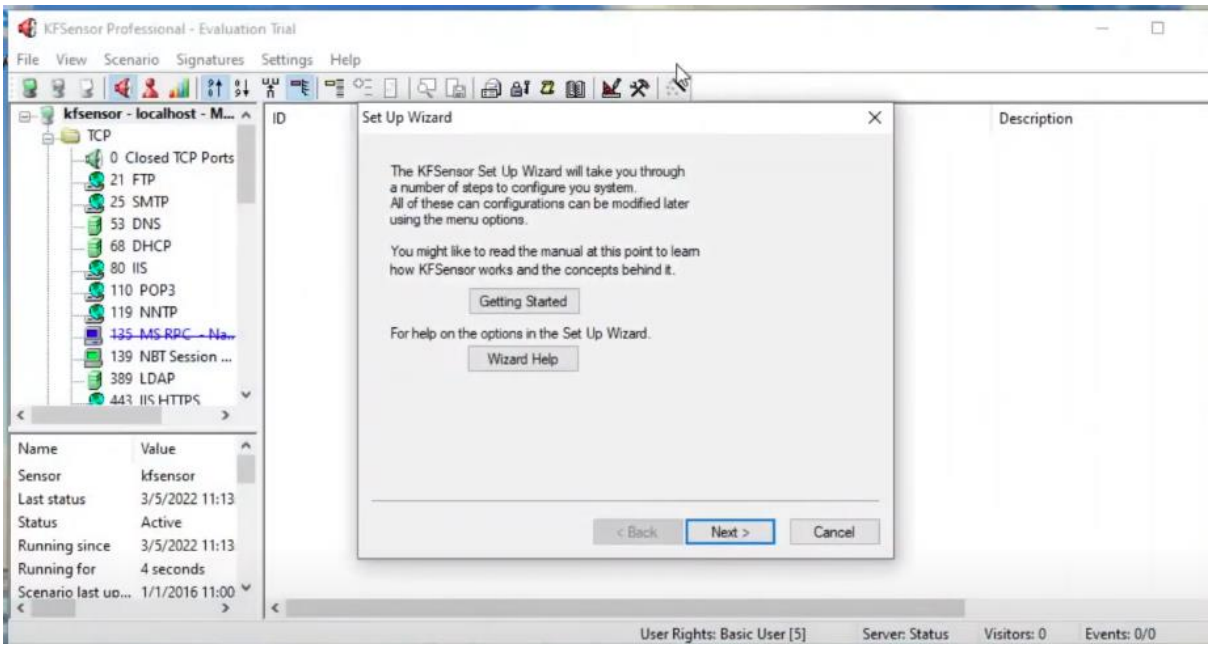


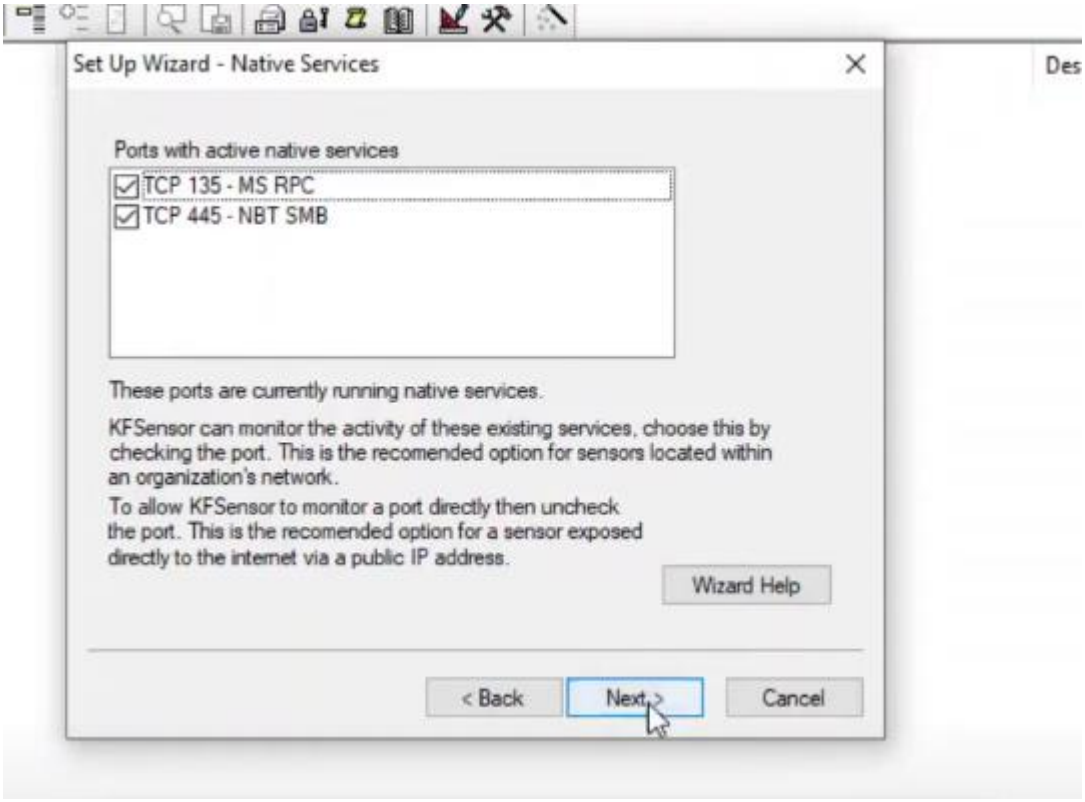

KFSensor Evaluation Setup
End-User License Agreement
Please read the following license agreement carefully
KFSensor Copyright © 2002-2023 by KeyFocus Ltd.
All rights reserved.
This license agreement applies to the
evaluation version of KFSensor.
KFSensor Evaluation License Agreement
KeyFocus Ltd.
BY INSTALLING OR USING THE PRODUCT OF KEYFOCUS
LTD. THAT ACCOMPANIES THIS AGREEMENT, YOU ARE
I accept the terms in the License Agreement
Print
Back
Next
Cancel

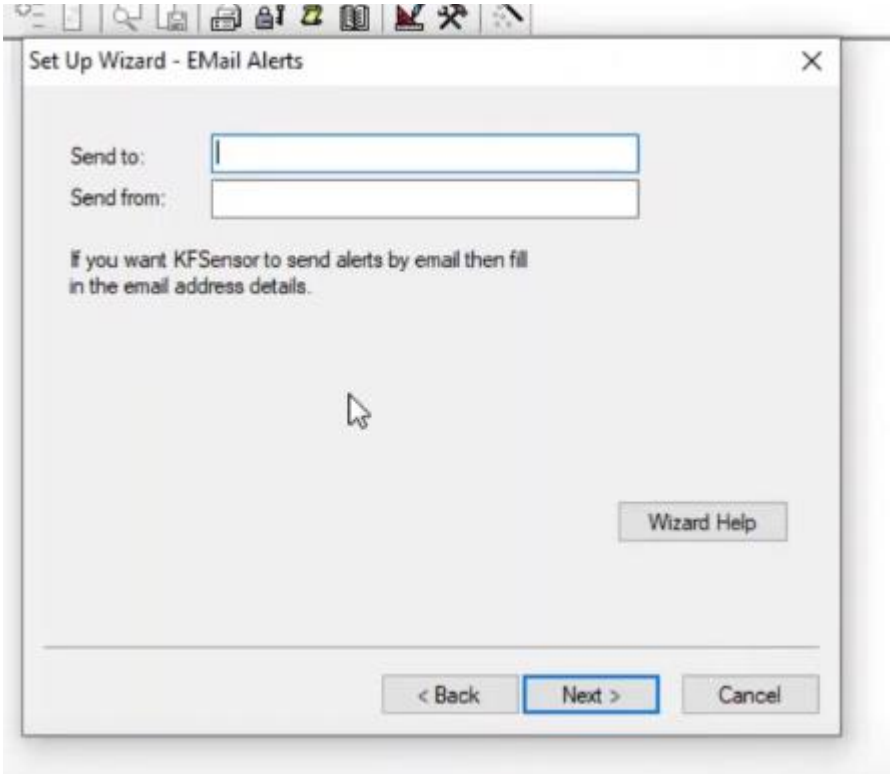

KFSensor Evaluation Setup
Destination Folder
Click Next to install to the default folder or click Change to choose another.
Install KFSensor Evaluation to:
C:\Program Files (x86)\KeyFocus\KFSensor\
Change...
Back
Next
Cancel



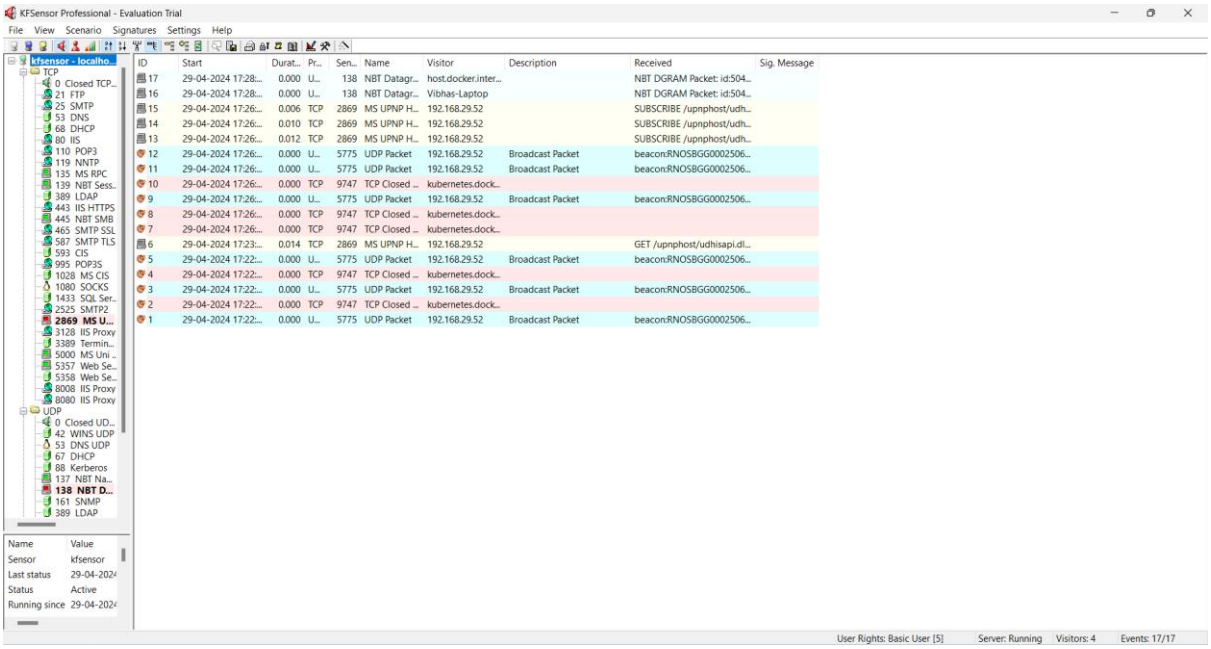
4. Click on Install and complete the Installation to launch KFSensor Tool.
5. Proceed to the corresponding steps for the setup wizard as follows;







6. Click the Finish Button to finish the setup of the KFSensor Tool.
7. The Events interface displays the list of the open hosts in the following format;



8. Similar Implementation can be shown through the Terminal in Windows or Linux as follows;

```
C:\Users\Asus>ipconfig

Windows IP Configuration


Unknown adapter Local Area Connection:

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

Wireless LAN adapter Local Area Connection* 3:

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

Wireless LAN adapter Local Area Connection* 4:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::6b46:79f8:e595:524f%15
   IPv4 Address. . . . . . . . . . . : 192.168.137.1
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :

Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . :
   IPv6 Address. . . . . . . . . . . : 2405:201:c028:811c:6bf1:260a:33ff:5b91
   Temporary IPv6 Address. . . . . . : 2405:201:c028:811c:2cfe:d205:5f8f:5855
   Link-local IPv6 Address . . . . . : fe80::5afd:51f5:af9a:583d%17
   IPv4 Address. . . . . . . . . . . : 192.168.29.12
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : fe80::aada:cff:fe23:7158%17
                                       192.168.29.1

Ethernet adapter Bluetooth Network Connection:

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

```
C:\Users\Asus>nmap 192.168.137.1
Starting Nmap 7.95 ( https://nmap.org ) at 2024-04-29 17:46 India Standard Time
Nmap scan report for 192.168.137.1
Host is up (0.00039s latency).
Not shown: 896 closed tcp ports (reset)
PORT      STATE SERVICE
1/tcp     open  tcpmux
7/tcp     open  echo
9/tcp     open  discard
13/tcp    open  daytime
17/tcp    open  qotd
19/tcp    open  chargen
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
42/tcp    open  nameserver
53/tcp    open  domain
80/tcp    open  http
81/tcp    open  hosts2-ns
82/tcp    open  xfer
83/tcp    open  mit-ml-dev
110/tcp   open  pop3
111/tcp   open  rpcbind
113/tcp   open  ident
119/tcp   open  nntp
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
143/tcp   open  imap
443/tcp   open  https
445/tcp   open  microsoft-ds
465/tcp   open  smtps
548/tcp   open  afp
587/tcp   open  submission
593/tcp   open  http-rpc-epmap
636/tcp   open  ldapssl
993/tcp   open  imaps
995/tcp   open  pop3s
1028/tcp  open  unknown
1080/tcp  open  socks
```

3. Install a jcrpt tool(or any other equivalent) and demonstrate Asymmetric ,Symmetric crypto algorithm ,Hash and Digital/PKI signatures studied in theory Cryptography and Network Security.

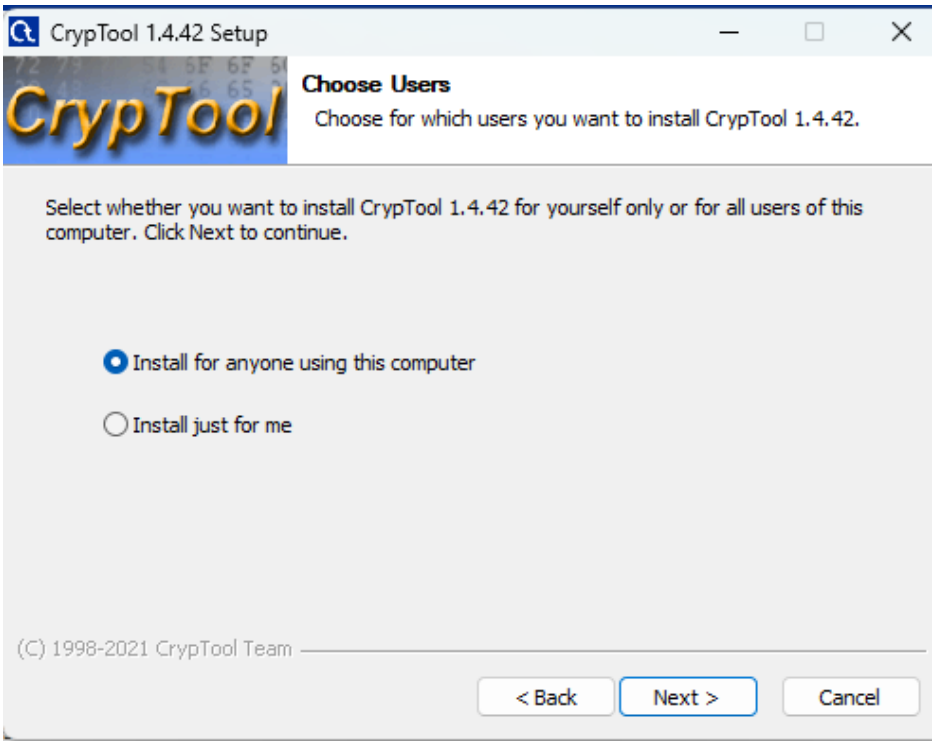
Pre-requisites:

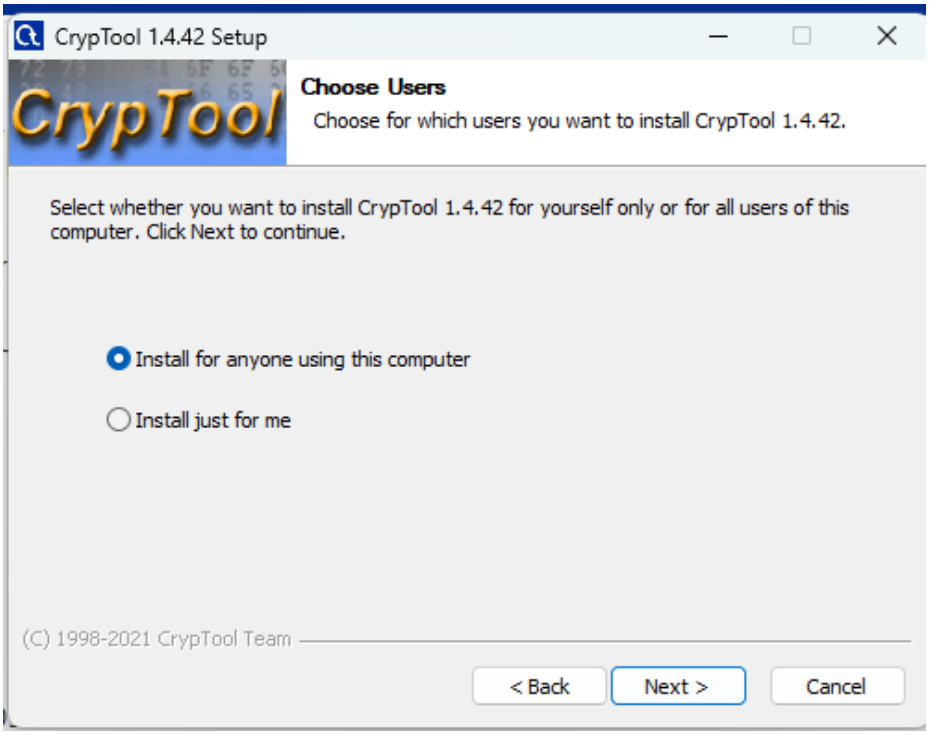
- **CrypTool**

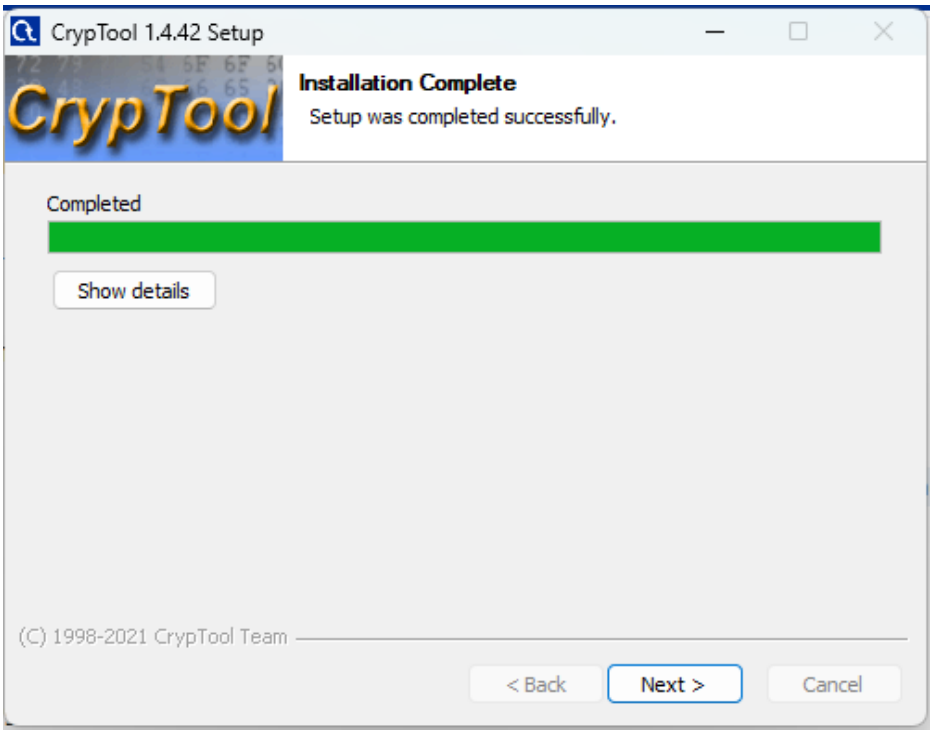
Installation:

1. Open the desired Website and navigate to the URL: https://www.cryptool.org/en/ct1/downloads/
2. Download the respective version from the Website.
3. Perform the installation of the given CrypTool as follows:

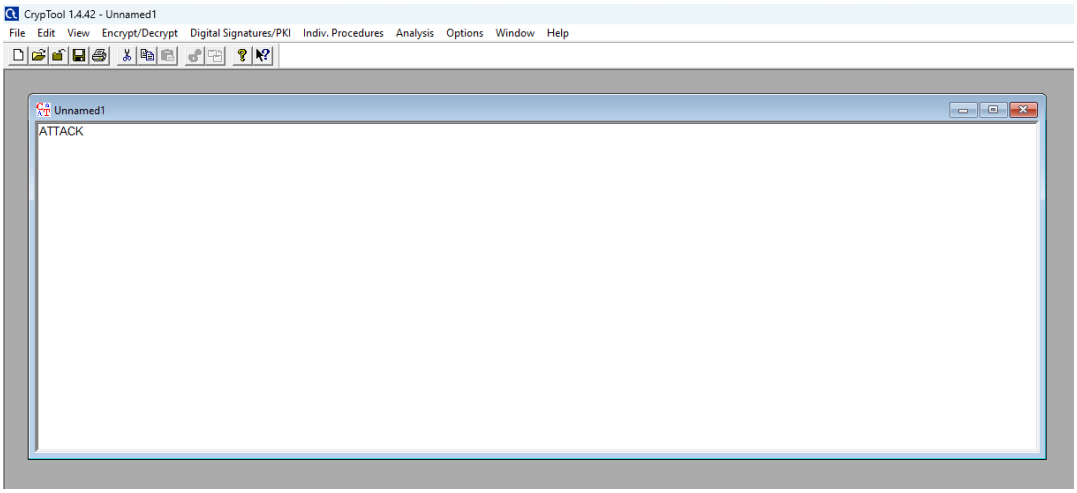

CrypTool 1.4.42 Setup
CrypTool
Welcome to CrypTool 1.4.42 Setup
Setup will guide you through the installation of CrypTool 1.4.42.
It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.
Click Next to continue.
Next >
Cancel


CrypTool 1.4.42 Setup
CrypTool
Choose Users
Choose for which users you want to install CrypTool 1.4.42.
Select whether you want to install CrypTool 1.4.42 for yourself only or for all users of this computer. Click Next to continue.
Install for anyone using this computer
Install just for me
(C) 1998-2021 CrypTool Team
< Back
Next >
Cancel

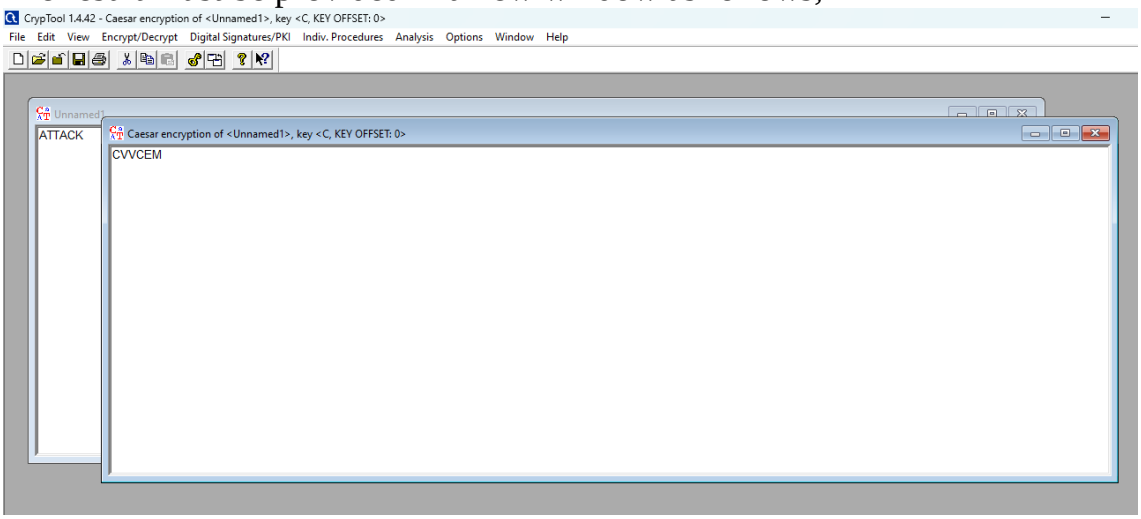




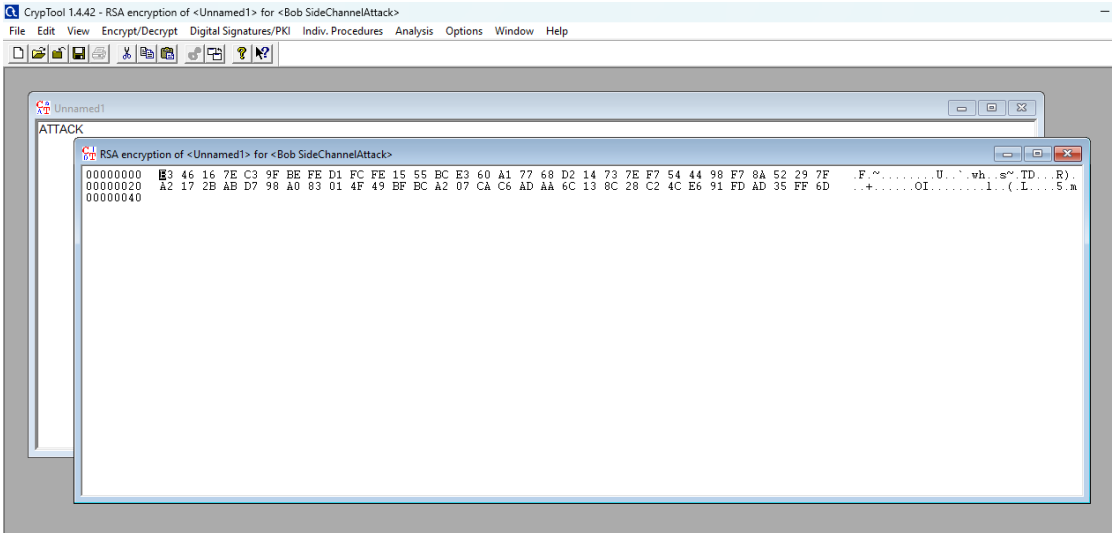
4. Upon launching of Cryptool, create a new window from the File menu for performing the cryptographic algorithms.
5. Type the Desired Text in the Window.
6. The final result must be as follows;



7. For Symmetric algorithms, say Caesar Cipher, click on the Encrypt/Decrypt menu.
8. Select the Symmetric option and choose the Caesar Cipher option from the list.
9. Provide the necessary details for the given Cryptographic algorithm. Finally click on either Encrypt or Decrypt.
10. The result must be provided in a new window as follows;



11. For Asymmetric algorithms, say RSA, click on the Encrypt/Decrypt menu.
12. Select the Asymmetric option and choose the RSA Encryption/Decryption option from the list.
13. Select the necessary Digital Signature provided with the necessary PIN.
14. Provide the necessary details for the given Cryptographic algorithm. Finally click on either Encrypt or Decrypt.
15. The result must be provided in a new window as follows;


CrypTool 1.4.42 - RSA encryption of <Unnamed1> for <Bob SideChannelAttack>
File Edit View Encrypt/Decrypt Digital Signatures/PKI Indiv. Procedures Analysis Options Window Help
Unnamed1
ATTACK
RSA encryption of <Unnamed1> for <Bob SideChannelAttack>
00000000 E3 46 16 7E C3 9F BE FE D1 FC FE 15 55 BC E3 60 A1 77 68 D2 14 73 7E F7 54 44 98 F7 8A 52 29 7F
00000020 A2 17 2B AB D7 98 A0 83 01 4F 49 BF BC A2 07 CA C6 AD AA 6C 13 8C 28 C2 4C E6 91 FD AD 35 FF 6D
00000040

4. Write a program to perform encryption and decryption using the following substitution ciphers.

A. Caesar Cipher:

```
#include <iostream>
#include <string>
using namespace std;

string encrypt(const string& msg, int key) {
   string encryptedMsg = msg;
   for (size_t i = 0; i < encryptedMsg.length(); ++i) {
      char ch = encryptedMsg[i];
      if (isalpha(ch)) {
         char base = islower(ch) ? 'a' : 'A';
         ch = base + (ch - base + key) % 26;
         encryptedMsg[i] = ch;
      }
   }
   return encryptedMsg;
}

string decrypt(const string& msg, int key) {
   string decryptedMsg = msg;
   for (size_t i = 0; i < decryptedMsg.length(); ++i) {
      char ch = decryptedMsg[i];
      if (isalpha(ch)) {
         char base = islower(ch) ? 'a' : 'A';
         ch = base + (ch - base - key + 26) % 26;
         decryptedMsg[i] = ch;
      }
   }
   return decryptedMsg;
}

int main() {
   string msg;
   int key;
```

```
    cout << "Enter the message: ";
    getline(cin, msg);
    cout << "Enter the key (shift value): ";
    cin >> key;
    string encryptedMsg = encrypt(msg, key);
    cout << "Encrypted message: " << encryptedMsg << endl;
    string decryptedMsg = decrypt(encryptedMsg, key);
    cout << "Decrypted message: " << decryptedMsg << endl;
    return 0;
}
```

Output:

```
Enter the message: Hello
Enter the key (shift value): 3
Encrypted message: Khoor
Decrypted message: Hello
```

B. PlayFair Cipher:

```
#include <iostream>
#include <string>
#include <vector>
using namespace std;

void gen_mat(string key,char mat[5][5]){
    string alpha="ABCDEFGHIKLMNOPQRSTUVWXYZ";
    vector<bool> used(26,false);
    int x=0,y=0;
    for(int i=0;i<key.size();i++){
        if(!used[key[i]-'A']){
            mat[x][y]=key[i];
            used[key[i]-'A']=true;
            if(++y==5){
                y=0;
                x++;
            }
```

```
        }
    }
    for(int i=0;i<alpha.size();i++){
        if(!used[alpha[i]-'A']){
            mat[x][y]=alpha[i];
            used[alpha[i]-'A']=true;
            if(++y==5){
                y=0;
                x++;
            }
        }
    }
}

string PlayFair(string pt,char key_mat[5][5],string E){
    string ct="";
    for(int i=0;i<pt.size();i+=2){
        char a=pt[i], b=(i+1<pt.size())?pt[i+1]:'X';
        if(a==b){
            b='X';
        }
        int ax,ay,bx,by;
        for(int x=0;x<5;x++){
            for(int y=0;y<5;y++){
                if(key_mat[x][y]==a){
                    ax=x;
                    ay=y;
                }
                if(key_mat[x][y]==b){
                    bx=x;
                    by=y;
                }
            }
        }
        if(ax==bx){
            ct+=(E=="encrypt")?key_mat[ax][(ay+1)%5]:key_mat[ax][((ay-1)+5)%5];
            ct+=(E=="encrypt")?key_mat[bx][(by+1)%5]:key_mat[bx][((by-1)+5)%5];
        }
        else if(ay==by){
```

```
            ct+=(E=="encrypt")?key_mat[(ax+1)%5][ay]:key_mat[((ax-1)+5)%5][ay];
            ct+=(E=="encrypt")?key_mat[(bx+1)%5][by]:key_mat[((bx-1)+5)%5][by];
        }
        else{
            ct+=key_mat[ax][by];
            ct+=key_mat[bx][ay];
        }
    }
    return ct;
}


int main(){
    char key_mat[5][5];
    string key,pt;
    cout<<"Key: ";
    getline(cin,key);
    cout<<"Plain Text: ";
    getline(cin,pt);
    gen_mat(key,key_mat);
    pt=PlayFair(pt,key_mat,"encrypt");
    cout<<"Encrypted Text: "<<pt<<endl;
    pt=PlayFair(pt,key_mat,"decrypt");
    if(pt.back()=='X'){
        pt.pop_back();
    }
    cout<<"Decrypted Text: "<<pt<<endl;
    return 0;
}
```

Output:

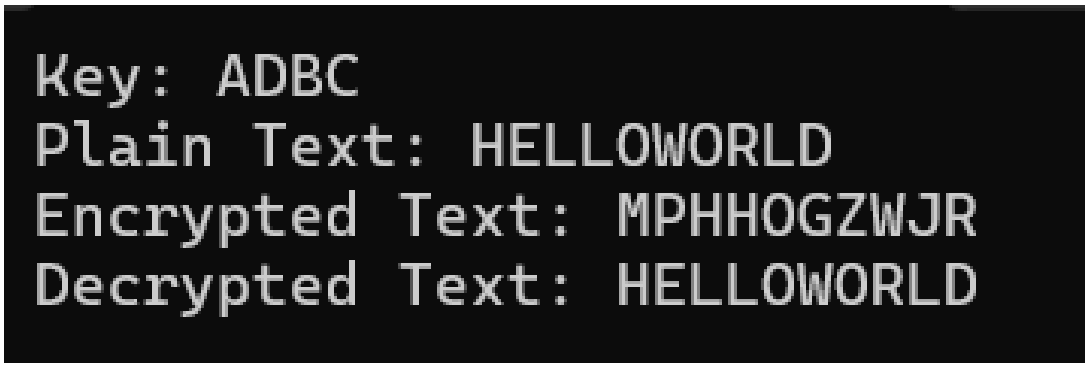
```
Key: HELLO
Plain Text: GOODMORNING
Encrypted Text: FALFNLTKKPDZ
Decrypted Text: GOODMORNING
```

C. Hill Cipher:

```
#include <iostream>
#include <string>
using namespace std;

int modInverse(int a, int m) {
   a = a % m;
   for (int x = 1; x < m; x++) {
      if ((a * x) % m == 1)
         return x;
   }
   return -1;
}

void multiply(int key_mat[2][2], int mat[2][1], string &res, const string
&alpha) {
   int mul[2][1];
   for(int i = 0; i < 2; i++) {
      mul[i][0] = 0;
      for(int k = 0; k < 2; k++)
         mul[i][0] += key_mat[i][k] * mat[k][0];
      mul[i][0] %= 26;
      if (mul[i][0] < 0)
         mul[i][0] += 26;
      res += alpha[mul[i][0]];
   }
}

string Hill(string pt, string key, bool decrypt = false) {
   string alpha = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
   int key_mat[2][2], x = 0;
   for (int i = 0; i < 2; i++) {
      for (int j = 0; j < 2; j++) {
         key_mat[i][j] = alpha.find(key[x++]);
      }
   }
   if(decrypt) {
      int det = (key_mat[0][0] * key_mat[1][1] - key_mat[0][1] *
key_mat[1][0]) % 26;
      if (det < 0) det += 26;
      int det_inv = modInverse(det, 26);
```

```
        if (det_inv == -1) {
            cout << "No modular inverse exists for the given key." << endl;
            return "";
        }
        int adj_mat[2][2] = {key_mat[1][1], -key_mat[0][1], -key_mat[1][0],
key_mat[0][0]};
        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 2; j++) {
                key_mat[i][j] = adj_mat[i][j] * det_inv % 26;
                if (key_mat[i][j] < 0)
                    key_mat[i][j] += 26;
            }
        }
    }
    string result;
    for (int i = 0; i < pt.size(); i += 2) {
        int mat[2][1];
        mat[0][0] = alpha.find(pt[i]);
        mat[1][0] = (i + 1 < pt.size()) ? alpha.find(pt[i + 1]) : alpha.find('X');
        multiply(key_mat, mat, result, alpha);
    }
    return result;
}

int main() {
    string key, pt;
    cout << "Key: ";
    getline(cin, key);
    cout << "Plain Text: ";
    getline(cin, pt);
    while (pt.size() % 2 != 0)
        pt += 'X';
    string encrypted_text = Hill(pt, key);
    cout << "Encrypted Text: " << encrypted_text << endl;
    string decrypted_text = Hill(encrypted_text, key, true);
    cout << "Decrypted Text: " << decrypted_text << endl;
    return 0;
}
```

Output:


Key: ADBC
Plain Text: HELLOWORLD
Encrypted Text: MPHHOGZWJR
Decrypted Text: HELLOWORLD

5. Write a program to implement the DES algorithm.

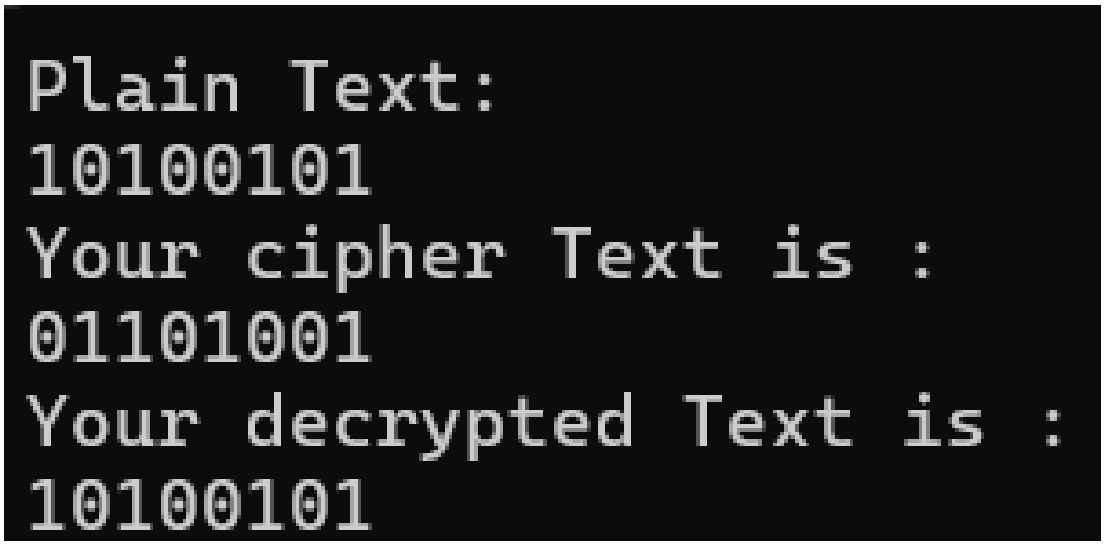
```
#include <iostream>
#include <string>
using namespace std;

class DES {
   int key[10] = {0,0,1,0,0,1,0,1,1,1};
   int P10[10] = {3,5,2,7,4,10,1,9,8,6};
   int P8[8] = {6,3,7,4,8,5,10,9};
   int key1[8];
   int key2[8];
   int IP[8] = {2,6,3,1,4,8,5,7};
   int EP[8] = {4,1,2,3,2,3,4,1};
   int P4[4] = {2,4,3,1};
   int IP_inv[8] = {4,1,3,5,7,2,8,6};
   int S0[4][4] = {{1,0,3,2},{3,2,1,0},{0,2,1,3},{3,1,3,2}};
   int S1[4][4] = {{0,1,2,3},{2,0,1,3},{3,0,1,0},{2,1,0,3}};

public:
   void key_generation() {
      int key_[10];
      for (int i = 0; i < 10; i++)
         key_[i] = key[P10[i] - 1];
      int Ls[5], Rs[5];
      for (int i = 0; i < 5; i++) {
         Ls[i] = key_[i];
         Rs[i] = key_[i + 5];
      }
      shift(Ls, 1);
      shift(Rs, 1);
      for (int i = 0; i < 8; i++) {
         key1[i] = key_[P8[i] - 1];
      }
      shift(Ls, 2);
      shift(Rs, 2);
      for (int i = 0; i < 8; i++) {
         key2[i] = key_[P8[i] - 1];
      }
```

```
}

void shift(int arr[], int n) {
    while (n > 0) {
        int temp = arr[0];
        for (int i = 0; i < 4 - 1; i++) {
            arr[i] = arr[i + 1];
        }
        arr[3] = temp;
        n--;
    }
}

void encryption(int plaintext[], int *ciphertext) {
    int arr[8];
    for (int i = 0; i < 8; i++)
        arr[i] = plaintext[IP[i] - 1];
    int arr1[8];
    function_(arr, key1, arr1);
    swapPositions(arr1);
    int arr2[8];
    function_(arr1, key2, arr2);
    for (int i = 0; i < 8; i++)
        ciphertext[i] = arr2[IP_inv[i] - 1];
}

void function_(int ar[], int key_[], int output[]) {
    int l[4], r[4];
    for (int i = 0; i < 4; i++) {
        l[i] = ar[i];
        r[i] = ar[i + 4];
    }
    int ep[8];
    for (int i = 0; i < 8; i++)
        ep[i] = r[EP[i] - 1];
    for (int i = 0; i < 8; i++)
        ar[i] = key_[i] ^ ep[i];
    int l_1[4], r_1[4];
    for (int i = 0; i < 4; i++) {
        l_1[i] = ar[i];
        r_1[i] = ar[i + 4];
```

```
    }
    int row, col, val;
    row = stoi(to_string(l_1[0]) + to_string(l_1[3]), nullptr, 2);
    col = stoi(to_string(l_1[1]) + to_string(l_1[2]), nullptr, 2);
    val = S0[row][col];
    string str_l = binary_(val);
    row = stoi(to_string(r_1[0]) + to_string(r_1[3]), nullptr, 2);
    col = stoi(to_string(r_1[1]) + to_string(r_1[2]), nullptr, 2);
    val = S1[row][col];
    string str_r = binary_(val);
    int r_[4];
    for (int i = 0; i < 2; i++) {
        char c1 = str_l[i];
        char c2 = str_r[i];
        r_[i] = c1 - '0';
        r_[i + 2] = c2 - '0';
    }
    int r_p4[4];
    for (int i = 0; i < 4; i++)
        r_p4[i] = r_[P4[i] - 1];
    for (int i = 0; i < 4; i++)
        l[i] = l[i] ^ r_p4[i];
    for (int i = 0; i < 4; i++) {
        output[i] = l[i];
        output[i + 4] = r[i];
    }
}

void swapPositions(int array[]) {
    for (int i = 0; i < 4; i++)
        swap(array[i], array[i + 4]);
}

void decryption(int ar[], int *decrypted) {
    int arr[8];
    for (int i = 0; i < 8; i++)
        arr[i] = ar[IP[i] - 1];
    int arr1[8];
    function_(arr, key2, arr1);
    swapPositions(arr1);
    int arr2[8];
```

```
        function_(arr1, key1, arr2);
        for (int i = 0; i < 8; i++)
            decrypted[i] = arr2[IP_inv[i] - 1];
    }

    string binary_(int val) {
        if (val == 0)
            return "00";
        else if (val == 1)
            return "01";
        else if (val == 2)
            return "10";
        else
            return "11";
    }
};

int main() {
    DES obj;
    obj.key_generation();
    int plaintext[8] = {1,0,1,0,0,1,0,1};
    cout << "Plain Text: " << endl;
    for (int i = 0; i < 8; i++)
        cout << plaintext[i];
    int ciphertext[8];
    obj.encryption(plaintext, ciphertext);
    cout << endl;
    cout << "Your cipher Text is :" << endl;
    for (int i = 0; i < 8; i++)
        cout << ciphertext[i];
    int decrypted[8];
    obj.decryption(ciphertext, decrypted);
    cout << endl;
    cout << "Your decrypted Text is :" << endl;
    for (int i = 0; i < 8; i++)
        cout << decrypted[i];
    return 0;
}
```

Output:


Plain Text:
10100101
Your cipher Text is :
01101001
Your decrypted Text is :
10100101

6. Write a program to implement RSA algorithm.

```
#include<iostream>
#include<vector>
#include<cmath>
using namespace std;

vector< pair<long int,int>> private_key, public_key;

int gcd(int a,int b){
   if(a==0){
      return b;
   }
   return gcd(b%a,a);
}

long int modInverse(long int a, long int m)
{
   a = a%m;
   for (int x=1; x<m; x++)
     if ((a*x) % m == 1)
        return x;
}

long int power(long int x, unsigned int y, int p)
{
   long int res = 1;
   x = x % p;
   while (y > 0)
   {
      if (y & 1)
         res = (res*x) % p;
      y = y>>1;
      x = (x*x) % p;
   }
   return res;
}

void generation(int p,int q){
   int n=p*q;
   int phi=(p-1)*(q-1);
   long int e=3;
   while (gcd(e, phi) != 1)
   {
```

```
        e += 2;
    }
    long int d = modInverse(e, phi);
    private_key.push_back(make_pair(e,n));
    public_key.push_back(make_pair(d,n));
}

long int RSA(int pt,int x,int y){
    return power(pt, x, y);
}

int main(){
    cout<<"Assigned Prime Numbers: p=17,q=11"<<endl;
    generation(17,11);
    long int pt;
    cout<<"Plain Text: ";
    cin>>pt;
    pt=RSA(pt,private_key[0].first,private_key[0].second);
    cout<<"Encrypted Text: "<<pt<<endl;
    pt=RSA(pt,public_key[0].first,public_key[0].second);
    cout<<"Decrypted Text: "<<pt<<endl;
    return 0;
}
```

Output:

```
Assigned Prime Numbers: p=17,q=11
Plain Text: 88
Encrypted Text: 44
Decrypted Text: 88
```

7. Calculate the message digest of a text using the SHA-1 algorithm.

```
#include <bits/stdc++.h>
using namespace std;

typedef unsigned long int uint32;

uint32 hexCharToInt(char hexChar) {
   return hexChar >= '0' && hexChar <= '9' ? hexChar - '0' : hexChar - 'A' + 10;
}

uint32 hexToBinary(const string &hexString) {
   uint32 binaryValue = 0;
   for (char hexChar : hexString) {
      binaryValue = (binaryValue << 4) | hexCharToInt(hexChar);
   }
   return binaryValue;
}

string binaryToHex(uint32 binaryValue) {
   stringstream ss;
   ss << hex << setw(8) << setfill('0') << binaryValue;
   return ss.str();
}

uint32 rotateLeft(uint32 x, uint32 n) {
   return (x << n) | (x >> (32 - n));
}

uint32 f(uint32 t, uint32 b, uint32 c, uint32 d) {
   if (t < 20) return (b & c) | ((~b) & d);
   else if (t < 40) return b ^ c ^ d;
   else if (t < 60) return (b & c) | (b & d) | (c & d);
   else return b ^ c ^ d;
}

uint32 K(uint32 t) {
   if (t < 20) return 0x5A827999;
   else if (t < 40) return 0x6ED9EBA1;
   else if (t < 60) return 0x8F1BBCDC;
   else return 0xCA62C1D6;
}
```

```
void processBlock(uint32 *W, uint32 *H) {
    uint32 a = H[0];
    uint32 b = H[1];
    uint32 c = H[2];
    uint32 d = H[3];
    uint32 e = H[4];
    for (uint32 t = 0; t < 80; t++) {
        if (t >= 16)
            W[t] = rotateLeft(W[t-3] ^ W[t-8] ^ W[t-14] ^ W[t-16], 1);
        uint32 TEMP = rotateLeft(a, 5) + f(t, b, c, d) + e + W[t] + K(t);
        e = d;
        d = c;
        c = rotateLeft(b, 30);
        b = a;
        a = TEMP;
    }
    H[0] += a;
    H[1] += b;
    H[2] += c;
    H[3] += d;
    H[4] += e;
}

int main() {
    string input;
    cout << "Enter binary string: ";
    cin >> input;
    uint64_t input_length = input.length();
    input += '1';
    while (input.length() % 512 != 448) {
        input += '0';
    }
    string input_length_bin = bitset<64>(input_length).to_string();
    input += input_length_bin;

    uint32 H[5] = {0x67452301, 0xEFCDAB89, 0x98BADCFE, 0x10325476,
0xC3D2E1F0};
    uint32 W[80];
    memset(W, 0, sizeof(W));
    for (size_t i = 0; i < input.length(); i += 512) {
        for (size_t j = 0; j < 16; j++) {
            W[j] = hexToBinary(input.substr(i + j * 32, 32));
        }
```

```
        processBlock(W, H);
    }

    for (int i = 0; i < 5; i++) {
        cout << binaryToHex(H[i]) << " ";
    }
    return 0;
}
```

Output:

```
Enter binary string: 1001010010101010
Message Digest
afc5c09b 2d965cde 46a820ea 8d4b91f8 50af8362
```

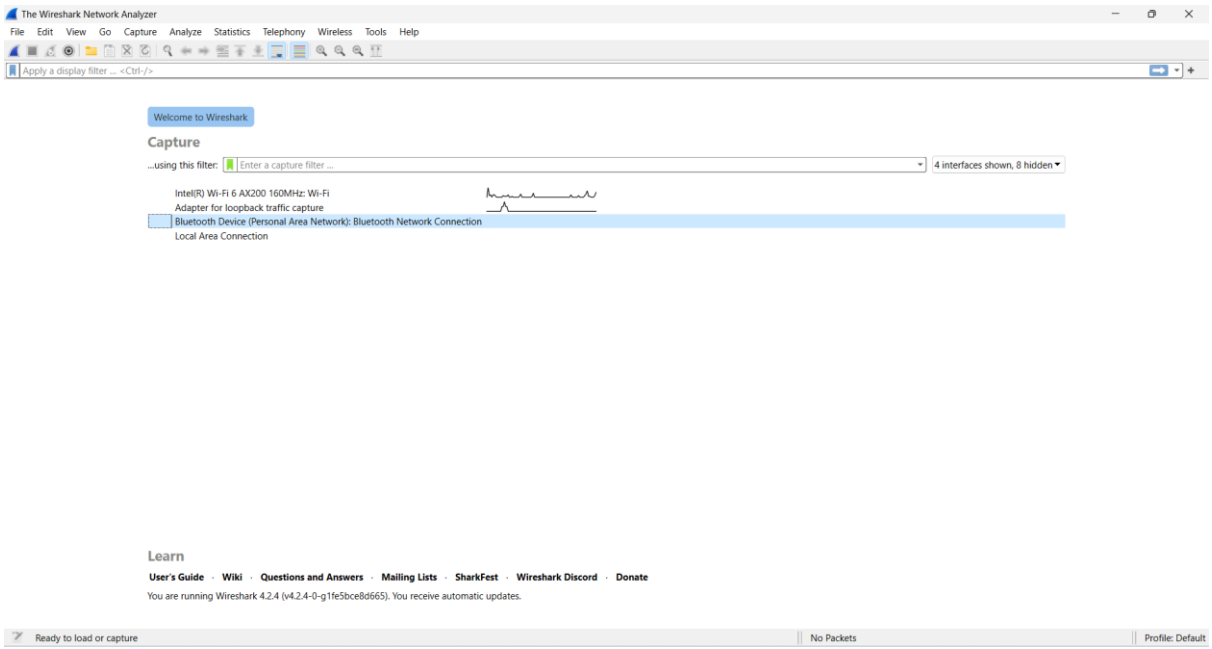
8. Working with sniffers for monitoring network communication (Wireshark).

Sniffing:

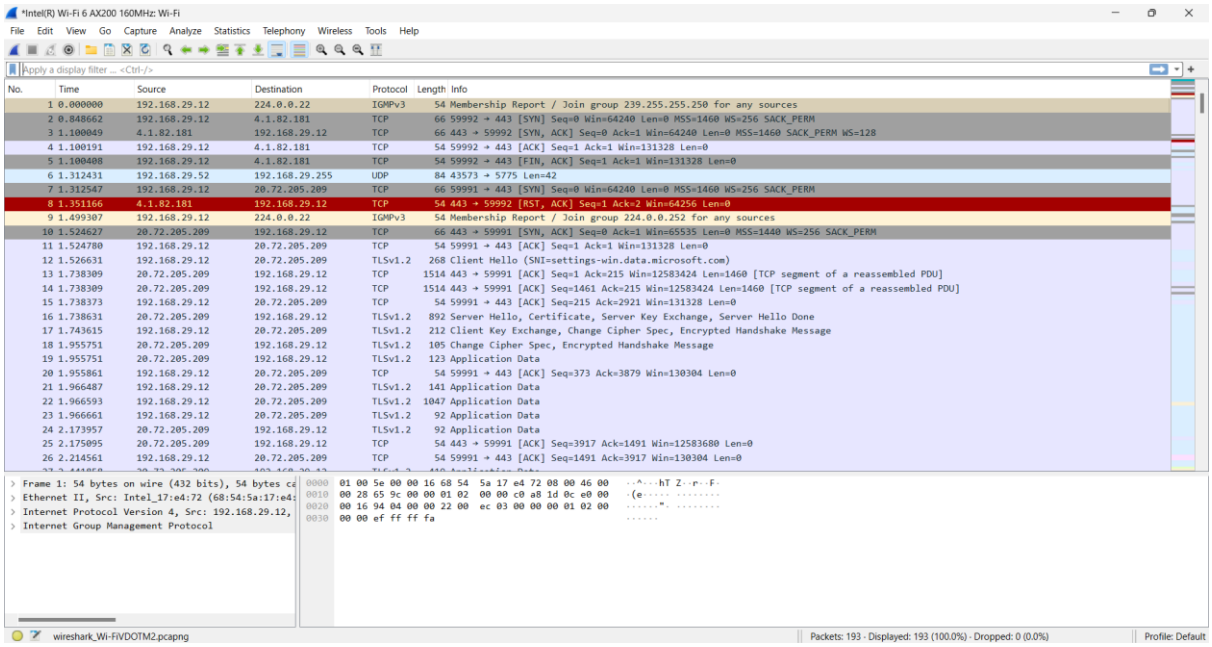
The process of intercepting and capturing network traffic is referred to as sniffing. A network sniffer, packet sniffer, or network analyzer, is a tool or piece of software that captures and analyzes data packets as they transit over a network. Sniffing is frequently used for valid reasons such as network troubleshooting, monitoring, and analysis. Sniffers can be used by network administrators to diagnose network problems, optimize performance, or investigate security vulnerabilities. It can, however, be abused for nefarious reasons.

Sniffers on Wireshark:

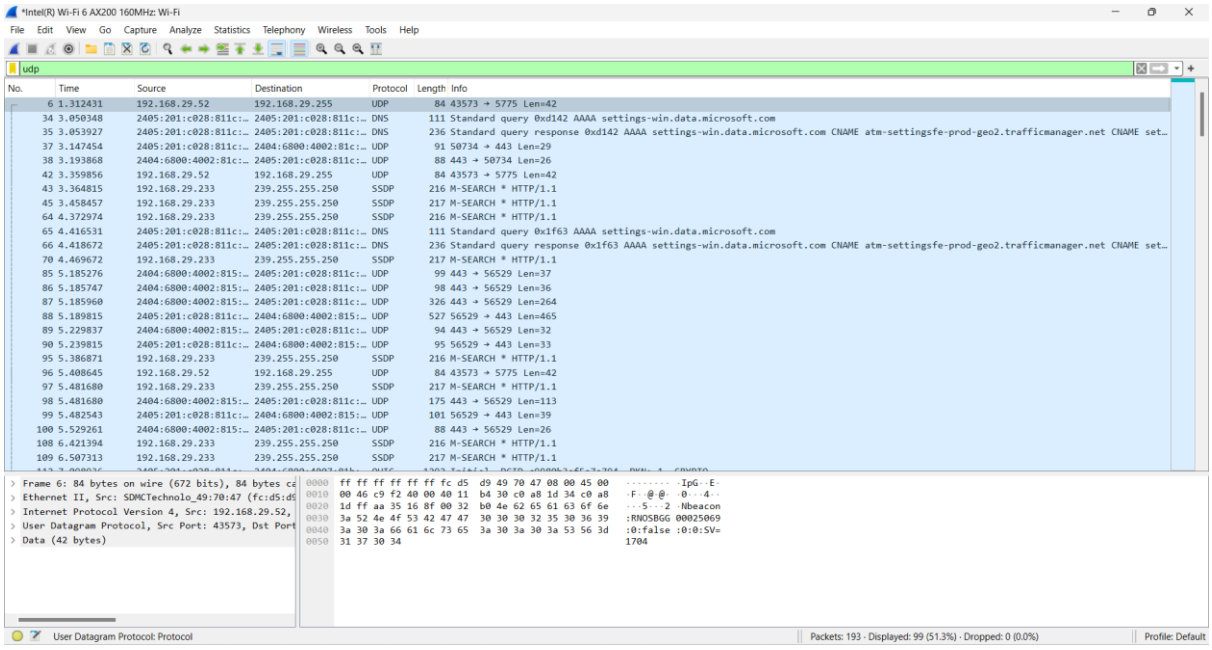
1. Open the Wireshark interface. Select the Wireless Fidelity as the mode for communication channel.



2. A list of scanned Data Packets is displayed on the Wireshark Interface. Click on the ‘Stop Scan’ icon to stop the scanning process.



3. For sniffing of certain data packets from the scan, type the respective protocol name on the ‘display filter’ search bar, say ‘udp’. The respective result on the interface is given as follows;



Coloring code for Scanned Data Packets: