INFANT JESUS COLLEGE OF ENGINEERING

KAMARAJAR NAGAR

KEELAVALLANADU-628 851

IT8761 – Security Laboratory

NAME : ----------

REGISTER NUMBER : ----------

INFANT JESUS COLLEGE OF ENGINEERING

KAMARAJAR NAGAR

KEELAVALLANADU-628 851

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NAME : ----------

REGISTER NUMBER : ----------ROLL NO: ----------

BRANCH : ----------

YEAR : ----------

Bonafide Record of work done in the **IT8761 – Security Laboratory** at Infant Jesus College of Engineering, Keelavallanadu during the year **2020-2021**

STAFF INCHARGE **H.O.D**

Submitted for the Practical Examination held on ----------

at Infant Jesus College of Engineering,Keelavallanadu.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

Ex. No.	*Name of the Experiment*
1.	Perform encryption, decryption using the following substitution techniques **i.** Ceaser cipher **ii.** Playfair cipher **iii.** Hill Cipher **iv.** Vigenere cipher
2.	Perform encryption and decryption using following transposition techniques **i.** Rail fence **ii.** Row & Column Transformation
3.	Apply DES algorithm for practical applications.
4.	Apply AES algorithm for practical applications.
5.	Implement RSA Algorithm using HTML and JavaScript
6.	Implement the Diffie-Hellman Key Exchange algorithm for a given problem.
7.	Calculate the message digest of a text using the SHA-1 algorithm.
8.	Implement the SIGNATURE SCHEME - Digital Signature Standard.
9.	Demonstrate intrusion detection system (ids) using any tool eg. Snort or any other s/w.
10.	Automated Attack and Penetration Tools Exploring N-Stalker, a Vulnerability Assessment Tool

11.	Defeating Malware **i.** Building Trojans **ii.** Rootkit Hunter

Ex. No : 1(a) **Date :**	**Encryption and Decryption Using Ceaser Cipher**

AIM:

To encrypt and decrypt the given message by using Ceaser Cipher encryption algorithm.

ALGORITHMS:

1. In Ceaser Cipher each letter in the plaintext is replaced by a letter some fixed number of positions down the alphabet.
2. For example, with a **left shift of 3**, **D** would be replaced by **A**, **E** would become **B**, and so on.
3. The encryption can also be represented using modular arithmetic by first transforming the letters into numbers, according to the scheme, **A = 0, B = 1, Z = 25.** 4. Encryption of a letter x by a shift n can be described mathematically as, ***En(x) = (x + n) mod26*** 5. Decryption is performed similarly,

$$Dn\ (x)=(x - n)\ mod26$$

PROGRAM:

CaesarCipher.java

```
class caesarCipher {
    public static String encode(String enc, int offset)
    {
        offset = offset % 26 + 26;
        StringBuilder encoded = new StringBuilder();
        for (char i : enc.toCharArray()) {
```

```
            if (Character.isLetter(i)) {
                if (Character.isUpperCase(i)) {
                    encoded.append((char) ('A' + (i - 'A' + offset) % 26));
                } else {
                    encoded.append((char) ('a' + (i - 'a' + offset) % 26));
                }
            } else {
                encoded.append(i);
                }
        }
        return encoded.toString();
    }
    public static String decode(String enc, int offset)
    {
        return encode(enc, 26 - offset);
    }
    public static void main(String[] args) throws java.lang.Exception {
        String msg = "Anna University";
        System.out.println("Simulating Caesar Cipher\n-----------------------");
        System.out.println("Input : " + msg);
        System.out.printf("Encrypted Message : ");
        System.out.println(caesarCipher.encode(msg, 3));
        System.out.printf("Decrypted Message : ");
        System.out.println(caesarCipher.decode(caesarCipher.encode(msg, 3), 3));
    }
}
```

OUTPUT:

Simulating Caesar Cipher

Input : Anna University

Encrypted Message : Dqqd Xqlyhuvlwb

Decrypted Message : Anna University

RESULT:

Thus the program for ceaser cipher encryption and decryption algorithm has been implemented and the output verified successfully.

Ex. No : 1(b) Date :	Playfair Cipher

AIM:

To implement a program to encrypt a plain text and decrypt a cipher text using play fair Cipher substitution technique.

ALGORITHM:

1. To encrypt a message, one would break the message into digrams (groups of 2 letters)
2. For example, "HelloWorld" becomes "HE LL OW OR LD".
3. These digrams will be substituted using the key table.
4. Since encryption requires pairs of letters, messages with an odd number of characters usually append an uncommon letter, such as "X", to complete the final digram.
5. The two letters of the digram are considered opposite corners of a rectangle in the key table. To perform the substitution, apply the following 4 rules, in order, to each pair of letters in the plaintext:

PROGRAM:

playfairCipher.java

```
import java.awt.Point;

class playfairCipher
{
    private static char[][] charTable;
    private static Point[] positions;
    private static String prepareText(String s, boolean chgJtoI)
    {
        s = s.toUpperCase().replaceAll("[^A-Z]", "");
        return chgJtoI ? s.replace("J", "I") : s.replace("Q", "");
    }

    private static void createTbl(String key, boolean chgJtoI)
    {
        charTable = new char[5][5];
        positions = new Point[26];
        String s = prepareText(key + "ABCDEFGHIJKLMNOPQRSTUVWXYZ", chgJtoI);
        int len = s.length();
```

```
        for (int i = 0, k = 0; i < len; i++)
        {
            char c = s.charAt(i);
            if (positions[c - 'A'] == null)
            {
                charTable[k / 5][k % 5] = c;
                positions[c - 'A'] = new Point(k % 5, k / 5);
                k++;
            }
        }
    }

    private static String codec(StringBuilder txt, int dir)
    {
        int len = txt.length();
        for (int i = 0; i < len; i += 2)
        {
            char a = txt.charAt(i);
            char b = txt.charAt(i + 1);
            int row1 = positions[a - 'A'].y;
            int row2 = positions[b - 'A'].y;
            int col1 = positions[a - 'A'].x;
            int col2 = positions[b - 'A'].x;
            if (row1 == row2)
            {
                col1 = (col1 + dir) % 5;
                col2 = (col2 + dir) % 5;
            }
            else if (col1 == col2)
            {
                row1 = (row1 + dir) % 5;
                row2 = (row2 + dir) % 5;
            }
            else
            {
                int tmp = col1;
                col1 = col2;
                col2 = tmp;
            }
            txt.setCharAt(i, charTable[row1][col1]);
            txt.setCharAt(i + 1, charTable[row2][col2]);
        }
        return txt.toString();
    }
```

```
    private static String encode(String s)
    {
        StringBuilder sb = new StringBuilder(s);
        for (int i = 0; i < sb.length(); i += 2)
        {
            if (i == sb.length() - 1)
            {
                sb.append(sb.length() % 2 == 1 ? 'X' : "");
            }
            else if (sb.charAt(i) == sb.charAt(i + 1))
            {
                sb.insert(i + 1, 'X');
            }
        }
        return codec(sb, 1);
    }

    private static String decode(String s)
    {
        return codec(new StringBuilder(s), 4);
    }
    public static void main(String[] args) throws java.lang.Exception
    {
        String key = "CSE";
        String txt = "Security Lab"; /* make sure string length is even */ /* cha
nge J to I */
        boolean chgJtoI = true;
        createTbl(key, chgJtoI);
        String enc = encode(prepareText(txt, chgJtoI));
        System.out.println("Simulating Playfair Cipher\n----------------------");
        System.out.println("Input Message : " + txt);
        System.out.println("Encrypted Message : " + enc);
        System.out.println("Decrypted Message : " + decode(enc));
    }
}
```

OUTPUT:

Simulating Playfair Cipher

Input Message : Security Lab

Encrypted Message : EABPUGYANSEZ

Decrypted Message : SECURITYLABX

RESULT:

Thus the program for playfair cipher encryption and decryption algorithm has been implemented and the output verified successfully.

Ex. No : 1(c) Date :	Hill Cipher

AIM:

To implement a program to encrypt and decrypt using the Hill cipher substitution technique

ALGORITHM:

1. In the Hill cipher Each letter is represented by a number modulo 26.
2. To encrypt a message, each block of n letters is multiplied by an invertible ***n x n*** matrix, again ***modulus 26***.
3. To decrypt the message, each block is multiplied by the inverse of the matrix used for encryption.
4. The matrix used for encryption is the cipher key, and it should be chosen randomly from the ***set of invertible n × n matrices (modulo 26).***
5. The cipher can, be adapted to an alphabet with any number of letters.
6. All arithmetic just needs to be done modulo the number of letters instead of modulo 26.

PROGRAM:

HillCipher.java

```
class hillCipher {
    /* 3x3 key matrix for 3 characters at once */
    public static int[][] keymat =
        new int[][] {{1,2,1}, {2,3,2}, {2,2,1}};
        /* key inverse matrix */
    public static int[][] invkeymat =
        new int[][] {{-1, 0, 1}, {2,-1,0}, {-2,2,-1}};

    public static String key = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

    private static String encode(char a, char b, char c) {
        String ret = "";
        int x, y, z;
        int posa = (int) a - 65;
        int posb = (int) b - 65;
        int posc = (int) c - 65;
        x = posa * keymat[0][0] + posb * keymat[1][0] + posc * keymat[2][0];
        y = posa * keymat[0][1] + posb * keymat[1][1] + posc * keymat[2][1];
        z = posa * keymat[0][2] + posb * keymat[1][2] + posc * keymat[2][2];
```

```
        a = key.charAt(x % 26);
        b = key.charAt(y % 26);
        c = key.charAt(z % 26);
        ret = "" + a + b + c;
        return ret;
    }

    private static String decode(char a, char b, char c) {
        String ret = "";
        int x, y, z;
        int posa = (int) a - 65;
        int posb = (int) b - 65;
        int posc = (int) c - 65;
        x = posa * invkeymat[0][0] + posb * invkeymat[1][0] + posc
                                                    * invkeymat[2][0];
        y = posa * invkeymat[0][1] + posb * invkeymat[1][1] + posc
                                                    * invkeymat[2][1];
        z = posa * invkeymat[0][2] + posb * invkeymat[1][2] + posc
                                                    * invkeymat[2][2];
        a = key.charAt((x % 26 < 0) ? (26 + x % 26) : (x % 26));
        b = key.charAt((y % 26 < 0) ? (26 + y % 26) : (y % 26));
        c = key.charAt((z % 26 < 0) ? (26 + z % 26) : (z % 26));
        ret = "" + a + b + c;
        return ret;
    }

    public static void main(String[] args) throws java.lang.Exception {
        String msg;
        String enc = "";
        String dec = "";
        int n;
        msg = ("SecurityLaboratory");
        System.out.println("simulation of Hill Cipher\n----------------------");
        System.out.println("Input message : " + msg);
        msg = msg.toUpperCase();
        msg = msg.replaceAll("\\s", "");
        /* remove spaces */
        n = msg.length() % 3;
        /* append padding text X */
        if (n != 0) {
            for (int i = 1; i <= (3 - n); i++) {
                msg += 'X';
            }
        }
        System.out.println("padded message : " + msg);
```

```
        char[] pdchars = msg.toCharArray();
        for (int i = 0; i < msg.length(); i += 3) {
            enc += encode(pdchars[i], pdchars[i + 1], pdchars[i + 2]);
        }
        System.out.println("encoded message : " + enc);
        char[] dechars = enc.toCharArray();
        for (int i = 0; i < enc.length(); i += 3) {
            dec += decode(dechars[i], dechars[i + 1], dechars[i + 2]);
        }
        System.out.println("decoded message : " + dec);
    }
}
```

OUTPUT:

Simulating Hill Cipher

Input Message : SecurityLaboratory

Padded Message : SECURITYLABORATORY

Encrypted Message : EACSDKLCAEFQDUKSXU

Decrypted Message : SECURITYLABORATORY

RESULT:

Thus the program for hill cipher encryption and decryption algorithm has been implemented and the output verified successfully.

Ex. No : 1(d) Date :	Vigenere Cipher

AIM:

To implement a program for encryption and decryption using vigenere cipher substitution technique

ALGORITHM:

1. The Vigenere cipher is a method of encrypting alphabetic text by using a series of different Caesar ciphers based on the letters of a keyword.
2. It is a simple form of *polyalphabetic* substitution.
3. To encrypt, a table of alphabets can be used, termed a Vigenere square, or Vigenere table.
4. It consists of the alphabet written out 26 times in different rows, each alphabet shifted cyclically to the left compared to the previous alphabet, corresponding to the 26 possible Caesar ciphers.
5. At different points in the encryption process, the cipher uses a different alphabet from one of the rows used.
6. The alphabet at each point depends on a repeating keyword.

PROGRAM:

vigenereCipher.java

```
public class vigenereCipher {
    static String encode(String text, final String key) {
        String res = "";
        text = text.toUpperCase();
        for (int i = 0, j = 0; i < text.length(); i++) {
            char c = text.charAt(i);
            if (c < 'A' || c > 'Z') {
                continue;
            }
            res += (char)((c + key.charAt(j) - 2 * 'A') % 26 + 'A');
            j = ++j % key.length();
        }
        return res;
    }

    static String decode(String text, final String key) {
        String res = "";
        text = text.toUpperCase();
```

```
        for (int i = 0, j = 0; i < text.length(); i++) {
            char c = text.charAt(i);
            if (c < 'A' || c > 'Z') {
                continue;
            }
            res += (char)((c - key.charAt(j) + 26) % 26 + 'A');
            j = ++j % key.length();
        }
        return res;
    }

    public static void main(String[] args) throws java.lang.Exception {
        String key = "VIGENERECIPHER";
        String msg = "SecurityLaboratory";
        System.out.println("Simulating Vigenere Cipher\n----------------------");
        System.out.println("Input Message : " + msg);
        String enc = encode(msg, key);
        System.out.println("Encrypted Message : " + enc);
        System.out.println("Decrypted Message : " + decode(enc, key));
    }
}
```

OUTPUT:

Simulating Vigenere Cipher

Input Message : SecurityLaboratory

Encrypted Message : NMIYEMKCNIQVVROWXC

Decrypted Message : SECURITYLABORATORY

RESULT:

Thus the program for vigenere cipher encryption and decryption algorithm has been implemented and the output verified successfully.

Ex. No : 2(a) Date :	Rail Fence Cipher Transposition Technique

AIM:

To implement a program for encryption and decryption using rail fence transposition technique.

ALGORITHM:

1. In the rail fence cipher, the plaintext is written downwards and diagonally on successive "rails" of an imaginary fence, then moving up when we reach the bottom rail.
2. When we reach the top rail, the message is written downwards again until the whole plaintext is written out.
3. The message is then read off in rows.

PROGRAM:

railFenceCipher.java

```
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 msg, enc, dec;
        msg = "Anna University, Chennai";
        int depth = 2;
        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.printf("Decrypted Message : " + dec);
    }
}
```

OUTPUT:

Simulating Railfence Cipher

Input Message : Anna University, Chennai
Encrypted Message : An nvriy hnanaUiest,Ceni
Decrypted Message : Anna University, Chennai

RESULT:

Thus the java program for Rail Fence Transposition Technique has been implemented and the output verified successfully.

Ex. No : 2(b) Date :	Row and Column Transformation Technique

AIM:

To implement a program for encryption and decryption by using row and column transformation technique.

ALGORITHM:

1. Consider the plain text hello world, and let us apply the simple columnar transposition technique as shown below

h	e	l	l
o	w	o	r
l	d		

2. The plain text characters are placed horizontally and the cipher text is created with vertical format as: **holewdlo lr**.
3. Now, the receiver has to use the same table to decrypt the cipher text to plain text.

PROGRAM:

TransCipher.java

```
import java.util.*;
class TransCipher {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the plain text");
        String pl = sc.nextLine();
        sc.close();
        String s = "";
        int start = 0;
        for (int i = 0; i < pl.length(); i++) {
            if (pl.charAt(i) == ' ') {
                s = s + pl.substring(start, i);
                start = i + 1;
            }
        }
        s = s + pl.substring(start);
```

```
        System.out.print(s);
        System.out.println();
        // end of space deletion

        int k = s.length();
        int l = 0;
        int col = 4;
        int row = s.length() / col;
        char ch[][] = new char[row][col];
        for (int i = 0; i < row; i++) {
            for (int j = 0; j < col; j++) {
                if (l < k) {
                    ch[i][j] = s.charAt(l);
                    l++;
                } else {
                    ch[i][j] = '#';
                }
            }
        }
        // arranged in matrix

        char trans[][] = new char[col][row];
        for (int i = 0; i < row; i++) {
            for (int j = 0; j < col; j++) {
                trans[j][i] = ch[i][j];
            }
        }

        for (int i = 0; i < col; i++) {
            for (int j = 0; j < row; j++) {
                System.out.print(trans[i][j]);
            }
        }
        // display
        System.out.println();
    }
}
```

OUTPUT:

Enter the plain text
Security Lab
SecurityLab
Sreictuy

RESULT:

Thus the java program for Row and Column Transposition Technique has been implemented and the output verified successfully.

Ex. No : 3 Date :	Data Encryption Standard (DES) Algorithm (User Message Encryption)

AIM:

To use Data Encryption Standard (DES) Algorithm for a practical application like User Message Encryption.

ALGORITHM:

1. Create a DES Key.
2. Create a Cipher instance from Cipher class, specify the following information and separated by a slash (/).
   a. Algorithm name
   b. Mode (optional)
   c. Padding scheme (optional)
3. Convert String into ***Byte[]*** array format.
4. Make Cipher in encrypt mode, and encrypt it with ***Cipher.doFinal()*** method.
5. Make Cipher in decrypt mode, and decrypt it with ***Cipher.doFinal()*** method.

PROGRAM:

DES.java

```
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;

import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.KeyGenerator;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.SecretKey;

public class DES {
    public static void main(String[] argv) {

        try {
            System.out.println("Message Encryption Using DES Algorithm\n-------
");
            KeyGenerator keygenerator = KeyGenerator.getInstance("DES");
            SecretKey myDesKey = keygenerator.generateKey();
            Cipher desCipher;
```

```
            desCipher = Cipher.getInstance("DES/ECB/PKCS5Padding");
            desCipher.init(Cipher.ENCRYPT_MODE, myDesKey);
            byte[] text = "Secret Information ".getBytes();
            System.out.println("Message [Byte Format] : " + text);
            System.out.println("Message : " + new String(text));
            byte[] textEncrypted = desCipher.doFinal(text);
            System.out.println("Encrypted Message: " + textEncrypted);
            desCipher.init(Cipher.DECRYPT_MODE, myDesKey);
            byte[] textDecrypted = desCipher.doFinal(textEncrypted);
            System.out.println("Decrypted Message: " + new String(textDecrypted))
;

        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        } catch (NoSuchPaddingException e) {
            e.printStackTrace();
        } catch (InvalidKeyException e) {
            e.printStackTrace();
        } catch (IllegalBlockSizeException e) {
            e.printStackTrace();
        } catch (BadPaddingException e) {
            e.printStackTrace();
        }

    }
}
```

OUTPUT:

Message Encryption Using DES Algorithm

Message [Byte Format] : [B@4dcbadb4
Message : Secret Information
Encrypted Message: [B@504bae78
Decrypted Message: Secret Information

RESULT:

Thus the java program for DES Algorithm has been implemented and the output verified successfully.

Ex. No : 4 Date :	Advanced Encryption Standard (AES) Algorithm (URL Encryption)

AIM:

To use Advanced Encryption Standard (AES) Algorithm for a practical application like URL Encryption.

ALGORITHM:

1. AES is based on a design principle known as a substitution–permutation.
2. AES does not use a Feistel network like DES, it uses variant of Rijndael.
3. It has a fixed block size of 128 bits, and a key size of 128, 192, or 256 bits.
4. AES operates on a 4 × 4 column-major order array of bytes, termed the state

PROGRAM:

AES.java

```
import java.io.UnsupportedEncodingException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Arrays;
import java.util.Base64;

import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;

public class AES {

    private static SecretKeySpec secretKey;
    private static byte[] key;

    public static void setKey(String myKey) {
        MessageDigest sha = null;
        try {
            key = myKey.getBytes("UTF-8");
            sha = MessageDigest.getInstance("SHA-1");
            key = sha.digest(key);
            key = Arrays.copyOf(key, 16);
            secretKey = new SecretKeySpec(key, "AES");
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
```

```
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }
    }

    public static String encrypt(String strToEncrypt, String secret) {
        try {
            setKey(secret);
            Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
            cipher.init(Cipher.ENCRYPT_MODE, secretKey);
            return Base64.getEncoder().encodeToString(cipher
                .doFinal(strToEncrypt.getBytes("UTF -8 ")));
        } catch (Exception e) {
            System.out.println("Error while encrypting: " + e.toString());
        }
        return null;
    }

    public static String decrypt(String strToDecrypt, String secret) {
        try {
            setKey(secret);
            Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5PADDING");
            cipher.init(Cipher.DECRYPT_MODE, secretKey);
            return new String(cipher.doFinal(Base64.getDecoder().
            decode(strToDecrypt)));
        } catch (Exception e) {
            System.out.println("Error while decrypting: " + e.toString());
        }
        return null;
    }

    public static void main(String[] args) {
        final String secretKey = "annaUniversity";
        String originalString = "www.annauniv.edu";
        String encryptedString =
            AES.encrypt(originalString, secretKey);
        String decryptedString =
            AES.decrypt(encryptedString, secretKey);
        System.out.println("URL Encryption Using AES Algorithm\n------------");
        System.out.println("Original URL : " + originalString);
        System.out.println("Encrypted URL : " + encryptedString);
        System.out.println("Decrypted URL : " + decryptedString);
    }
}
```

OUTPUT:

URL Encryption Using AES Algorithm

Original URL : www.annauniv.edu

Encrypted URL : vibpFJW6Cvs5Y+L7t4N6YWWe07+JzS1d3CU2h3mEvEg=

Decrypted URL : www.annauniv.edu

RESULT:

Thus the java program for AES Algorithm has been implemented for URL Encryption and the output verified successfully.

Ex. No : 5 Date :	RSA Algorithm

AIM:

To implement RSA (Rivest–Shamir–Adleman) algorithm by using HTML and Javascript.

ALGORITHM:

1. Choose two prime number p and q
2. Compute the value of n and **p**
3. Find the value of ***e*** (public key)
4. Compute the value of ***d*** (private key) using gcd()
5. Do the encryption and decryption
   a. Encryption is given as,
   $c = t^e \bmod n$
   b. Decryption is given as,
   $t = c^d \bmod n$

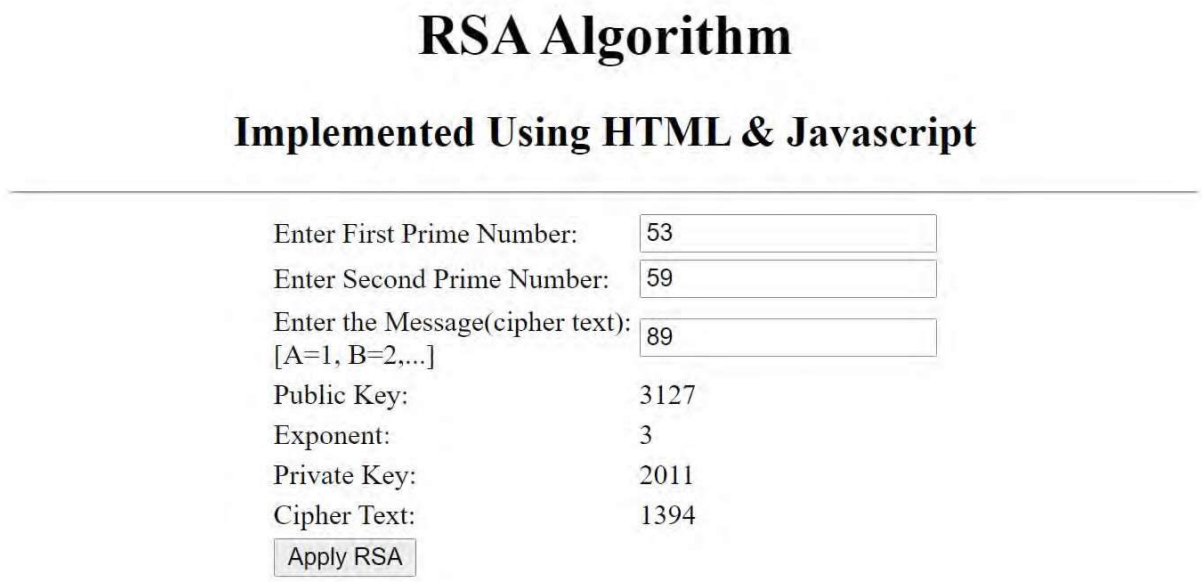
PROGRAM: ***rsa.html***

```
<html>
   <head>
      <title>RSA Encryption</title>
      <meta name="viewport" content="width=device-width, initial-scale=1.0">
   </head>
   <body>
      <center>
         <h1>RSA Algorithm</h1>
         <h2>Implemented Using HTML & Javascript</h2>
         <hr>
         <table>
            <tr>
               <td>Enter First Prime Number:</td>
               <td><input type="number" value="53" id="p"></td>
            </tr>
            <tr>
               <td>Enter Second Prime Number:</td>
               <td><input type="number" value="59" id="q"></p>
               </td>
            </tr>
            <tr>
```

```
            <td>Enter the Message(cipher text):<br>[A=1, B=2,...]</td>
            <td><input type="number" value="89" id="msg"></p>
            </td>
         </tr>
         <tr>
            <td>Public Key:</td>
            <td>
               <p id="publickey"></p>
            </td>
         </tr>
         <tr>
            <td>Exponent:</td>
            <td>
               <p id="exponent"></p>
            </td>
         </tr>
         <tr>
            <td>Private Key:</td>
            <td>
               <p id="privatekey"></p>
            </td>
         </tr>
         <tr>
            <td>Cipher Text:</td>
            <td>
               <p id="ciphertext"></p>
            </td>
         </tr>
         <tr>
            <td><button onclick="RSA();">Apply RSA</button></td>
         </tr>
      </table>
   </center>
</body>
<script type="text/javascript">
     function RSA()
     {
         var gcd, p, q, no, n, t, e, i, x;
         gcd = function (a, b) {
             return (!b) ? a : gcd(b, a % b);
         };
         p = document.getElementById('p').value;
         q = document.getElementById('q').value;
         no = document.getElementById('msg').value;
         n = p * q;
```

```
            t = (p - 1) * (q - 1);

            for (e = 2; e < t; e++) {
                if (gcd(e, t) == 1) {
                    break;
                }
            }

            for (i = 0; i < 10; i++) {
                x = 1 + i * t
                if (x % e == 0) {
                    d = x / e;
                    break;
                }
            }

            ctt = Math.pow(no, e).toFixed(0);
            ct = ctt % n;

            dtt = Math.pow(ct, d).toFixed(0);
            dt = dtt % n;

            document.getElementById('publickey').innerHTML = n;
            document.getElementById('exponent').innerHTML = e;
            document.getElementById('privatekey').innerHTML = d;
            document.getElementById('ciphertext').innerHTML = ct;
        }
    </script>
</html>
```

OUTPUT:



RESULT:

Thus the RSA algorithm has been implemented using HTML & CSS and the output has been verified successfully.

Ex. No : 6 Date :	Diffie-Hellman key exchange algorithm

AIM:

To implement the Diffie-Hellman Key Exchange algorithm for a given problem .

ALGORITHM:

1. Alice and Bob publicly agree to use a modulus $p = 23$ and base $g = 5$ (which is a primitive root modulo 23).
2. Alice chooses a secret integer $a = 4$, then sends Bob $A = g^a \bmod p$ ∘ $A = 5^4 \bmod 23 = 4$
3. Bob chooses a secret integer $b = 3$, then sends Alice $B = g^b \bmod p$ ∘ $B = 5^3 \bmod 23 = 10$
4. Alice computes $s = B^a \bmod p$ ∘ $s = 10^4 \bmod 23 = 18$
5. Bob computes $s = A^b \bmod p$ ∘ $s = 4^3 \bmod 23 = 18$ 6. Alice and Bob now share a secret (the number 18).

PROGRAM:

DiffieHellman.java

```
class DiffieHellman {
    public static void main(String args[]) {
        int p = 23; /* publicly known (prime number) */
        int g = 5; /* publicly known (primitive root) */
        int x = 4; /* only Alice knows this secret */
        int y = 3; /* only Bob knows this secret */
        double aliceSends = (Math.pow(g, x)) % p;
        double bobComputes = (Math.pow(aliceSends, y)) % p;
        double bobSends = (Math.pow(g, y)) % p;
        double aliceComputes = (Math.pow(bobSends, x)) % p;
        double sharedSecret = (Math.pow(g, (x * y))) % p;
        System.out.println
        ("simulation of Diffie-Hellman key exchange algorithm\n---------------");
        System.out.println("Alice Sends : " + aliceSends);
        System.out.println("Bob Computes : " + bobComputes);
        System.out.println("Bob Sends : " + bobSends);
        System.out.println("Alice Computes : " + aliceComputes);
```

```
        System.out.println("Shared Secret : " + sharedSecret);
        /* shared secrets should match and equality is transitive */
        if ((aliceComputes == sharedSecret) && (aliceComputes == bobComputes))
            System.out.println("Success: Shared Secrets Matches!"+sharedSecret);
        else
            System.out.println("Error: Shared Secrets does not Match");
    }
}
```

OUTPUT:

simulation of Diffie-Hellman key exchange algorithm
---------- Alice
Sends : 4.0
Bob Computes : 18.0
Bob Sends : 10.0
Alice Computes : 18.0
Shared Secret : 18.0
Success: Shared Secrets Matches! 18.0

RESULT:

Thus the *Diffie-Hellman key exchange algorithm* has been implemented using Java Program and the output has been verified successfully.

Ex. No : 7 Date :	SHA-1 Algorithm

AIM:

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

ALGORITHM:

1. Append Padding Bits
2. Append Length - 64 bits are appended to the end
3. Prepare Processing Functions
4. Prepare Processing Constants
5. Initialize Buffers
6. Processing Message in 512-bit blocks (L blocks in total message)

PROGRAM: ***sha1.java***

```
import java.security.*;

public class sha1 {
    public static void main(String[] a) {
        try {
            MessageDigest md = MessageDigest.getInstance("SHA1");
            System.out.println("Message digest object info:\n----------------");
            System.out.println("Algorithm=" + md.getAlgorithm());
            System.out.println("Provider=" + md.getProvider());
            System.out.println("ToString=" + md.toString());
            String input = "";
            md.update(input.getBytes());
            byte[] output = md.digest();
            System.out.println();
            System.out.println("SHA1(\"" + input + "\")=" + bytesToHex(output));
            input = "abc";
            md.update(input.getBytes());
            output = md.digest();
            System.out.println();
            System.out.println("SHA1(\"" + input + "\")=" + bytesToHex(output));
            input = "abcdefghijklmnopqrstuvwxyz";
            md.update(input.getBytes());
```

```
            output = md.digest();
            System.out.println();
            System.out.println("SHA1(\"" + input + "\")=" + bytesToHex(output));
            System.out.println();
        } catch (Exception e) {
            System.out.println("Exception:" + e);
        }
    }

    private static String bytesToHex(byte[] b) {
        char hexDigit[] =
        {
            '0','1','2','3','4','5',
            '6','7','8','9','A','B',
            'C','D','E','F'
        };
        StringBuffer buf = new StringBuffer();

        for (byte aB: b) {
            buf.append(hexDigit[(aB >> 4) & 0x0f]);
            buf.append(hexDigit[aB & 0x0f]);
        }

        return buf.toString();
    }
}
```

OUTPUT:

Message digest object info:

Algorithm=SHA1
Provider=SUN version 12
ToString=SHA1 Message Digest from SUN, <initialized>

SHA1("")=DA39A3EE5E6B4B0D3255BFEF95601890AFD80709
SHA1("abc")=A9993E364706816ABA3E25717850C26C9CD0D89D
SHA1("abcdefghijklmnopqrstuvwxyz")=32D10C7B8CF96570CA04CE37F2A19
D84240D3A89

RESULT:

Thus the *Secure Hash Algorithm (SHA-1)* has been implemented and the output has been verified successfully.

Ex. No : 8 Date :	Digital Signature Standard

AIM: To implement the SIGNATURE SCHEME - Digital Signature Standard.

ALGORITHM:

1. Create a KeyPairGenerator object.
2. Initialize the KeyPairGenerator object.
3. Generate the KeyPairGenerator. ...
4. Get the private key from the pair.
5. Create a signature object.
6. Initialize the Signature object.
7. Add data to the Signature object
8. Calculate the Signature

PROGRAM:

```
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.PrivateKey;
import java.security.Signature;
import java.util.Scanner;

public class CreatingDigitalSignature {
    public static void main(String args[]) throws Exception {

        Scanner sc = new Scanner(System.in);
        System.out.println("Enter some text");
        String msg = sc.nextLine();

        KeyPairGenerator keyPairGen = KeyPairGenerator.getInstance("DSA");

        keyPairGen.initialize(2048);

        KeyPair pair = keyPairGen.generateKeyPair();

        PrivateKey privKey = pair.getPrivate();
```

```
        Signature sign = Signature.getInstance("SHA256withDSA");
        sign.initSign(privKey);
        byte[] bytes = "msg".getBytes();

        sign.update(bytes);

        byte[] signature = sign.sign();

        System.out.println
        ("Digital signature for given text: " + new String(signature,"UTF8"));
    }
}
```

OUTPUT:

Enter some text
Hi how are you
Digital signature for given text: 0=@gRD???-?.???? /yGL?i??a!?

RESULT:

Thus the Digital Signature Standard Signature Scheme has been implemented and the output has been verified successfully.

Ex. No : 9 Date :	Demonstration of Intrusion Detection System(IDS)

AIM:

To demonstrate Intrusion Detection System (IDS) using Snort software tool.

STEPS ON CONFIGURING AND INTRUSION DETECTION:

1. Download Snort from the Snort.org website. (http://www.snort.org/snortdownloads)

2. Download Rules(https://www.snort.org/snort-rules). You must register to get the rules. (You should download these often)

3. Double click on the .exe to install snort. This will install snort in the "C:\Snort" folder.It is important to have WinPcap (https://www.winpcap.org/install/) installed

4. Extract the Rules file. You will need WinRAR for the .gz file.

5. Copy all files from the "rules" folder of the extracted folder. Now paste the rules into *"C:\Snort\rules"* folder.

6. Copy "snort.conf" file from the "etc" folder of the extracted folder. You must paste it into "C:\Snort\etc" folder. Overwrite any existing file. Remember if you modify your snort.conf file and download a new file, you must modify it for Snort to work.

7. Open a command prompt (cmd.exe) and navigate to folder "C:\Snort\bin" folder. (at the Prompt, type cd\snort\bin)

8. To start (execute) snort in sniffer mode use following command:

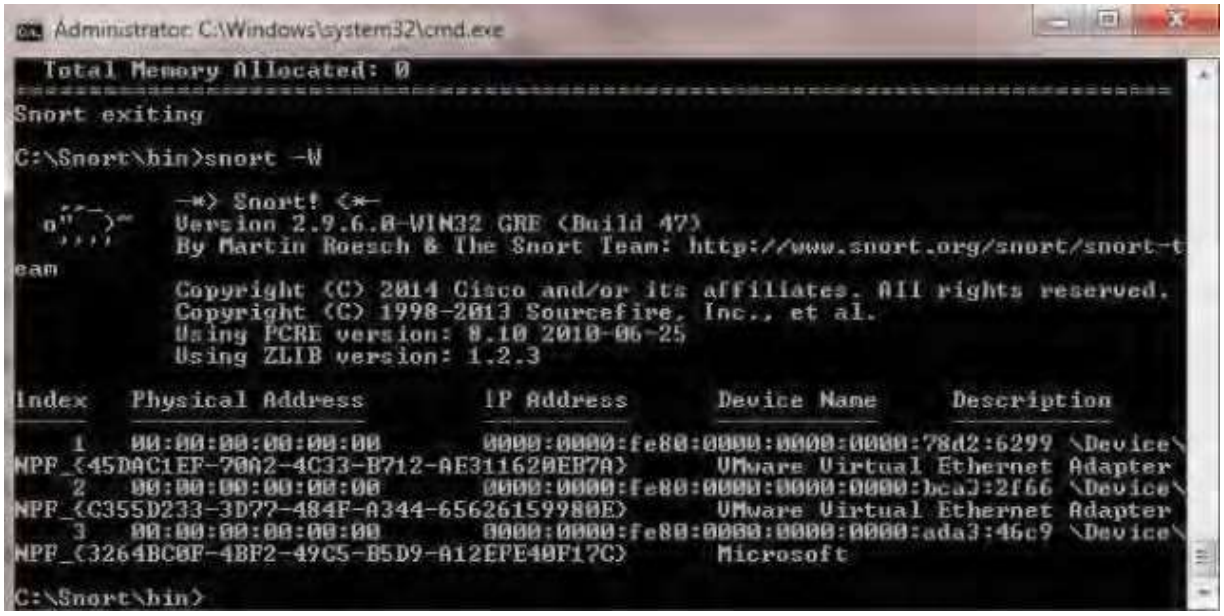
snort -dev -i 3

-i indicates the interface number. You must pick the correct interface number. In my case, it is 3.

-dev is used to run snort to capture packets on your network.

To check the interface list, use following command:

snort -W



Finding an interface

You can tell which interface to use by looking at the Index number and finding Microsoft. As you can see in the above example, the other interfaces are for VMWare. My interface is 3.

9. To run snort in IDS mode, you will need to configure the file "snort.conf" according to your network environment.
10. To specify the network address that you want to protect in snort.conf file, look for the following line.

var HOME_NET 192.168.1.0/24 (You will normally see any here)

11. You may also want to set the addresses of DNS_SERVERS, if you have some on your network.

Example:

example snort

12. Change the RULE_PATH variable to the path of rules folder.

var RULE_PATH c:\snort\rules

path to rules

13. Change the path of all library files with the name and path on your system. and you must change the path of snort_dynamicpreprocessorvariable.

C:\Snort\lib\snort_dynamiccpreprocessor

You need to do this to all library files in the "C:\Snort\lib" folder. The old path might be: "/usr/local/lib/…". you will need to replace that path with your system path. Using C:\Snort\lib

14. Change the path of the "dynamicengine" variable value in the "snort.conf" file..

Example:

dynamicengine C:\Snort\lib\snort_dynamicengine\sf_engine.dll

15 Add the paths for "include classification.config" and "include reference.config" files.

include c:\snort\etc\classification.config include c:\snort\etc\reference.config

16. Remove the comment (#) on the line to allow ICMP rules, if it is commented with a #.

include $RULE_PATH/icmp.rules

17. You can also remove the comment of ICMP-info rules comment, if it is commented.

include $RULE_PATH/icmp-info.rules

18. To add log files to store alerts generated by snort, search for the "output log" test in snort.conf and add the following line:

output alert_fast: snort-alerts.ids

19. Comment (add a #) the whitelist $WHITE_LIST_PATH/white_list.rules and the blacklist

Change the nested_ip inner , \ to nested_ip inner #, \

20. Comment out (#) following lines:

#preprocessor normalize_ip4
#preprocessor normalize_tcp: ips ecn stream
#preprocessor normalize_icmp4 #preprocessor normalize_ip6
#preprocessor normalize_icmp6

21. Save the "snort.conf" file.
22. To start snort in IDS mode, run the following command:

snort -c c:\snort\etc\snort.conf -l c:\snort\log -i 3
(Note: 3 is used for my interface card)

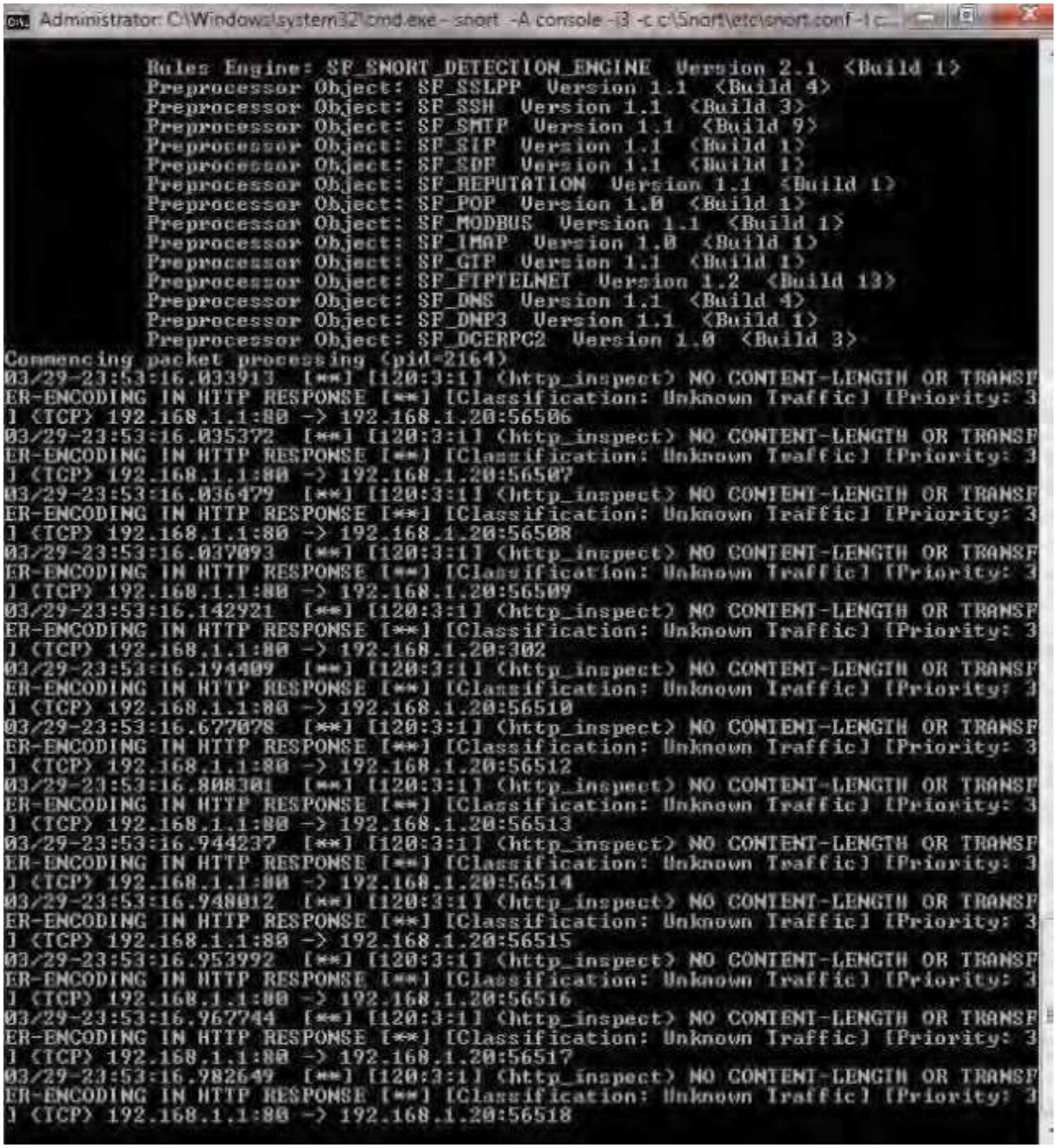
If a log is created, select the appropriate program to open it. You can use WordPard or NotePad++ to read the file.

To generate Log files in ASCII mode, you can use following command while running snort in IDS mode:
snort -A console -i3 -c c:\Snort\etc\snort.conf -l c:\Snort\log -K ascii

23. Scan the computer that is running snort from another computer by using PING or NMap (ZenMap).

After scanning or during the scan you can check the snort-alerts.ids file in the log folder to insure it is logging properly. You will see IP address folders appear.

Snort monitoring traffic –



RESULT:

Thus the Intrusion Detection System(IDS) has been demonstrated by using the Open Source Snort Intrusion Detection Tool.

Ex. No : 10 Date :	Exploring N-Stalker, a Vulnerability Assessment Tool

AIM:

To download the N-Stalker Vulnerability Assessment Tool and exploring the features.

EXPLORING N-STALKER:

- N-Stalker Web Application Security Scanner is a Web security assessment tool.
- It incorporates with a well-known N-Stealth HTTP Security Scanner and 35,000 Web attack signature database.
- This tool also comes in both free and paid version.
- Before scanning the target, go to "License Manager" tab, perform the update.
- Once update, you will note the status as up to date.
- You need to download and install N-Stalker from www.nstalker.com.

1. Start N-Stalker from a Windows computer. The program is installed under Start ➪ Programs ➪ N-Stalker ➪ N-Stalker Free Edition.
2. Enter a host address or a range of addresses to scan.
3. Click Start Scan.
4. After the scan completes, the N-Stalker Report Manager will prompt 5. you to select a format for the resulting report as choose Generate HTML.
6. Review the HTML report for vulnerabilities.

Now goto “Scan Session”, enter the target URL.

In scan policy, you can select from the four options,

- Manual test which will crawl the website and will be waiting for manual attacks.
- full xss assessment
- owasp policy
- Web server infrastructure analysis.

Once, the option has been selected, next step is “Optimize settings” which will crawl the whole website for further analysis.

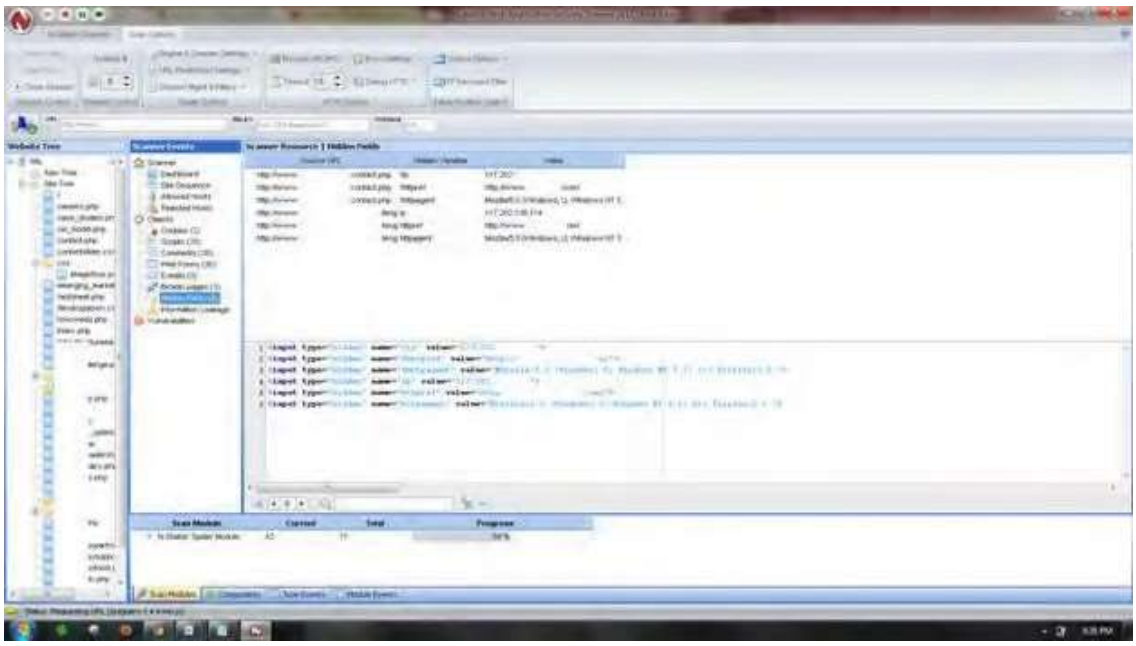
In review option, you can get all the information like host information, technologies used, policy name, etc.


N-Stalker Scan Wizard
Start Web Application Security Scan Session
You must enter an URL and choose policy. Scan Settings may be configured.
Enter Web Application URL
www.target.com
(E.g: http://www.example.tl/, https://www.test.tl/VirtualDirectory/, etc)
Choose Scan Policy
(choose one)
Load Scan Session
(choose one)
(You may load scan settings from previously saved scan sessions)
Load Spider Data
(You may load spider data from previously saved scan sessions)
Choose URL & Policy
Optimize Settings
Review Summary
Start Scan Session
Scan Settings
Cancel
Next >>

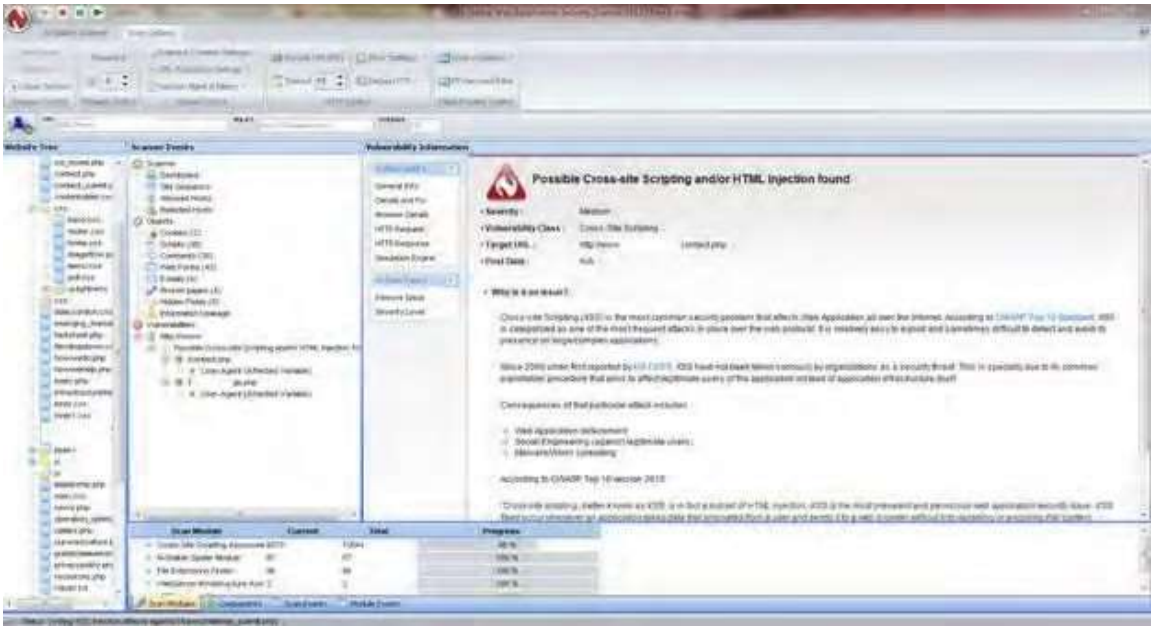

N-Stalker Scan Wizard
Start Web Application Security Scan Session
You must enter an URL and choose policy. Scan Settings may be configured.
Review Summary
Scanning Settings
Scan Setting
Value
Host Information
Restricted Directory
Not configured.
Policy Name
Spider Only
False-Positive Settings
New Server Discovery
Enabled (recommended in most cases)
Spider Engine
HTML Parser
Server Technologies
N/A
Allowed Hosts
No additional hosts configured.
Choose URL & Policy
Optimize Settings
Review Summary
Start Scan Session
Scan Settings
<< Back
Cancel
Start Session

Once done, start the session and start the scan.

The scanner will crawl the whole website and will show the scripts, broken pages, hidden fields, information leakage, web forms related information which helps to analyze further.



Once the scan is completed, the NStalker scanner will show details like severity level, vulnerability class, why is it an issue, the fix for the issue and the URL which is vulnerable to the particular vulnerability?



RESULT:

Thus the N-Stalker Vulnerability Assessment tool has been downloaded, installed and the features has been explored by using a vulnerable website.

Ex. No : 11(a) Date :	Defeating Malware - Building Trojans

AIM:

To build a Trojan and know the harmness of the trojan malwares in a computer system.

PROCEDURE:

1. Create a simple trojan by using Windows Batch File (***.bat***) 2.
Type these below code in notepad and save it as **Trojan.bat**
3. Double click on ***Trojan.bat*** file.
4. When the trojan code executes, it will open MS-Paint, Notepad, Command Prompt, Explorer, etc., infinitely. 5. Restart the computer to stop the execution of this trojan.

TROJAN:

- In computing, a Trojan horse,or trojan, is any malware which misleads users of its true intent.

- Trojans are generally spread by some form of social engineering, for example where a user is duped into executing an email attachment disguised to appear not suspicious, (e.g., a routine form to be filled in), or by clicking on some fake advertisement on social media or anywhere else.

- Although their payload can be anything, many modern forms act as a backdoor, contacting a controller which can then have unauthorized access to the affected computer.
- Trojans may allow an attacker to access users' personal information such as banking information, passwords, or personal identity.

- ***Example:*** *Ransomware* attacks are often carried out using a *trojan*.

CODE:

Trojan.bat

```
@echo off
:x start
mspaint start
notepad start
cmd start
explorer
start control
start calc
goto x
```

OUTPUT

(MS-Paint, Notepad, Command Prompt, Explorer will open infinitely)

RESULT:

Thus a trojan has been built and the harmness of the trojan viruses has been explored.

Ex. No : 11(b) Date :	Defeating Malware - Rootkit hunter

AIM: To install a rootkit hunter and find the malwares in a computer.

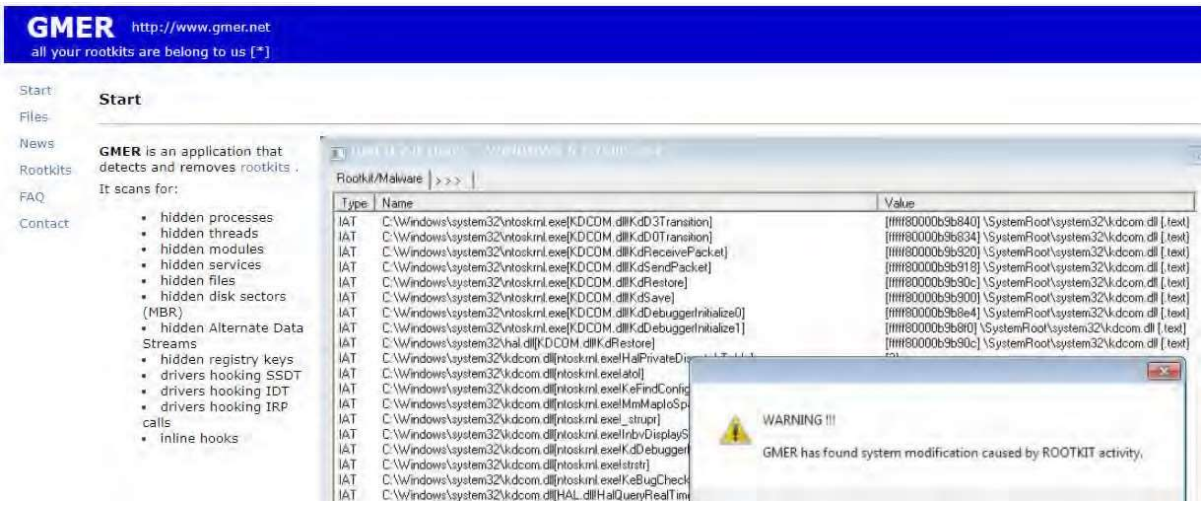
ROOTKIT HUNTER:

- rkhunter (Rootkit Hunter) is a Unix-based tool that scans for rootkits, backdoors and possible local exploits.
- It does this by comparing SHA-1 hashes of important files with known good ones in online databases, searching for default directories (of rootkits), wrong permissions, hidden files, suspicious strings in kernel modules, and special tests for Linux and FreeBSD.
- rkhunter is notable due to its inclusion in popular operating systems (Fedora, Debian, etc.)
- The tool has been written in Bourne shell, to allow for portability. It can run on almost all UNIX-derived systems.

GMER ROOTKIT TOOL:

- GMER is a software tool written by a Polish researcher Przemysław Gmerek, for detecting and removing rootkits.
- It runs on Microsoft Windows and has support for Windows NT, 2000, XP, Vista, 7, 8 and 10. With version 2.0.18327 full support for Windows x64 is added.

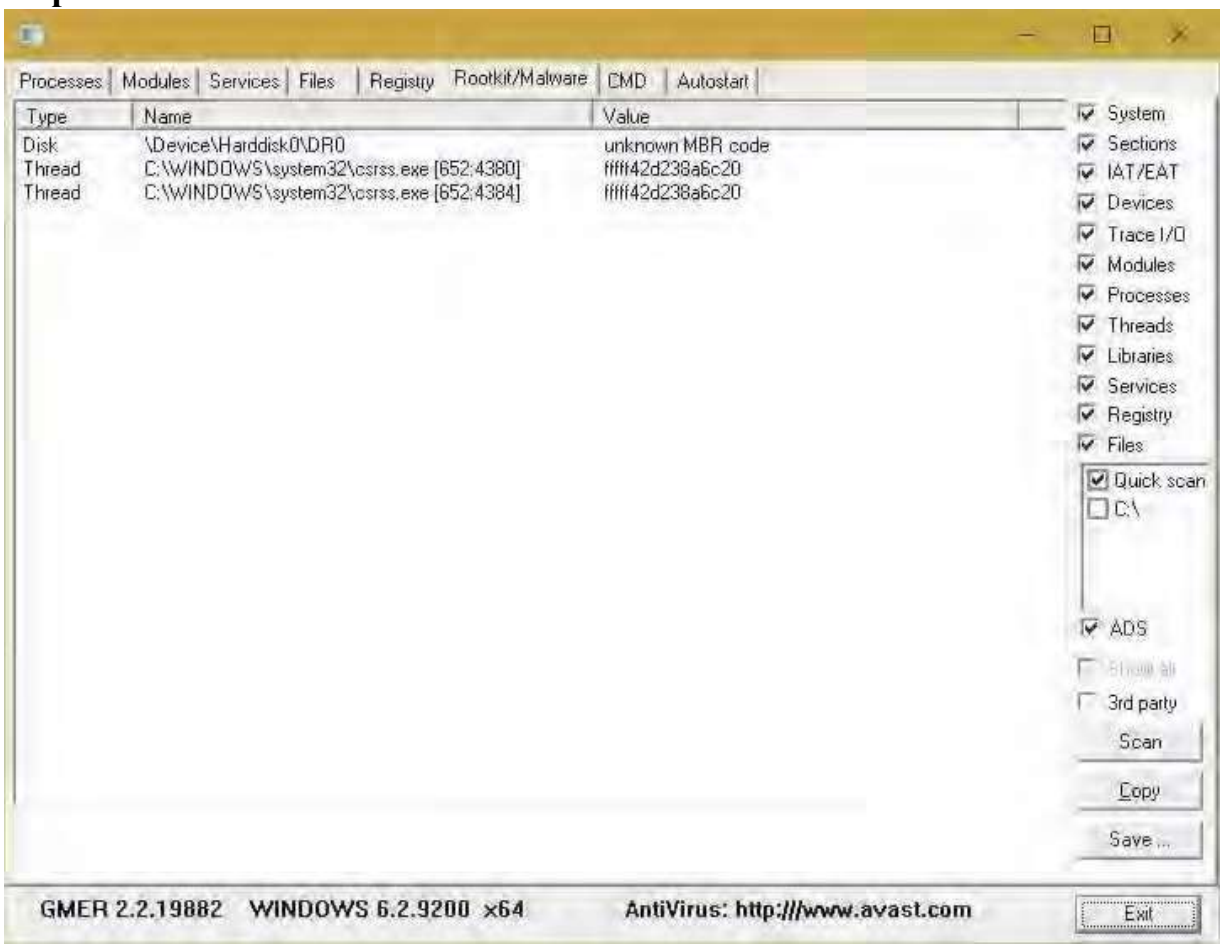
Step 1



Visit GMER's website (see Resources) and download the GMER executable.

Click the "Download EXE" button to download the program with a random file name, as some rootkits will close “gmer.exe” before you can open it.

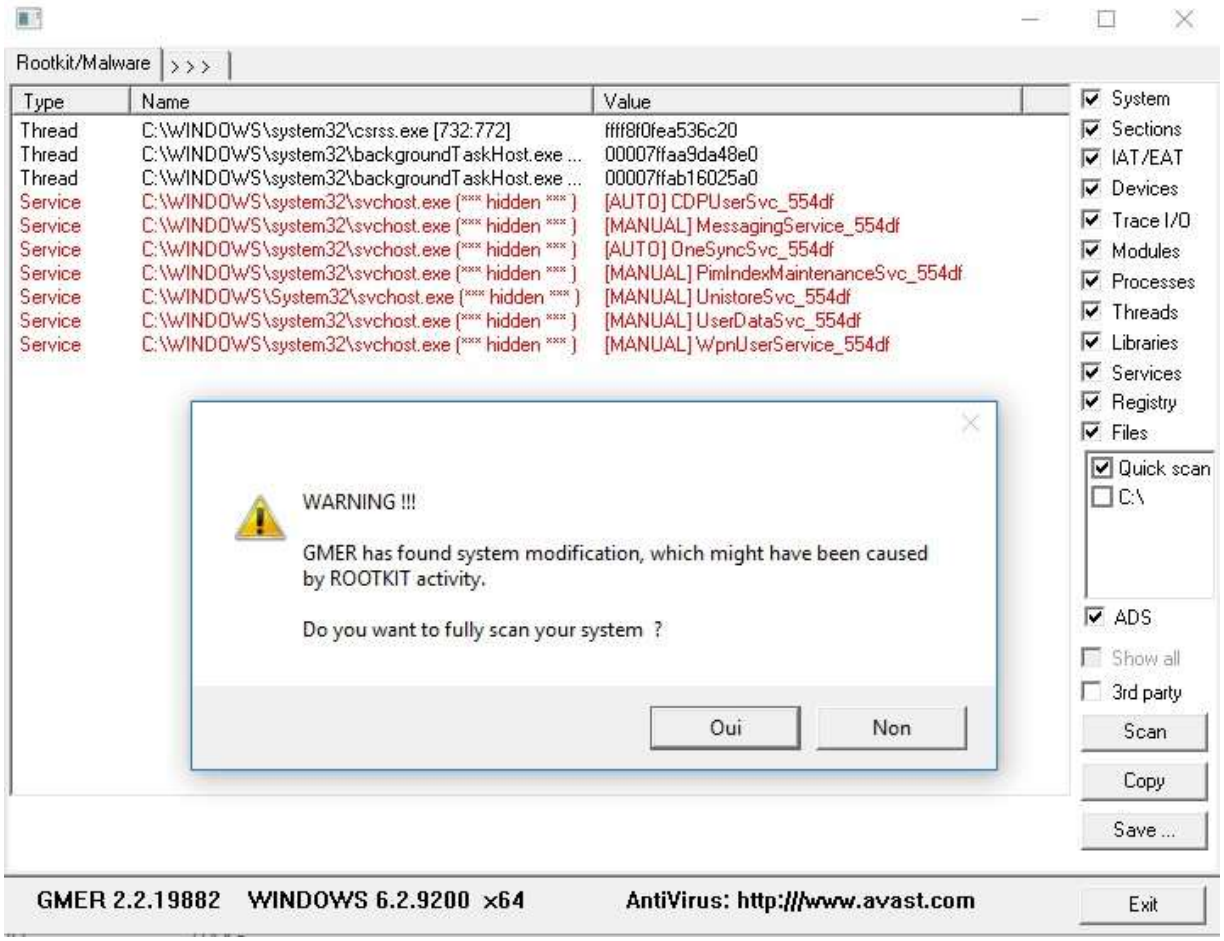
Step 2



Double-click the icon for the program.

Click the "Scan" button in the lower-right corner of the dialog box. Allow the program to scan your entire hard drive.

Step 3



When the program completes its scan, select any program or file listed in red. Right-click it and select "Delete."

If the red item is a service, it may be protected. Right-click the service and select "Disable." Reboot your computer and run the scan again, this time selecting "Delete" when that service is detected.

When your computer is free of Rootkits, close the program and restart your PC.

RESULT:

In this experiment a rootkit hunter software tool has been installed and the rootkits have been detected.