Certificate

Name: Aditi Indoori Class: H1. (IT-1)

Roll No: 160118737001 Exam No:

Institution Chaitanya bharathi Institute of Technology.

This is certified to be the bonafide work of the student in the Information Security. *Laboratory during the academic year 20 / 20 .*

No. of practicals certified _____ *out of* _____ *in the subject of* _____

.....
Teacher In-charge

.....
Examiner's Signature

.....
Principal

Date: .....

Institution Rubber Stamp

(N.B: The candidate is expected to retain his/her journal till he/she passes in the subject.)

Index

Date

Expt. No. 1

Page No. 1

(1) Program to implement encryption & decryption using the following

(a) Substitution cipher — Polyalphabetic cipher.

```
import java.util.*;
import java.util.scanner;
class Polyalpha.{
    // This function generates the key in a cyclic
    // manner until its length isn't equal to.
    // the length of original text.

    static. String generateKey (String str, String key) {
        int x = str.(length());
        for (int i=0; ; i++) {
            if (x == i)
                i = 0;
            if (key.length() == str.length())
            break;
            Key += (key.charAt(i));
        }
        return key;
    }

// This function generates the encrypted text
// Generated with the help of the key
    static String cipherText (String str, String key) {
        String cipher_text = "";
        for (int i=0; i<str.length(); i++) {
            //converting in range 0-25
```

Teacher's Signature

```
            int x = (str.charAt(i) + key.charAt(i)) %. 26;
            //convert into alphabets (ASCII)
            x += 'A';
            cipher_text += (char)(x);
        }
        return cipher_text;
    }
    // this function decrypts the encrypted text
    // and returns the original text
    static String originalText (String cipher_text, String key) {
        String orig_text = "";
        for (int i=0; i< cipher_text.length() && i< key.length(); i++){
            //converting in range 0-25
            int x = (cipher_text.charAt(i) - key.charAt(i) +26)%26;
            //convert into alphabets (ASCII)
            x += 'A';
            orig_text += (char)(x);
        }
        return orig_text;
    }

public static void main (String[] args) {
    Scanner myObj = new Scanner (System.in); //Create a Scanner obj
    System.out.println ("Enter username");
    String str = myObj.nextLine();
    System.out.println ("Enter key");
    String keyword = myObj.nextLine();
    String key = generateKey (str, keyword);
    String cipher_text = cipherText (str, key);
```

```
System.out.println("Ciphertext : " + cipher_text + "\n");
System.out.println("Original/Decrypted Text :" +
                    originalText(cipher_text, key));
    }
}
```

Description

→ Polyalphabetic cipher is any cipher based on substitution, using multiple substitution alphabets.

→ The relationship between a character in the plain text & the characters in the cipher text is one-to-many

→ Polyalphabetic cipher is described as substitution cipher in which plain text letters in different positions are enciphered using different cryptoalphabets.

output

Substitution cipher - Polyalphabetic cipher.

Enter username
ADITI
Enter Key
SHH
Ciphertext : SKPLP
original / Decrypted Text : ADITI

(b) Transposition Cipher — Rail Fence Cipher.

```
import java.util.Scanner;
class railfenceCipherHelper {
    int depth;
    String encode(String msg, int depth) throws Exception {
        int r = depth;
        int l = msg.length();
        int c = l / depth;
        int k = 0;
        char mat[][] = new char[r][c];
        String enc = "";
        for (int i=0; i<c; i++) {
            for (int j=0; j<r; j++) {
                if (k!=l) {
                    mat[j][i] = msg.charAt(k++);
                } else {
                    mat[j][i] = 'x';
                }
            }
        }
        for(int i=0; i<r; i++) {
            for (int j=0; j<c; j++) {
                enc += mat[i][j];
            }
        }
        return enc;
    }
```

```
String decode(string encmsg, int depth) throws Exception {
    int r=depth;
    int l = encmsg.length();
    int c = l/depth;
    int k=0;
    char mat[][] = new char[r][c];
    String dec = "";
    for (int i=0; i<r; i++) {
      for (int j=0; j<c; j++) {
          mat[i][j] = encmsg.charAt(k++);
      }
    }
    for (int i=0; i<c; i++) {
      for (int j=0; j<r; j++) {
          dec += mat[j][i];
      }
    }
    return dec;
    }
}

class railfenceCipher {
    public static void main(string[] args) throws java.lang.Exception {
    railfenceCipherHelper rf = new railfenceCipherHelper();
    string enc, dec;
    Scanner myObj = new Scanner (System.in); //create a Scanner obj
    System.out.println("Enter username");
    String msg = myObj.nextLine();
    System.out.println("Enter depth);
```

```
int depth = myObj.nextInt();
enc = rf.encode(msg, depth);
dec = rf.decode(enc, depth);
System.out.println("Simulating Railfence Cipher\n.....");
System.out.println("Input Message:" + msg);
System.out.println("Encrypted Message:" + enc);
System.out.println("Decrypted Message:" + dec);
}
}
```

Description

→ The rail fence cipher is a form of transposition cipher.

→ In a transposition cipher, the order of the alphabets is rearranged to obtain the cipher-text.

→ Railfence cipher is a transposition cipher consisting in writing a text in zig-zag & read it from left to right.

encryption :

→ Railfence encryption uses an integer for the number of levels

→ The encoded message is written in zig-zag (like a railfence) along a path with N levels.

Eg: Encrypt DCODEZIGZAG with N=3 is writing

D E Z
C D Z G A
O I G

The cipher is read by rows.

Encrypted message is : DEZCDZGAOIG

Decryption :

→ Railfence decryption requires to know the number of levels N.

Output

Transposition cipher – Rail Fence cipher

```
Enter Username
ADITI
Enter depth
2
Simulating the Rail fence cipher.
- ---- --- ---- -
Input Message : ADITI
Encrypted Message : AIDT
Decrypted Message : ADIT
```

Date

Expt. No. 1

Page No. 7

(c) Product Cipher.

```
import java.util.*;
class Product Cipher1 {
    public static void main (String args[]) {
        System.out.println ("Enter the input to be encrypted:");
        String substitutionInput = new Scanner(System.in).nextLine();
        System.out.println ("Enter the key value");
        int k = new Scanner (System.in).nextInt();
        System.out.println ("Enter the depth of the transposition
                                cipher");
        int n = new Scanner (System.in).nextInt();
        // Substitution encryption
        StringBuffer substitutionOutput = new StringBuffer();
        for (int i=0; i<substitutionInput.length(); i++) {
            char c = substitutionInput.charAt (i);
            substitutionOutput.append ((char) (c+k));
        }
        System.out.println ("In Substituted text:");
        System.out.println (substitutionOutput);
        // Transposition encryption.
        String transpositionInput = substitutionOutput.toString();
        int modulus;
        if ((modulus = transpositionInput.length() % n) != 0) {
            modulus = n - modulus;
            // 'modulus' is now the number of blanks/padding (x)
            // to be appended
            for (; modulus != 0; modulus--) {
                transpositionInput += "/";
            }
        }
    }
```

Teacher's Signature

```
        StringBuffer transpositionOutput = new StringBuffer();
        System.out.println("\nTransposition Matrix:");
        for (int i=0; i<n; i++) {
            for (int j=0; j<transpositionInput.length()/n; j++) {
                char c = transpositionInput.charAt(i+(j*n));
                System.out.print(c);
                transpositionOutput.append(c);
            }
            System.out.println();
        }
        System.out.println("\nFinal encrypted text:");
        System.out.println(transpositionOutput);
        //Transposition decryption
        n= transpositionOutput.length()/n;
        StringBuffer transpositionPlaintext = new StringBuffer();
        for (int i=0; i<n; i++) {
            for (int j=0; j<transpositionOutput.length()/n; j++) {
                char c = transpositionOutput.charAt(i+(j*n));
                transpositionPlaintext.append(c);
            }
        }
        //substitution decry
        StringBuffer plaintext = new StringBuffer();
        for (int i=0; i<transpositionPlaintext.length(); i++) {
            char c = transpositionPlaintext.charAt(i);
            plaintext.append((char)(c-k));
        }
        System.out.println("\nPlain text:");
        System.out.println(plaintext);
    }
}
```

Teacher's Signature _____

Description

→ Product cipher, data encryption scheme in which the ciphertext produced by encrypting a plaintext document is subjected to further encryption.

→ By combining two or more simple transposition ciphers or substitution ciphers, a more secure encryption may result.

Output

Product cipher

Enter the input to be encrypted :

ADITI

Enter the key value

1

Enter the depth for transposition cipher.

1

Substituted text :

BEJUJ

Transposition Matrix :

BEJUJ

Final encrypted text -

BEJUJ

Plain text :

ADITI

Date

Expt. No. 2

Page No. 9

22. Program to implement Diffie-Hellman key exchange algorithm.

```
import java.io.*;
import java.math.BigInteger;
public class DEFFIE_HELLMAN {
    public static void main(String[] args) throws IOException {
        BufferedReader br = new BufferedReader(new InputStreamReader
                                                (System.in));
        System.out.println("Enter prime number:");
        BigInteger p = new BigInteger(br.readLine());
        System.out.print("Enter primitive root of "+p+":");
        BigInteger g = new BigInteger(br.readLine());
        System.out.println("Enter value for x less than"+p+":");
        BigInteger x = new BigInteger(br.readLine());
        BigInteger R1 = g.modPow(x,p);
        System.out.println("R1 ="+R1);
        System.out.println("Enter value for y less than"+p+":");
        BigInteger y = new BigInteger(br.readLine());
        BigInteger R2 = g.modPow(y,p);
        System.out.println("R2 ="+R2);
        BigInteger K1 = R2.modPow(x,p);
        System.out.println("key calculated at Alice's side:"+K1);
        BigInteger K2 = R1.modPow(y,p);
        System.out.println("Key calculated at Bob's side:"+K2);
        System.out.println("diffie hellman Secret Key Encryption has
                                                    taken");
    }
}
```

Teacher's Signature

Output

Enter prime number :
13
Enter primitive root of 13 : 5
Enter value for x less than 13 :
6
$R_1 = 12$
Enter value for y less than 13 : 3
$R_2 = 8$
Key calculated at Alice's side : 12
Key calculated at Bob's side : 12
deffie hellman secret key Encryption has Taken

Description

→ Deffie Hellman (DH) Key exchange algorithm is a method for securely exchanging cryptographic keys over a public communications channels.

→ Keys are not actually exchanged. They are jointly derived; named after their inventors whitfield Deffie & Martin Hellman

(3) Program to Implement AES Algorithm.

```
import java.util.Base64;
import java.util.Scanner;
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;

public class EncryptionDecryptionAES {
    static Cipher cipher;
    public static void main(String[] args) throws Exception {
        KeyGenerator keyGenerator = KeyGenerator.getInstance("AES");
        keyGenerator.init(128);
        SecretKey secretKey = keyGenerator.generateKey();
        cipher = Cipher.getInstance("AES");
        Scanner myobj = new Scanner(System.in);
        System.out.println("Enter plaintext");
        String plainText = myobj.nextLine();
        System.out.println("Plain text before encryption:" + plainText);
        String encryptedText = encrypt(plainText, secretKey);
        System.out.println("Encrypted text after encryption:" + encryptedText);
        String decryptedText = decrypt(encryptedText, secretKey);
        System.out.println("Decrypted text after decryption:" + decryptedText);
    }

    public static String encrypt(String plainText, SecretKey secretKey)
            throws Exception {
        byte[] plainTextByte = plainText.getBytes();
        cipher.init(Cipher.ENCRYPT_MODE, secretKey);
        byte[] encryptedByte = cipher.doFinal(plainTextByte);
```

Teacher's Signature _____

```
        Base64.Encoder encoder = Base64.getEncoder();
        String encryptedText = encoder.encodeToString(encryptedByte);
        return encryptedText;
    }

    public static String decrypt(String encryptedText, SecretKey secretKey)
            throws Exception {
        Base64.Decoder decoder = Base64.getDecoder();
        byte[] encryptedTextByte = decoder.decode(encryptedText);
        cipher.init(Cipher.DECRYPT_MODE, secretKey);
        byte[] decryptedByte = cipher.doFinal(encryptedTextByte);
        String decryptedText = new String(decryptedByte);
        return decryptedText;
    }
}
```

Description

The AES algorithm (also known as Rijndael algorithm) is a symmetrical block cipher algorithm that takes plain text in blocks of 128 bits & converts them to cipher text using keys of 128, 192, 256 bits.

Output

Enter plain text

aditi

Plain text before encryption : aditi

Encrypted Text after Encryption : w2dq55nbfQXInOtpPgI8HQ==

Decrypted Text after Decryption : aditi

Date _____

Expt. No. 4 Page No. 12

(4) Program to calculate the message digest of a text using the SHA-1 algorithm.

```
import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

public class SHA1 {
    public static String encryptThisString(String input)
    {
        try {
            // getInstance() method is called with algorithm SHA-1
            MessageDigest md = MessageDigest.getInstance("SHA-1");

            // digest() method is called to calculate message digest
            // of the input string returned as array of byte
            byte[] messageDigest = md.digest(input.getBytes());

            // convert byte array into signum representation
            BigInteger no = new BigInteger(1, messageDigest);

            // Convert message digest into hex value
            String hashtext = no.toString(16);

            // Add preceding 0s to make it 32 bit
            while (hashtext.length() < 32) {
                hashtext = "0" + hashtext;
            }
```

Teacher's Signature _____

```
        //return the hash text
        return hashtext;
    }
    // for specifying wrong message digest algorithms
    Catch (NoSuchAlgorithmException e) {
        throw new runtimeException(e);
    }
}

public static void main(String args[]) throws NoSuchAlgorithmException
{
    System.out.println("HashCode Generated by SHA-1 for ;");
    String S1 = "Aditi Indoori";
    System.out.println("\n" + S1 + ":" + encryptThisString(S1));
}
}
```

Description

SHA-1 or Secure Hash Algorithm 1 is a cryptographic hash function which takes an input & produces a 160-bit (20 byte) hash-value (message digest).

Output

Hashcode Generated by SHA-1 for :

Aditi Indoori : 64c34f6446a25c9cd52ad703a537cf00a49c5d6e

Date _____

Expt. No. 5 Page No. 14

(5) Program to calculate the message digest of a text using the MD5 algorithm.

```
import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

public class MD5 {
    public static String getMd5(String input)
    {
        try {
            // Static getInstance method is called with
            // hashing MD5
            MessageDigest md = MessageDigest.getInstance("MD5");
            // digest() method is called to calculate message
            // digest of an input digest() return array of byte
            byte[] messageDigest = md.digest(input.getBytes());
            // convert byte array into signum representation.
            BigInteger no = new BigInteger(1, messageDigest);
            // convert message digest into hex value
            String hashtext = no.toString(16);
            while (hashtext.length() < 32) {
                hashtext = "0" + hashtext;
            }
            return hashtext;
        }
    }
```

Teacher's Signature _____

```
// for specifying wrong message digest algorithms
catch (NoSuchAlgorithmException e) {
    throw new RuntimeException(e);
}
}

public static void main(String args[]) throws NoSuchAlgorithmException
{
    String s = "Aditi Indoori";
    System.out.println("Your Hash Code Generated by MD5 is: " +
                       getMD5(s));
}
}
```

Description

The MD5 hashing algorithm is a one-way cryptographic function that accepts a message of any length as input & returns as output a fixed-length digest value to be used for authenticating the original message.

output

Your hash code generated by MD5 is : b9f620e9b3cb069a0f04596----e5

Date _____

Expt. No. 6 Page No. 16

(6) Program to implement Digital Signature Algorithm.

```
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.security.SecureRandom;
import java.security.Signature;
import java.util.Scanner;
import javax.xml.bind.DatatypeConverter;

public class Digital_Signature {
    //Signing Algorithm.
    private static final String
        SIGNING_ALGORITHM = "SHA256withRSA";
    private static final String RSA = "RSA";
    private static Scanner sc;

    //function to implement Digital signature. Using
    //SHA256 & RSA algorithm, by passing private key
    public static byte[] create_Digital_Signature (
        byte[] input,
        PrivateKey key)
    throws Exception {
        Signature signature = Signature.getInstance(
            SIGNING_ALGORITHM);
        Signature.initSign(key);
        Signature.update(input);
        return Signature.sign();
    }
```

Teacher's Signature _____

```
//Generating asymmetric key pair using SecureRandom
//class functions & RSA algorithm
public static KeyPair Generate_RSA_key_pair() throws
    Exception {
    SecureRandom secureRandom = new SecureRandom();
    KeyPairGenerator keyPairGenerator = KeyPairGenerator.getInstance(RSA);
    keyPairGenerator.initialize(2048, secureRandom);
    return keyPairGenerator.generateKeyPair();
}
//function for verification of the digital signature using
//public key
public static boolean verify_Digital_signature(
    byte[] input,
    byte[] signatureToVerify,
    PublicKey key)
    throws Exception.
{
    Signature signature = Signature.getInstance(
                    ~~SIGNATURE~~
                    SIGNING_ALGORITHM);
    signature.initVerify(key);
    signature.update(input);
    return signature.verify(signatureToVerify);
}

public static void main(String args[]) throws exception {
    Scanner myobj = new Scanner(System.in);
    System.out.println("Enter plaintext");
    String input = myobj.nextLine();
```

```
KeyPair KeyPair = Generate_RSA_KeyPair();

byte[] Signature = create_Digital_Signature(
        input.getBytes(),
        KeyPair.getPrivate());

System.out.println("Signature Value : \n" + DataTypeConverter
        .printHexBinary(Signature));

System.out.println("Verification: " + verify_Digital_Signature(
        input.getBytes(),
        Signature, KeyPair.getPublic()));

    }
}
```

Description

Digital signatures are the public-key primitives of message authentication. A digital signature is a technique that binds a person/entity to the digital data. It is a cryptographic value that is calculated from the data & a secret key only known by the signer.

output

Enter plain text : Hi

Signature value 9000625684 7E 06 D5 ..... C65

Verification : True

Date _____

Expt. No. 7 Page No. 19

(7) Blowfish Algorithm.

```
import javax.crypto.cipher;
import javax.crypto.spec.secretkeyspec;
import java.util.Base64;

class Blowfish {
    private static final string key = '128';
    public static string encrypt (string password) {
        try {
            byte[] keydata = (key).getBytes();
            SecretKeySpec secretkeySpec = new SecretKeySpec
                                    (Keydata, "Blowfish");
            Cipher cipher = cipher.getInstance ("Blowfish");
            Cipher.init (Cipher ENCRYPT_MODE, secretkeySpec);
            byte[] hasil = cipher.doFinal (password.getBytes())
            return new string (Base64.getEncoder().encode (hasil))
        }

        catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }

    public static void main (string[] args) {
        system.out.println ("your encrypted word is" + encrypt ("Hello"));
    }
}
```

Teacher's Signature _____

O/P:

Your encrypted word is c d w ix a b J I Q L

Description

Blowfish is an encryption technique designed as an alternative to DES encryption technique. It is significantly faster than DES & provides good encryption rate with no effective cryptanalysis technique found to date.

Date _____

Expt. No. 8 Page No. 20.

WIRESHARK.

Wireshark is an opensource packet analyser which is used for education, analysis, software development, communication protocol development & network troubleshooting.

Features :

→ It is a multiplatform software i.e. it can run on Linux, windows, OS X, NetBSD etc.
→ It is a standard three-pane packet browser.
→ It performs deep inspection of the hundreds of protocols.
→ It can also capture raw USB traffic
→ It is also useful in VOIP analysis.

Installation :

(1) Open the web browser
(2) Search for 'Download wireshark'
(3) Select the windows installer according to your system configuration, either 32-bit or 64-bit. Save the program in browser.
(4) Now open the software & follow the install instruction by accepting the license.
(5) Wireshark is ready for use.

Why wireshark ?

→ wireshark is used to track the packets so that each one is filtered to meet our specific needs
→ It is commonly called as sniffer, network protocol analyser & network analyser.
→ It is also used by network security engineers to examine security problems

Teacher's Signature _____

Date _____

Expt. No. 8 Page No. 21

Uses of wireshark

→ It is used by network security engineers to examine security problems

→ It allows the users to watch all the traffic being passed over the network

→ It is used by network engineers to troubleshoot network issues.

→ It also helps to troubleshoot latency issues & malicious activities on your computer.

Filters in wireshark

(1) HTTP

Time	Source	Destination	Protocol	Length.
204624.966343	192.250.195.174	192.168.1.104	HTTP	137

(2) TCP

Time	Source	Destination	Protocol	Length
29356180.039468	192.168.1.104	23.58.36.195	TCP	54 [FIN ACK] Seq. =1699 Win = 131328 Len = 0.

(3) UDP

Time	Source	Destination	Protocol	Length	Info.
10.000	192.168.1.104	74.125.250.13	UDP	120	51[illegible] → 19305 Len = 78

Teacher's Signature _____

Date _____

Expt. No. 9 Page No. 22

NMAP

NMAP, short for Network Mapper is a network discovery & security auditing tool.

Uses:

→ NMAP is widely used by network administrators to scan for:

- Open ports & services
- Discover services along with their versions
- Guess the operating system running on a target machine
- Monitoring tools.

Common NMAP features

- PING SCANNING
- PORT SCANNING
- HOST SCANNING
- OS SCANNING
- SCAN TOP PORTS
- OUTPUT TO FILES
- DISABLE DNS RESOLUTION.

Scans:

(1) Intense Scan: A comprehensive scan. Contains operating system (OS) detection, version detection, script scanning, traceroute & has aggressive scan timing. This is considered an intrusive scan.

Teacher's Signature _____

(2) Ping Scan : This scan simply detects if the targets are online, it doesn't scan any ports.

(3) Quick scan : This is quicker than a regular scan due to aggressive timing & only scanning select ports.

(4) Regular scan : This is the standard Nmap scan without any modifiers. It will return ping & return open ports on the target.

Ex: Target IP address : 192.168.1.1

HOST DETAILS :

open ports : 2
filtered ports : 0
closed ports : 998

Addresses
IPV4 : 192.168.1.1
IPV6 : Not available
MAC : 80:3F:5D:87:3F:66

Operating system
Name : Linux 2.6.9 – 2.6.33

Date _____

Expt. No. 10 Page No. 24

Kf Sensor

→ Kf Sensor is a windows based honeypot intrusion Detection System (IDS). It acts as a honeypot to attract & detect hackers & worms by simulating vulnerable system services & trojans. By acting as a decoy server it can divert attacks from critical systems and provide a higher level of information than can be achieved by using firewalls & NIDS alone. KfSensor is a system installed in a network in order to divert & study an attackers behaviour. This is a new technique that is effective in detecting attacks.

→ The main feature of KfSensor is that every connection it receives is a suspect hence it results in very few false alerts. At the heart of KfSensor sits a powerful internet daemon service that is built to handle multiple ports & IP addresses.

→ It is written to resist denial of service & buffer overflow attacks. Building on this flexibility KfSensor can respond to connections in a variety of ways from simple port listening & basic services (such as echo, to complex simulations of standard system services.

Teacher's Signature _____

Date _____

Expt. No. 10 Page No. 25

→ For HTTP protocol KF Sensor accurately simulates the way microsoft's web server (IIS) responds to both valid & invalid requests

→ As well as being able to host a website it also handles complexities such as range requests & client side cache negotiations. This is extremely difficult for an attacker to fingerprint, or identify KF Sensor as a honeypot.

Teacher's Signature _____

Date _____

Expt. No. 11 Page No. 26

Rootkits

→ A rootkit is a stealthy type of malicious software (malware) designed to hide the existence of certain processes or programs from normal method of detections & enables continued privileged access to a computer.

→ The term rootkit is a concatenation of "root" (the privileged account on Unix OS) & "kit" (software components that implement the tool).

→ A rootkit is a collection of tools that enable administrator level access to a computer.

→ A rootkit may consist of spyware & other programs that monitor traffic & keystrokes; create a "backdoor" into the system for hacker's use; alter log files; attack other mechanisms on the network & alter existing system tools to escape detection.

→ Types of Rootkits

- User-mode or application rootkit
- Kernel-mode
- Bootkits
- Firmware rootkits
- Rootkit Hypervisors

Teacher's Signature _____

Date _____

Expt. No. 11 Page No. 27

→ Study about variety options

Steps:

(1) Double click on rootkit folder
(2) Double click on the GMER rootkit application.
(3) Now the rootkit screen will be displayed.
(4) Select any one of the drive which is shown at right side of the screen.
(5) After selecting the drive click on Scan button
(6) Click on the option processes the screen will be displayed.
(7) Click on option services
(8) Now click on different options to perform different actions.

Teacher's Signature _____

Date _____

Expt. No. 12 Page No. 28

TCP Dump & Dumpcap

→ TCP dump is a data-network packet analyzer. computer program that runs under a command. line interface. It allows the user to display TCP/IP and other packets being transmitted or received. over a network to which the computer is attached.

→ TCP dump uses libcap library to capture the network packets & is available on almost all Linux / unix flavors.

→ It is used to capture, filter & analyze network. traffic such as TCP/IP packets going through your system. It is mainly used as a security tool. It saves the captured information in a pcap file, these pcap files can be opened through wireshark or through the command tool itself.

* To capture the packets of current network interface.
→ Sudo tcp dump.

* To display all available interfaces.
→ Sudo tcp dump -D.

* To capture packets from a specified network interface.
→ Sudo tcpdump -i wlo1

Teacher's Signature _____

Date _____

Expt. No. 12 Page No. 29

* To capture specific number of packets
→ sudo tcpdump -c 4 -i wlo1

* To print captured packets in ASCII format.
→ sudo tcpdump -A -i wlo1

* To capture only TCP packets
→ sudo tcpdump -i wlo1 tcp.

* Set all the packets based on the IP addresses whether source or destination or both, using the following command.
→ sudo tcpdump host 192.168.1.100.

* To get packets based on source or destination of an IP address, use.
→ sudo tcpdump src 192.168.1.100
→ sudo tcpdump dst 192.168.1.100

→ Dumpcap is a network traffic dump tool. It captures packet data from a live network & writes the packets to a file.

→ Dumpcap's native capture file format is pcapng, which is also the format used by Wireshark.

→ By default dumpcap uses the pcap library to capture traffic from the first available network interface & writes the received raw packet data, along with the packets time stamps into a pcapng file.

Teacher's Signature _____

Date _____

Expt. No. 13 Page No. 30

Snort Tool

Intrusion detection using SNORT TOOL :

→ A general thought is that if a firewall is protecting one's network, the network can be considered secure.

→ Intrusion detection system is used to evaluate. aggressive (or) unexpected. packets & generate an alert before these programs harm the network.

→ An advantage of host based intrusion detection systems is that it can also detect anomalies (or) malicious traffic, generated from the host itself.

→ IDS work by monitoring & analysing network traffic & compare it with an established ruleset.

→ SNORT is a flexible, light weight & popular system that can be deployed according to the needs of the network.

Teacher's Signature _____

INFORMATION SECURITY

LAB RECORD – PROGRAM OUTPUTS

Aditi Indoori
160118737001
IT1 (H1)

1. Program to implement encryption and decryption using the following:

(a) Substitution Cipher – Polyalphabetic Cipher

```
D:\6th semester\Information Security Lab>javac Polyalpha.java

D:\6th semester\Information Security Lab>java Polyalpha
Enter username
ADITI
Enter key
SHH
Ciphertext : SKPLP

Original/Decrypted Text : ADITI
```

(b) Transposition Cipher – Railfence Cipher

```
D:\6th semester\Information Security Lab>javac railFenceCipher.java

D:\6th semester\Information Security Lab>java railFenceCipher
Enter username
ADITI
Enter depth
2
Simulating Railfence Cipher
------------------------
Input Message : ADITI
Encrypted Message : AIDT
Decrypted Message : ADIT
```

(c) Product Cipher

```
D:\6th semester\Information Security Lab>javac ProductCipher1.java

D:\6th semester\Information Security Lab>java ProductCipher1
Enter the input to be encrypted:
ADITI
Enter the key value
1
Enter the depth for transposition cipher
1

Substituted text:
BEJUJ

Transposition Matrix:
BEJUJ

Final encrypted text:
BEJUJ

Plaintext:
ADITI
```

2. Program to implement Diffie Hellman Key Exchange Algorithm

```
D:\6th semester\Information Security Lab>javac DEFFIE_HELLMAN.java

D:\6th semester\Information Security Lab>java DEFFIE_HELLMAN
Enter prime number:
13
Enter primitive root of 13:5
Enter value for x less than 13:
6
R1=12
Enter value for y less than 13:3
R2=8
Key calculated at Alice's side:12
Key calculated at Bob's side:12
deffie hellman secret key Encryption has Taken
```

3. Program to implement AES Algorithm

```
D:\6th semester\Information Security Lab>javac EncryptionDecryptionAES.java

D:\6th semester\Information Security Lab>java EncryptionDecryptionAES
Enter plaintext
aditi
Plain Text Before Encryption: aditi
Encrypted Text After Encryption: bNzgIh3d3t5aDRTLgICtWA==
Decrypted Text After Decryption: aditi
```

4. Program to implement SHA-1

```
D:\6th semester\Information Security Lab>javac SHA1.java

D:\6th semester\Information Security Lab>java SHA1
HashCode Generated by SHA-1 for:

AditiIndoori : 64c34f6446a25c9cd52ad703a537cf00a49c5d6e
```

5. Program to implement MD5

```
D:\6th semester\Information Security Lab>javac MD5.java

D:\6th semester\Information Security Lab>java MD5
Your HashCode Generated by MD5 is: b9f620e9b3cb069af045a6ab1ffa5aeb
```

6. Program to implement Digital Signature Algorithm

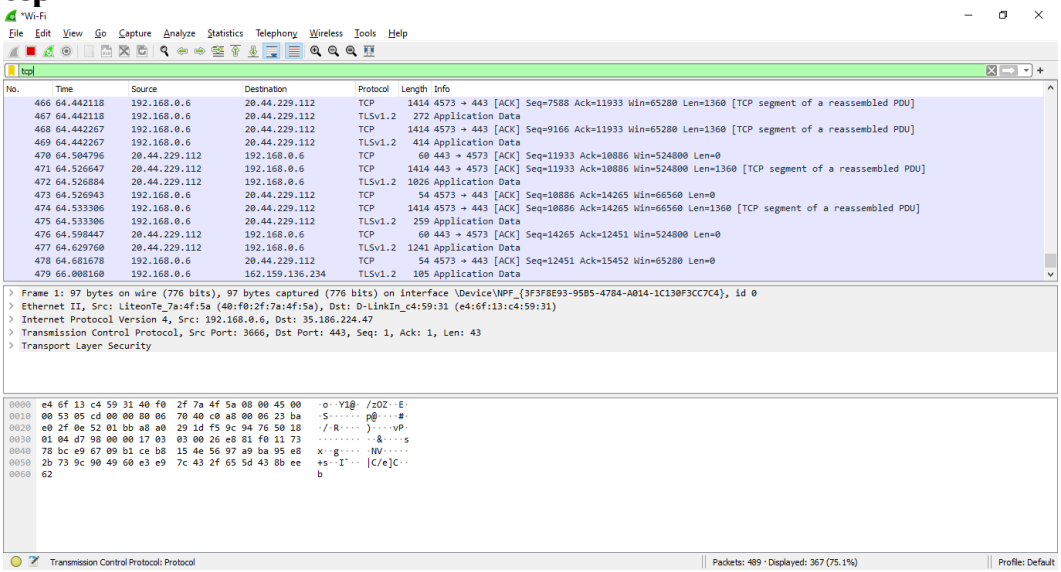
```
Enter plaintext
hi
Signature Value:
 885D717C9177137B88350440967145D5455DE44089A4AFF1E87915C5A2FDD1BCE9DD5708B76BC1F5EDD978242245FABE0C44C1CE2E0CB679E7B146
40BB6288194BB5A46C01DBE97AAF3D67B7E4A4F64012BE5002D08A03BE14AAEF86A77A75DB69002A31DF251403A30FBB11F400FC030B9EF2BB7527C
B2BDAABBACDAEAD1F673277CA2376DF9160A6EE933C05C1829B58C8BE1C49CEB4776A328C0AA1AE004EB2A358CA005BB583331FEDDF2338F6B6BC87
CB8B75FF07D814BB6290E1727E045D7BD865048D48B1E39489A54377ABC783999CD9BF1914E8CBB46D4EB4FF86F065FB34BBFF7A0963B62D7F9064D
6B92DB05C4F67886DB1D2EBE020FDD5A8D2F6
Verification: true
```

7. Blowfish Algorithm

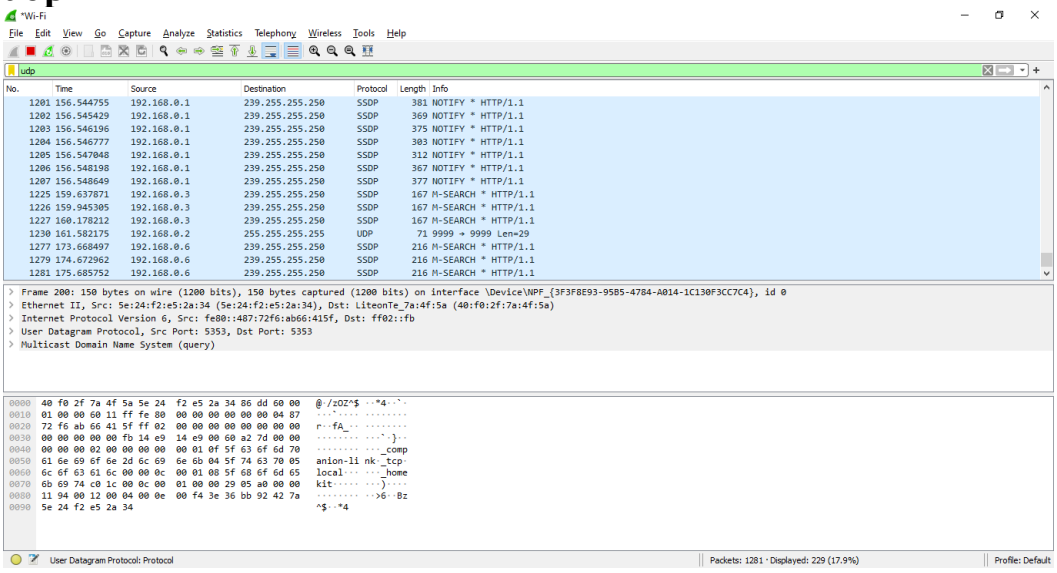
```
D:\6th semester\Information Security Lab>javac Blowfish.java

D:\6th semester\Information Security Lab>java Blowfish
your encrypted word isaE9JrBKj0xk=
```

8. Wireshark

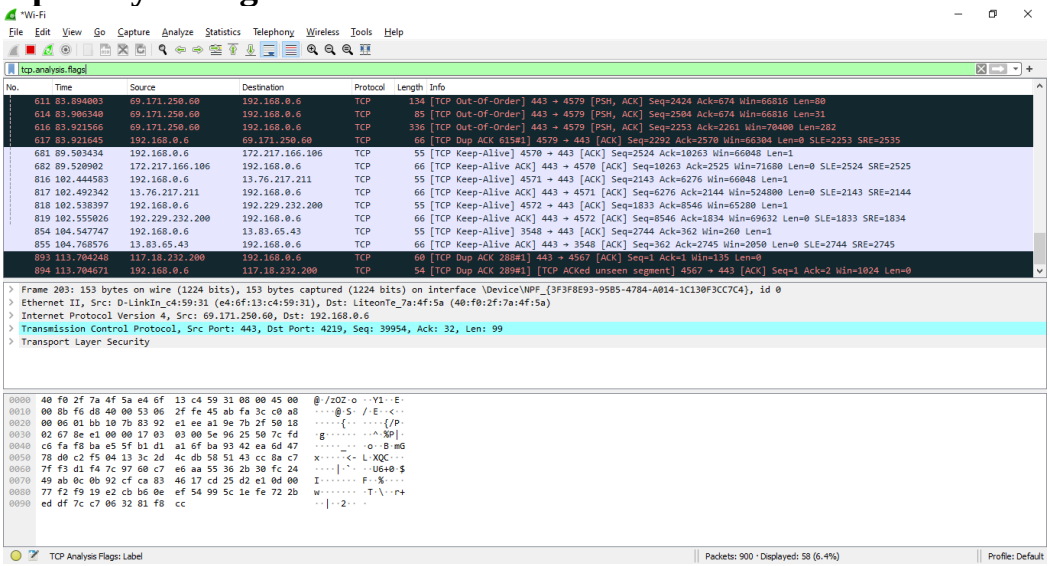
tcp



udp

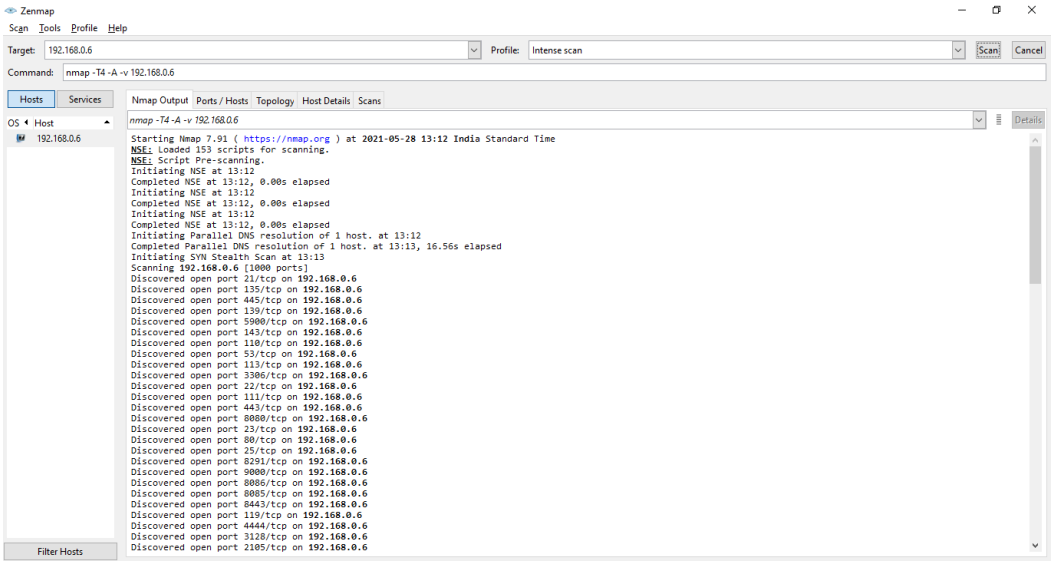


tcp.analysis.flags

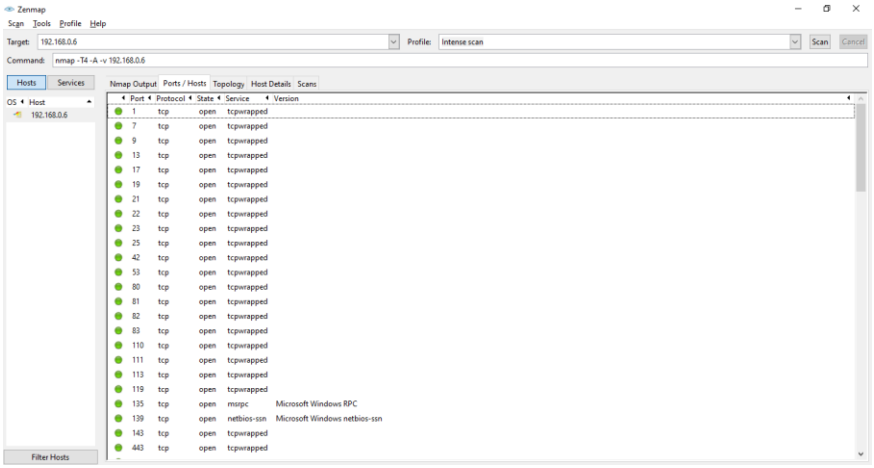


9. Nmap

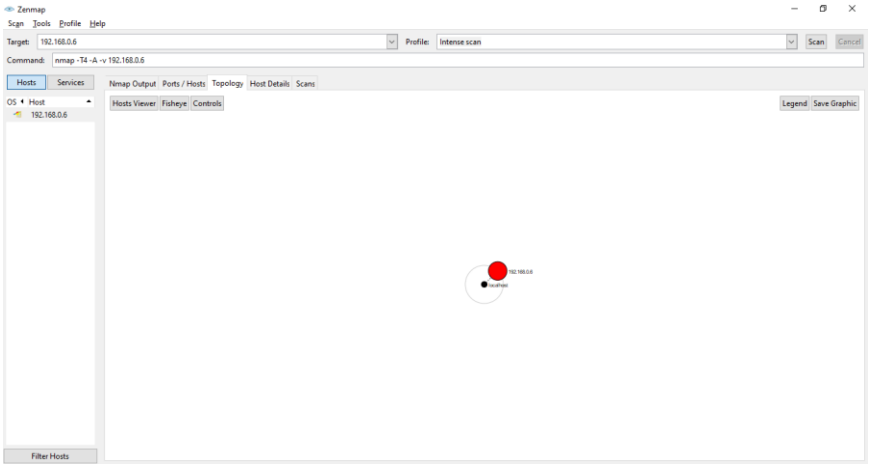
Intense scan



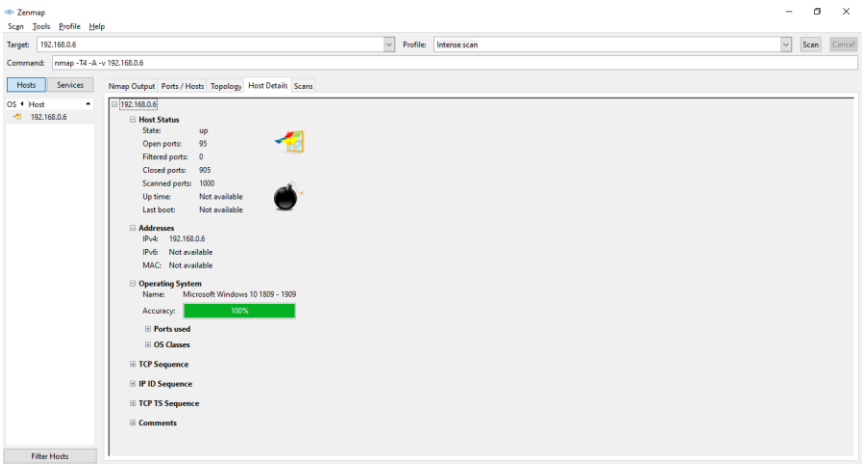
- **Ports/Hosts**



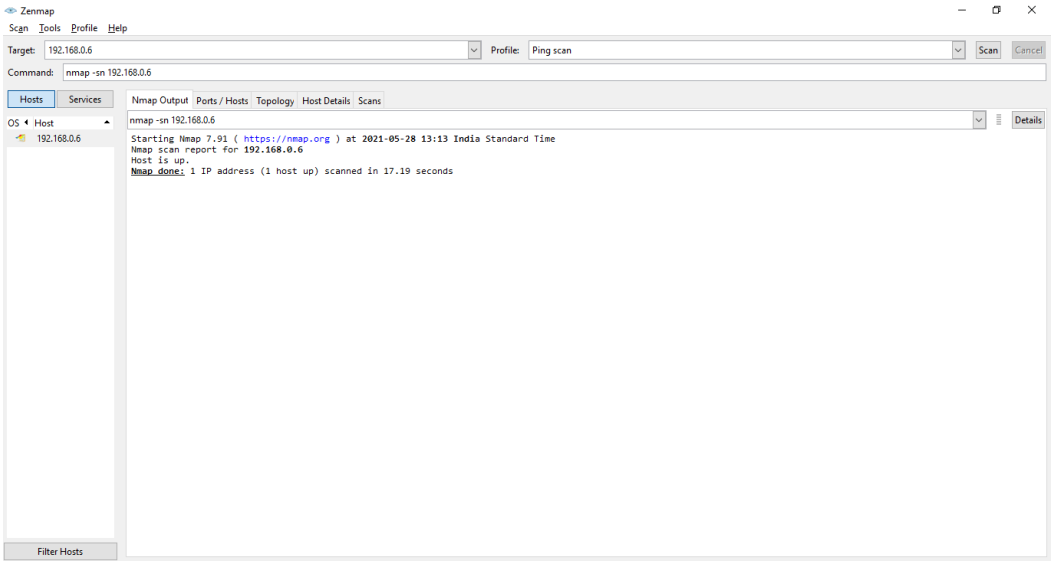
- **Topology**



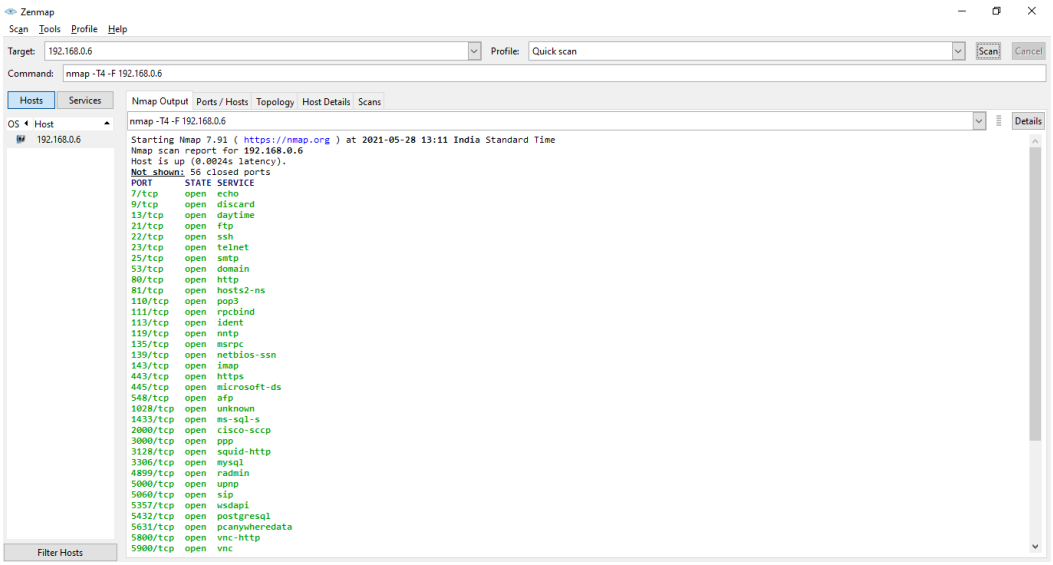
- **Host Details**



Ping scan

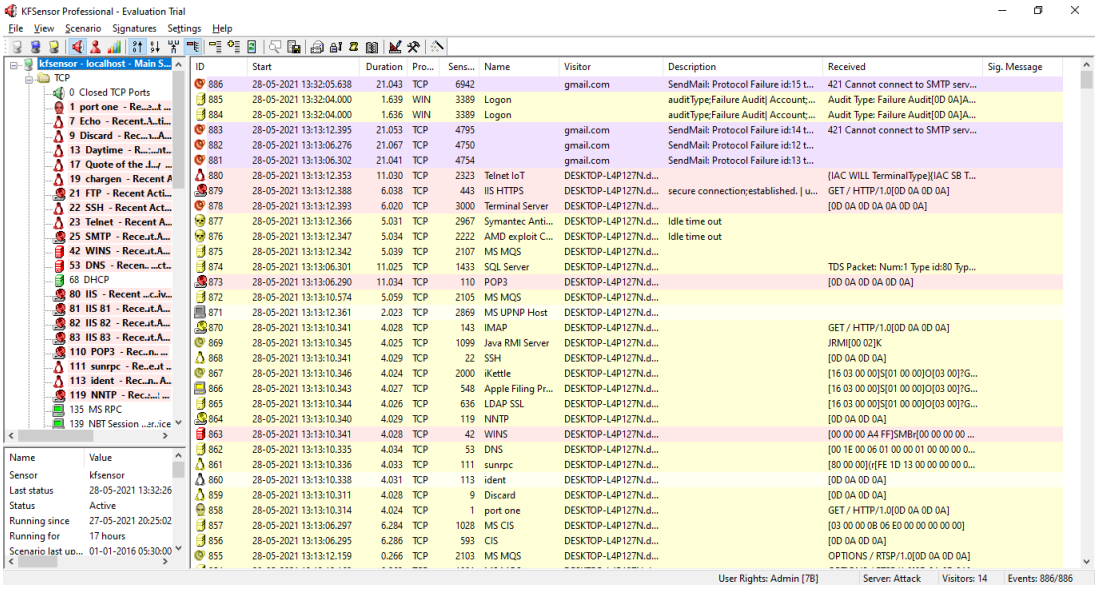


Quick scan

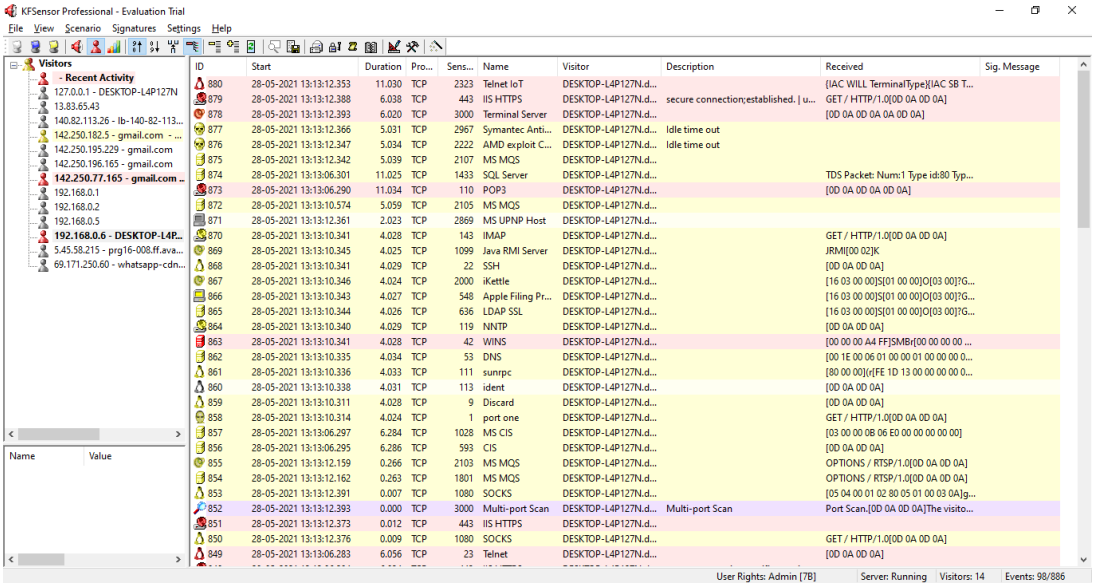


10. KFSensor

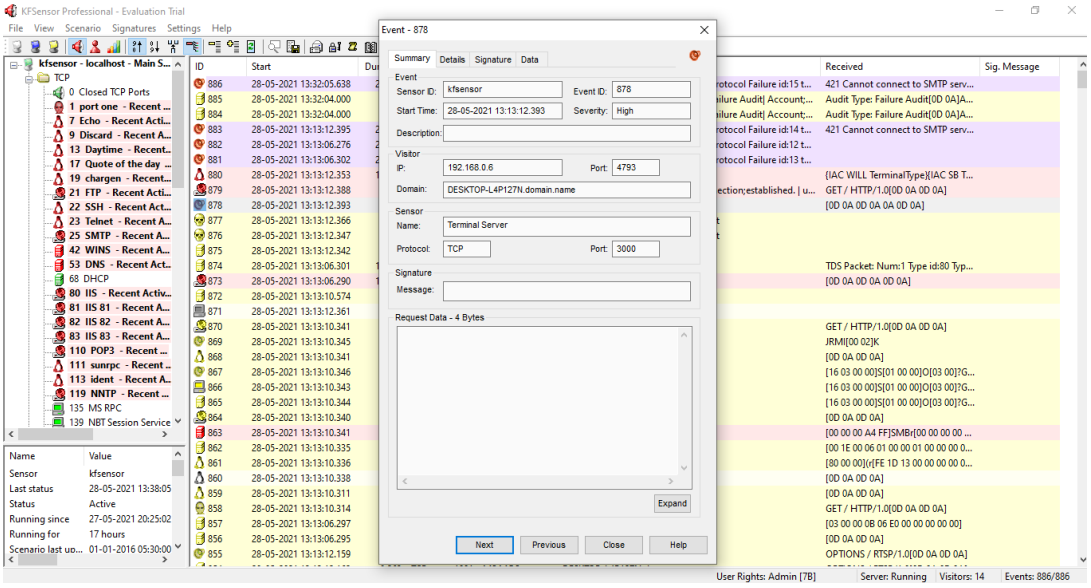
Ports



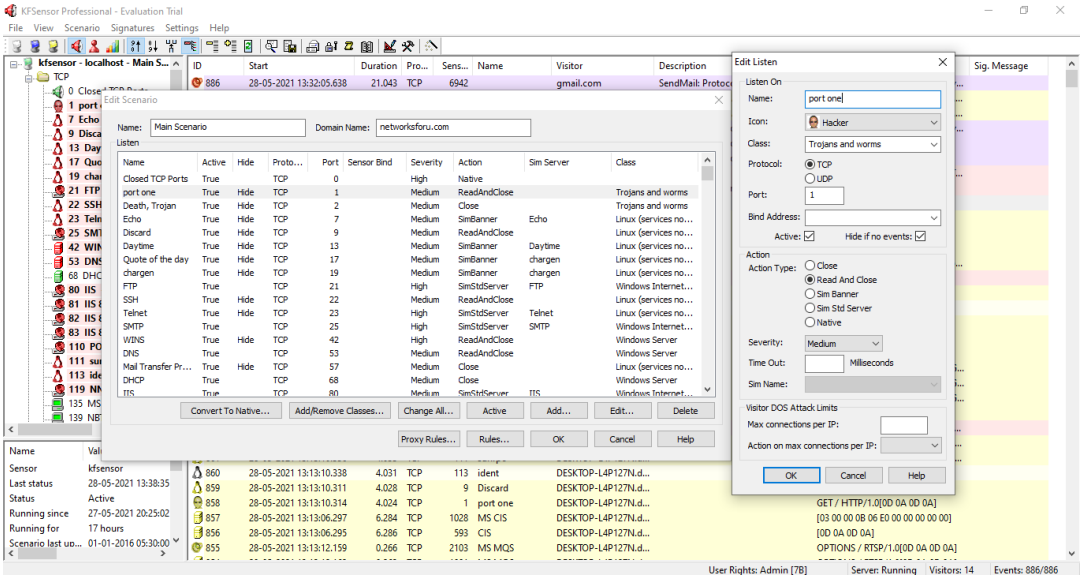
Visitors



Event Details

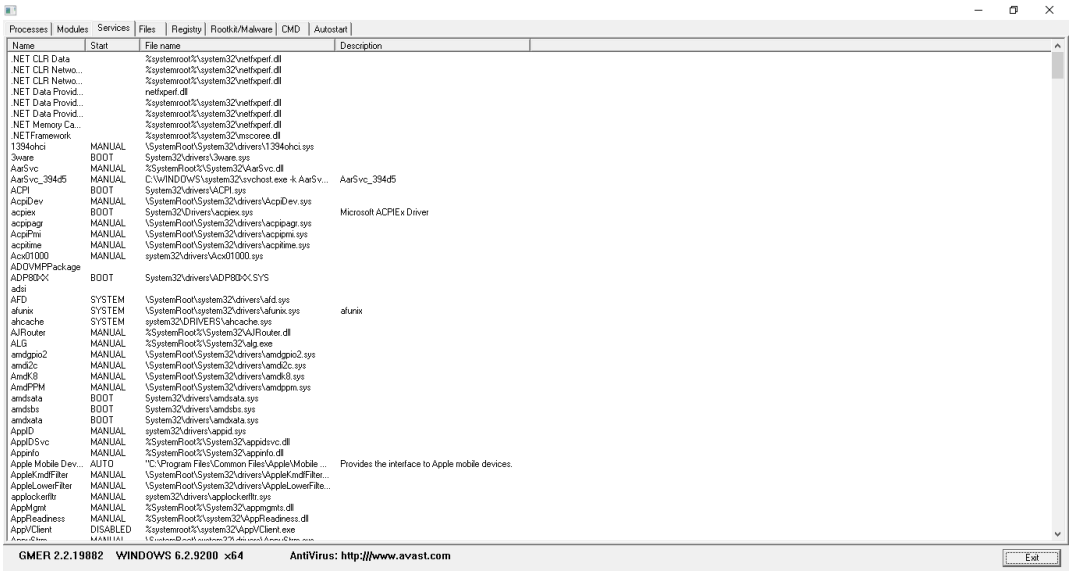


Edit Scenarios

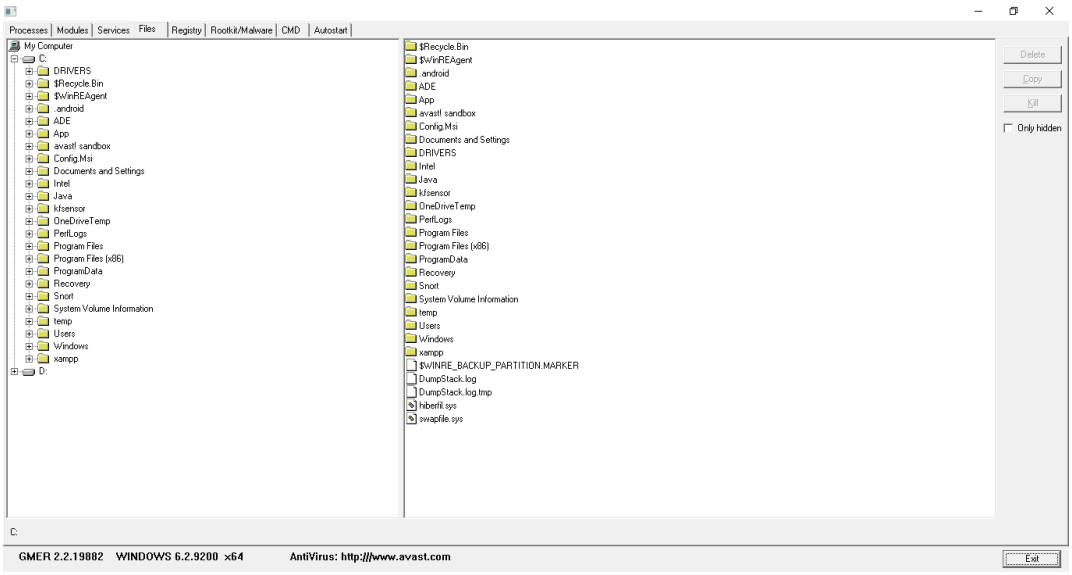


11. Rootkits

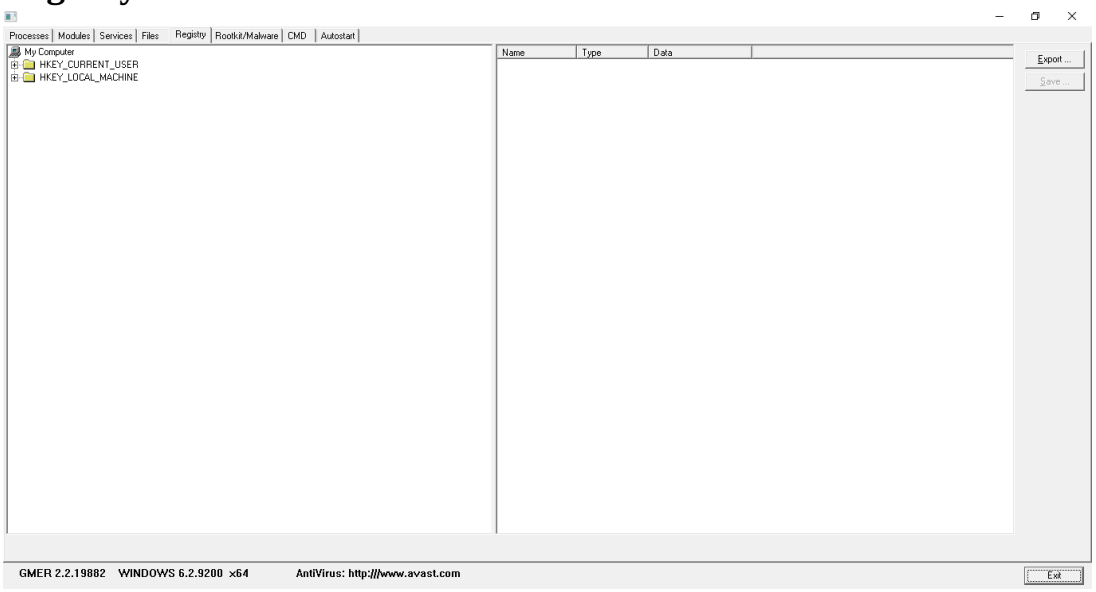
Services



Files

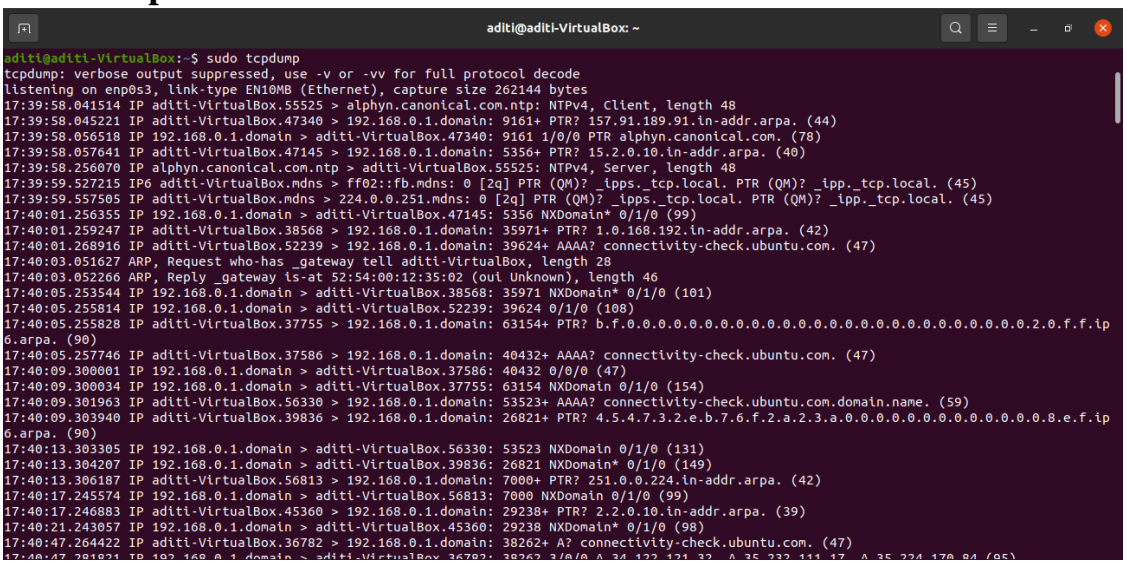


Registry

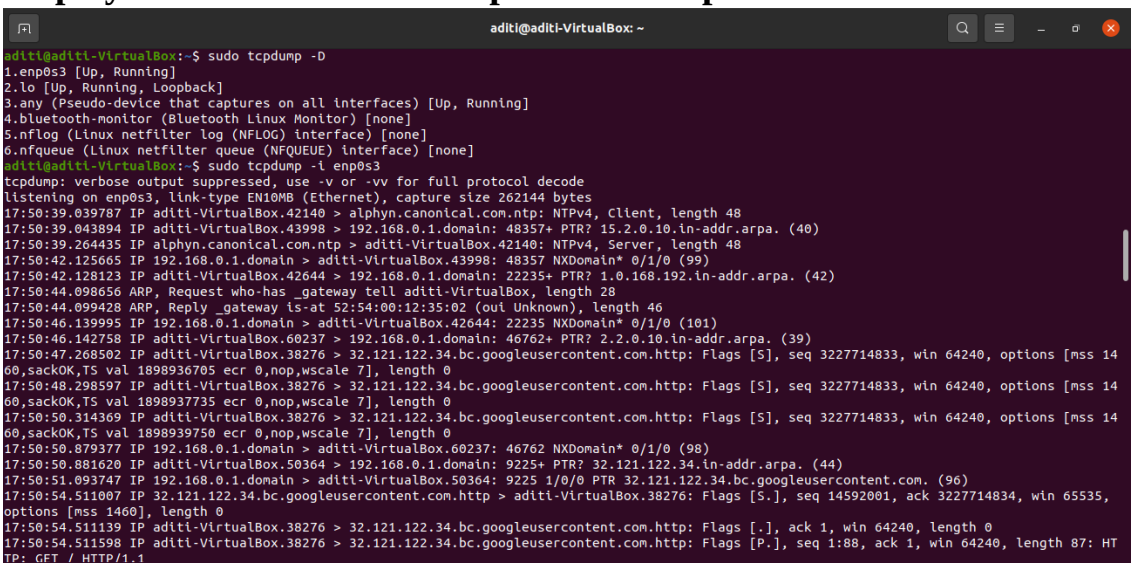


12. TCPDump & Dumpcap

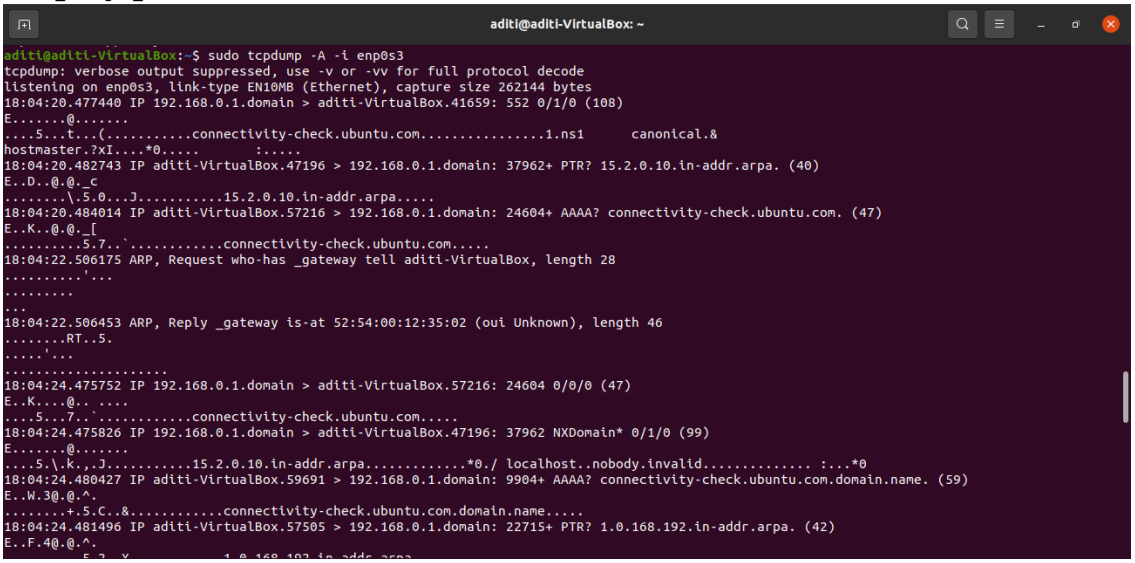
Show all packets



Display all interfaces and find packets of a specific interface



Display packets information in ASCII format



13. Snort Tool