ERODE SENGUNTHAR
ENGINEERING COLLEGE
(An Autonomous Institution)

Approved by AICTE, New Delhi, Permanently Affiliated to Anna University - Chennai, Accredited by National Board of Accreditation (NBA), New Delhi &
National Assessment and Accreditation Council (NAAC), Bangalore with 'A' Grade

PERUNDURAI -638 057, TAMILNADU, INDIA

DEPARTMENT OF INFORMATION TECHNOLOGY

19IT705 - SECURITY LABORATORY

Regulation 2019

Year / Semester: IV / VII

2024 – 2025 (ODD SEMESTER)

PREPARED BY

Mr. M.DHANAPAL,
Assistant Professor / IT

OBJECTIVES: The purpose of learning this course is:

- To know the methods of conventional encryption.
- To understand the concept of public key encryption and number theory.
- To understand authentication and Hash function.
- To know the network security tools and applications.

OUTCOMES: At the end of this course, learners will be to:

- Develop code for classical Encryption Techniques to solve the problems.
- Build cryptosystems by applying symmetric and public key encryption algorithms.
- Construct code for authentication algorithms.
- Develop a signature scheme using Digital signature standard.
- Demonstrate the network security system using open source tools

LIST OF EXPERIMENTS

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. Implement the SIGNATURE SCHEME – Digital Signature Standard.
8. Demonstrate intrusion detection system (ids) using any tool eg. Snort or any other s/w.
9. Automated Attack and Penetration Tools Exploring N-Stalker, a Vulnerability Assessment Tool
10. Defeating Malware
    i) Building Trojans ii) Rootkit Hunter
11. Content Beyond the Syllabus : Triple DES

TABLE OF CONTENTS

Ex.No. : 1a **CAESAR CIPHER**
Date :

AIM:

To implement a Caesar cipher substitution technique in Java.

ALGORITHM:

1. Assign the 26 letters in alphabet to the variable named ALPHABET.
2. Convert the plaintext letters into lowercase.
3. To encrypt a plaintext letter, the first set of plaintext letters and slides it to LEFT by the number of positions of the secret shift.
4. The plaintext letter is then encrypted to the ciphertext letter on the sliding ruler underneath.
5. On receiving the ciphertext, the receiver who also knows the secret shift, positions his sliding ruler underneath the ciphertext alphabet and slides it to RIGHT by the agreed shift number, 3 in this case.
6. Then replaces the ciphertext letter by the plaintext letter on the sliding ruler underneath.

PROGRAM:

```
import java.util.Scanner;
public class ceasercipher
{
        public static final String ALPHABET="abcdefghijklmnopqrstuvwxyz";
        public static String encrypt(String plainText,int shiftKey)
        {
                plainText=plainText.toLowerCase();
                String cipherText="";
                for (int i=0; i<plainText.length();i++)
                {
                        int charPosition=ALPHABET.indexOf(plainText.charAt(i));
                        int keyVal=(shiftKey+charPosition)%26;
                        char replaceVal=ALPHABET.charAt(keyVal);
                        cipherText+=replaceVal;
                }
                return cipherText;
        }
        public static String decrypt(String cipherText,int shiftKey)
        {
                cipherText = cipherText.toLowerCase();
                String plainText = "";
                for(int i=0;i<cipherText.length();i++)
                {
                        int charPosition= ALPHABET. indexOf(cipherText. charAt(i));
                        int keyVal=(charPosition-shiftKey)%26;
                        if (keyVal< 0)
                        {
                                keyVal=ALPHABET.length()+keyVal;
                        }
                        char replaceVal=ALPHABET.charAt(keyVal);
                        plainText+=replaceVal;
```

```
            }
            return plainText;
        }
        public static void main(String[] args)
        {
            Scanner sc=new Scanner(System.in);
            System.out.println("Enter the Plain text for Encryption: ");
            String message=new String();
            message=sc.next();
            System.out.println("Encrypted message:Cipher Text="+encrypt(message,3));
            System.out.println("Decrypted message:Plain Text="+decrypt (encrypt(
            message,3),3));
            sc.close();
        }
}
```

OUTPUT:

```
F:\bin>javac ceasercipher.java
F:\bin>java ceasercipher
Enter the Plain text for Encryption:
covid
Encrypted message:Cipher Text=frylg
Decrypted message:Plain Text=covid
```

RESULT:

Thus the Caesar cipher substitution technique was implemented and executedsuccessfully.

Ex.No. : 1b **PLAYFAIR CIPHER**
Date :

AIM:

To implement a Playfair cipher substitution technique in Java.

ALGORITHM:

1. Read the keyword.
2. Then create the key table of 5x5 grid of alphabets.
3. Read the word to encrypt.
4. Ifthe input word should be even and then process it.
5. Then the plaintext message is split into pairs of two letters (digraphs).
6. Ifboth the letters are in the same column, take the letter below each one.
7. Ifboth letters are in the same row, take the letter to the right of each one.
8. If neither of the preceding two rules are true, form a rectangle with the two letters and take the letters on the horizontal opposite corner of the rectangle.

PROGRAM:

```
import java.util.Scanner;
public class Playfair1
{
	public static void main(String[] args)
	{
		Scanner in=new Scanner(System.in);
		System.out.print("Enter keyword: ");
		String key=in.nextLine();
		System.out.print("Enter message to encrypt: ");
		String msg=in.nextLine();
		PFEncryption pfEncryption=new PFEncryption();
		pfEncryption.makeArray(key);
		msg=pfEncryption.manageMessage(msg);
		pfEncryption.doPlayFair(msg, "Encrypt");
		String en=pfEncryption.getEncrypted();
		System.out.println("Encrypting. .. \n\nThe encrypted text is: " + en);
		System.out.println("============================");
		pfEncryption.doPlayFair(en, "Decrypt");
		System.out.print("\nDecrypting... \n\nThe encrypted text is: " +
		pfEncryption.getDecrypted());
	}
}
class PFEncryption
{
	private char [][] alphabets= new char[5][5];
	private char[] uniqueChar= new char[26];
	private String ch="ABCDEFGHIKLMNOPQRSTUVWXYZ";
	private String encrypted="";
	private String decrypted="";
	void makeArray(String keyword)
```

```
    {
            keyword=keyword.toUpperCase().replace("J","I");
            boolean present, terminate=false;
            int val=0;
            int uniqueLen;
            for (int i=0; i<keyword.length(); i++)
            {
                present=false;
               uniqueLen=0;
                if (keyword.charAt(i)!= ' ')
                {
                        for (int k=0; k<uniqueChar.length; k++)
                        {
                           if (Character.toString(uniqueChar[k])==null)
                            {
                              break;
                            }
                           uniqueLen++;
                        }
                        for (int j=0; j<uniqueChar.length; j++)
                        {
                             if (keyword.charAt(i)==uniqueChar[j])
                            {
                                present=true;
                             }
                         }
                        if (!present)
                          {
                           uniqueChar[val]=keyword.charAt(i);
                           val++;
                        }
                }
              ch=ch.replaceAll(Character.toString(keyword.charAt(i)), "");
    }
    for (int i=0; i<ch.length(); i++)
    {
            uniqueChar[val]=ch.charAt(i);
           val++;
 }
 val=0;
 for (int i=0; i<5; i++)
{
   for (int j=0; j<5; j++)
   {
            alphabets[i][j]=uniqueChar[val];
          val++;
            System.out.print(alphabets[i][j] + "\t");
}
  System.out.println();
}
```

```
}
String manageMessage(String msg)
{
   int val=0;
   int len=msg.length()-2;
   String newTxt="";
   String intermediate="";
   while (len>=0)
  {
      intermediate=msg.substring(val, val+2);
      if (intermediate.charAt(0)==intermediate.charAt(1))
     {
         newTxt=intermediate.charAt(0) + "x" + intermediate.charAt(1);
         msg=msg.replaceFirst(intermediate,  newTxt);
         len++;
     }
     len-=2;
     val+=2;
  }
   if (msg.length()%2!=0)
  {
      msg=msg+'x';
  }
   return msg.toUpperCase().replaceAll("J","I").replaceAll(" ","");
}
void doPlayFair(String msg, String tag)
{
   int val=0;
   while (val<msg.length())
   {
      searchAndEncryptOrDecrypt(msg.substring(val,val+2),tag);
      val+=2;
   }}
void searchAndEncryptOrDecrypt(String doubblyCh, String tag)
{
   char ch1=doubblyCh.charAt(0);
   char ch2=doubblyCh.charAt(1);
   int row1=0, col1=0, row2=0, col2=0;
   for (int i=0; i<5; i++)
  {
        for (int j=0; j<5; j++)
        {
              if (alphabets[i][j]==ch1)
              {
                 row1=i;
                   col1=j;
              }
              else if (alphabets[i][j]==ch2)
              {
                 row2=i;
```

```
                    col2=j;
                }       }   }
   if (tag=="Encrypt")
      encrypt(row1, col1, row2, col2);
   else if(tag=="Decrypt")
      decrypt(row1, col1, row2, col2);
}
void encrypt(int row1, int col1, int row2, int col2)
{
   if (row1==row2)
   {
      col1=col1+1;
      col2=col2+1;
      if (col1>4)
          col1=0;
       if (col2>4)
          col2=0;
encrypted+=(Character.toString(alphabets[row1][col1])+
Character.toString(alphabets[row1][col2]));
   }
         else if(col1==col2)
         {
               row1=row1+1;
               row2=row2+1;
               if (row1>4)
                    row1=0;
              if (row2>4)
          row2=0;
encrypted+=(Character.toString(alphabets[row1][col1])+
Character.toString(alphabets[row2][col1]));
   }
else
{ encrypted+=(Character.toString(alphabets[row1][col2])+
Character.toString(alphabets[row2][col1]));
}}
void decrypt(int row1, int col1, int row2, int col2)
{
   if (row1==row2)
  {
      col1=col1-1;
      col2=col2-1;
      if (col1<0)
          col1=4;
       if (col2<0)
          col2=4;
decrypted+=(Character.toString(alphabets[row1][col1])+
Character.toString(alphabets[row1][col2]));
   }
else if(col1==col2)
{
```

```
        row1=row1-1;
        row2=row2-1;
        if (row1<0)
            row1=4;
         if (row2<0)
            row2=4;
decrypted+=(Character.toString(alphabets[row1][col1])+
Character.toString(alphabets[row2][col1]));
    }
else
{
decrypted+=(Character.toString(alphabets[row1][col2])+
Character.toString(alphabets[row2][col1]));
    }}
String getEncrypted( )
{
    return encrypted;
}
String getDecrypted( )
{
    return decrypted;
}}
```

OUTPUT:

```
F:\bin>javac Playfair1.java

F:\bin>java Playfair1
Enter keyword: INFOSEC
Enter message to encrypt: cryptography
```

I	N	F	O	S
E	C	A	B	D
G	H	K	L	M
P	Q	R	T	U
V	W	X	Y	Z

```
Encrypting....

The encrypted text is: AQVTYBKPERLW
========================================

Decrypting....

The encrypted text is: CRYPTOGRAPHY
```

RESULT:

Thus the Playfair cipher substitution technique was implemented and executedsuccessfully.

Ex.No. : 1c **HILL CIPHER**
Date :

AIM:

To implement a Hill cipher substitution technique in Java.

ALGORITHM:

1. Obtain a plaintext message to encode in standard English with no spaces.
2. Split the plaintext into group of length three. To fill this, add X at the end.
3. Convert each group of letters with length three into plaintext vectors.
4. Replace each letter by the number corresponding to its position in the alphabet i.e. A=1, B=2, C=3…Z=0.
5. Create the keyword in a 3*3 matrix.
6. Multiply the two matrices to obtain the cipher text of length three.
7. For decryption, convert each entry in the ciphertext vector into its plaintext vector by multiplying the cipher text vector and inverse of a matrix.
8. Thus plain text is obtained fromcorresponding plaintext vector by corresponding position in the alphabet.

PROGRAM:

```
import java.util.Scanner;
import javax.swing.JOptionPane;
public class hillcipher
{
   //the 3x3 key matrix for 3 characters at once
   public static int[][] keymat = new int[][]
   {
            { 1, 2, 1 },
            { 2, 3, 2 },
            { 2, 2, 1 },
   };
   public static int[][] invkeymat = new int[][]
   {
            { -1, 0, 1 },
            { 2, -1, 0 },
            { -2, 2, -1},
   };
   public static String key = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

   public static void main(String[] args)
   {
      String text,outtext ="",outtext1="";
      int ch, n;
         Scanner sc=new Scanner(System.in);
      System.out.println("Enter the Plain text for Encryption: ");
      //String text=new String();
         text=sc.next();

         text = text.toUpperCase();
      text = text.replaceAll("\\s",""); //removing spaces
```

```
        n = text.length() % 3;
        if(n!=0)
        {
           for(int i = 1; i<= (3-n);i++)
           {
              text+= 'X';
           }
        }
        System.out.println("Padded Text:" + text);
        char[] ptextchars = text.toCharArray();
                for(int i=0;i< text.length(); i+=3)
                      {
                   outtext += encrypt(ptextchars[i],ptextchars[i+1],ptextchars[i+2]);
                }
                            System.out.println("Encypted Message: " + outtext);

                 char[] ptextchars1 = outtext.toCharArray();
           for(int i=0;i< outtext.length(); i+=3)
                      {
                   outtext1 += decrypt(ptextchars1[i],ptextchars1[i+1],ptextchars1[i+2]);
                }
           System.out.println("Decrypted Message: " + outtext1);
}

private static String encrypt(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 decrypt(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;
}
}
```

OUTPUT:

```
F:\bin>javac hillcipher.java

F:\bin>java hillcipher
Enter the Plain text for Encryption:
mothertheresa
Padded Text:MO THERTHERESAXX
Encypted Message: AAHXIGPPLJEROLR
Decrypted Message: MOTHERTHERESAXX

F:\bin>java hillcipher
Enter the Plain text for Encryption:
hilcipher
Padded Text:HILCIPHER
Encypted Message: TIIWGHXIG
Decrypted Message: HILCIPHER
```

RESULT:

Thus the Hill cipher substitution technique was implemented and executedsuccessfully.

Ex.No. : 1d **VIGENERE CIPHER**
Date :

AIM:

To implement a Java 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 Vigeneretable.
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:

```
public class vigenerecipher1
{
    public static String encrypt(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;
    }
    public static String decrypt(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)
    {
          System.out.println("Enter the key: ");
          String key = System.console().readLine();
          System.out.println("Enter the message for encrytption: ");
          String message = System.console().readLine();
          String encryptedMsg=encrypt(message,key);
          System.out.println("String :"+message);
          System.out.println("Encrypted message:Cipher Text=" +encryptedMsg);
          System.out.println("Decrypted message:Plain Text="
         +decrypt(encryptedMsg,key));
    }
}
```

OUTPUT:

```
F:\bin>javac vigenerecipher1.java
F:\bin>java  vigenerecipher1
Enter the key:
SECURITY
Enter the message for encrytption:
CRYPTOGRAPHY
String :CRYPTOGRAPHY
Encrypted message:UVAJKWZPSTJS
Decrypted message:CRYPTOGRAPHY
```

RESULT:

Thus the Vigenere cipher substitution technique was implemented and executedsuccessfully.

Ex.No. : 2a **RAIL FENCE CIPHER**
Date :

AIM:

To implement a rail fence transposition technique in Java.

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:

```
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;
            System.out.println("Enter the Plain text: ");
            msg = System.console().readLine();
               int depth = 2;
            enc = rf.encode(msg, depth);
            dec = rf.decode(enc, depth);
            System.out.println("Plain Text:"+msg);
            System.out.println("Encrypted Message-Cipher Text:"+enc);
            System.out.printf("Decrypted Message-:"+dec);
         }
}
```

OUTPUT:

```
F:\bin>javac railfencecipher.java
F:\bin>java  railfencecipher
Enter the Plain text:
paulinefreeda
Plain Text:paulinefreeda
Encrypted Message-Cipher Text:puiereaalnfed
Decrypted Message-:paulinefreeda
```

RESULT:

Thus the Rail Fence Transposition Technique was implemented and executedsuccessfully.

Ex.No.: 2b ROW AND COLUMN TRANSFORMATION TECHNIQUE
Date :

AIM:

To implement a rail fence transposition technique in Java.

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.

EXAMPLE:

```
A U T H O R
1 6 5 2 3 4
-----------
W E A R E D
I S C O V E
R E D S A V
E Y O U R S
E L F A B C
```

yields the cipher

```
W I R E E R O S U A E V A R B D E V S C A C D O F E S E Y L .
```

PROGRAM:

```
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] = '#';
                }
            }
        }
      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]);
            } }
        System.out.println();
    }
}
```

OUTPUT:

```
F:\bin>javac TransCipher.java
F:\bin>java  TransCipher
Enter the plain text
altrozcarshervin
altrozcarshervin
aorrlzsvtchiraen
```

a	**l**	**t**	**r**
o	**z**	**c**	**a**
r	**s**	**h**	**e**
r	**v**	**I**	**n**

RESULT:

Thus the Row and Column Transposition Technique was implemented and executed successfully.

Ex.No. : 3 **DATA ENCRYPTION STANDARD (DES)**
Date :

AIM:

To apply 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 (/).
   - Algorithm name
   - Mode (optional)
   - 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.

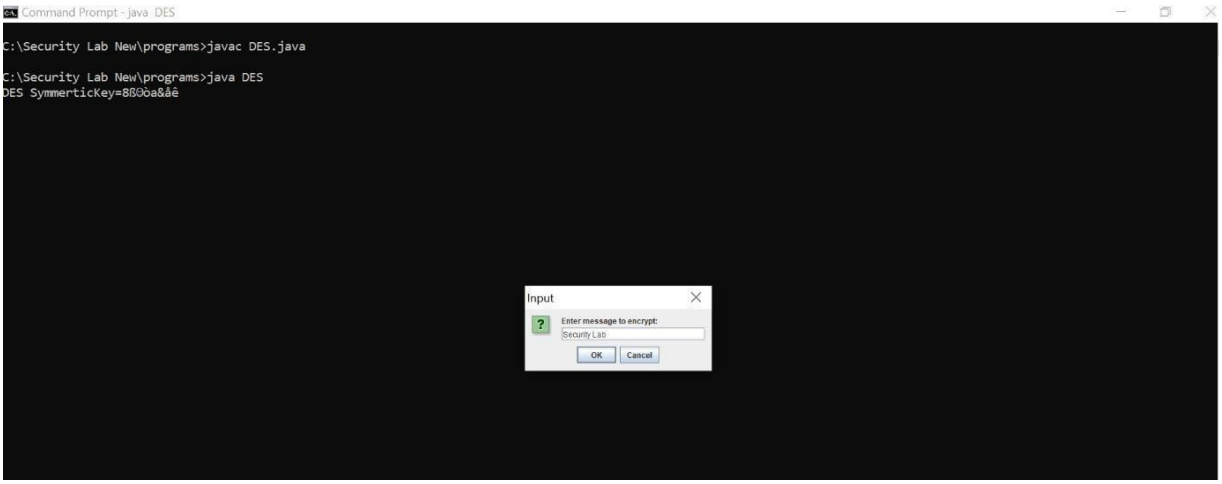
PROGRAM:

```
import javax.swing.*;
import java.security.SecureRandom;
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import java.util.Random;
class DES
{
        byte[] skey=new byte[1000];
        String skeystring;
        static byte[] raw;
        String inputmessage,encryptedata,decryptedmessage;
        public DES()
        {
                try
                {
                        generatesymmetrickey();
                        inputmessage=JOptionPane.showInputDialog(null,"Enter message to
                        encrypt:");
                        byte[] ibyte =inputmessage.getBytes();
                        byte[] ebyte=encrypt(raw, ibyte);
                        String encrypteddata=new String(ebyte);
                        System.out.println("Encrypted message:"+encrypteddata);
                        JOptionPane.showMessageDialog(null,"Encrypted
                        Data"+"\n"+encrypteddata);
                        byte[] dbyte=decrypt(raw,ebyte);
                        String decryptedmessage=new String(dbyte);
                        System.out.println("Decrypted message:"+decryptedmessage);
```

```
                    JOptionPane.showMessageDialog(null,"Decrypted Data
                    "+"\n"+decryptedmessage);
                }
                catch(Exception e)
                {
                    System.out.println(e);
                }
        }
void generatesymmetrickey()
{
         try
        {
                Random r = new Random();
                int num=r.nextInt(10000);
                String knum=String.valueOf(num);
                byte[] knumb=knum.getBytes();
                skey=getRawKey(knumb);
                skeystring=new String(skey);
                System.out.println("DES
                SymmerticKey="+skeystring);
        }
        catch(Exception e)
        {
                System.out.println(e);
        }
}

private static byte[] getRawKey(byte[] seed) throws Exception
{
        KeyGenerator kgen=KeyGenerator.getInstance("DES ");
        SecureRandom sr =SecureRandom.getInstance("SHA1PRNG");
        sr.setSeed(seed);
        kgen.init(56,sr);
        SecretKey skey=kgen.generateKey();
        raw=skey.getEncoded();
        return raw;
}

private static byte[] encrypt(byte[] raw,byte[] clear) throws Exception
{
        SecretKey seckey = new SecretKeySpec(raw, "DES");
        Cipher cipher = Cipher.getInstance("DES");
        cipher.init(Cipher.ENCRYPT_MODE,seckey);
        byte[] encrypted=cipher.doFinal(clear);
        return encrypted;
}

private static byte[] decrypt(byte[] raw,byte[] encrypted) throws Exception
{
        SecretKey seckey = new SecretKeySpec(raw, "DES");
```

```
        Cipher cipher = Cipher.getInstance("DES");
        cipher.init(Cipher.DECRYPT_MODE,seckey);
        byte[] decrypted = cipher.doFinal(encrypted);
        return decrypted;
 }
 public static void main(String args[])
{
        DES des=new DES();
}

}
```

OUTPUT:



RESULT:

Thus the java program for applying Data Encryption Standard (DES) Algorithm for a practical application of User Message Encryption is written and executed successfully.

Ex.No. : 4 **AES ALGORITHM**
Date :

AIM:

To apply 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:

```
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) {

        System.out.println("Enter the secret key: ");
        String secretKey = System.console().readLine();

        System.out.println("Enter the original URL: ");
        String originalString = System.console().readLine();

        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:

```
C:\Security Lab New\programs>java AES
Enter the secret key:
annaUniversity
Enter the original URL:
www.annauniv.edu
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 applying Advanced Encryption Standard (AES) Algorithm for a practical application of URL encryption is written and executed successfully.

Ex.No. : 5 **RSA ALGORITHM**
Date :

AIM:

To implement a RSA algorithm using HTML and Javascript.

ALGORITHM:

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

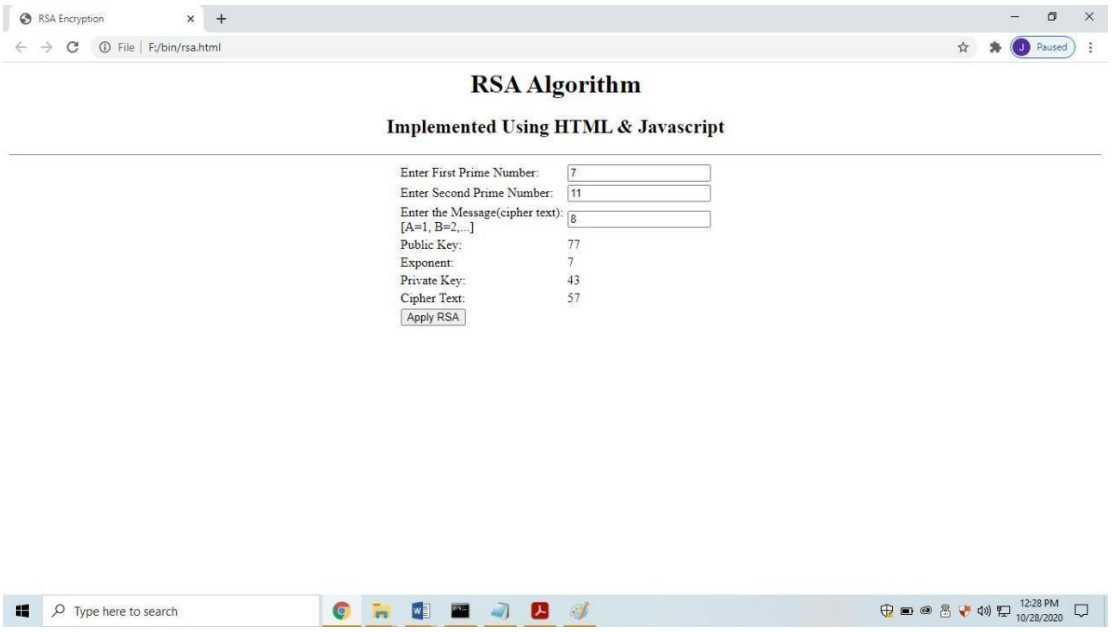
PROGRAM:

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 was implemented using HTML and Javascript and executed successfully.

Ex.No. : 6 DIFFIE-HELLMAN KEY EXCHANGE ALGORITHM
Date :

AIM:

To implement a Diffie-Hellman Key Exchange algorithm.

ALGORITHM:

1. Sender and receiver publicly agree to use a modulus p and base g which is a primitive root modulo p.
2. Sender chooses a secret integer x then sends Bob R1 = g^x mod p
3. Receiver chooses a secret integer y, then sends Alice R2 = g^y mod p
4. Sender computes k1 = B^x mod p
5. Receiver computes k2 = A^y mod p
6. Sender and Receiver now share a secret key.

PROGRAM:

```
import java.io.*;
import java.math.BigInteger;
class dh
{
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.print("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 Sender's side:"+k1);
BigInteger k2=R1.modPow(y,p);
System.out.println("Key calculated at Receiver's side:"+k2);
System.out.println("Diffie-Hellman secret key was calculated.");
}
}
```

OUTPUT

```
C:\Security Lab New\programs>javac dh.java

C:\Security Lab New\programs>java dh
Enter prime number:
11
Enter primitive root of 11:7
Enter value for x less than 11:
3
R1=2
Enter value for y less than 11:6
R2=4
Key calculated at Sender's side:9
Key calculated at Receiver's side:9
Diffie-Hellman secret key was calculated.
```

RESULT:

Thus the Diffie-Hellman key exchange algorithm was implemented and executed successfully.

Ex.No. : 7 **DIGITAL SIGNATURE SCHEME**
Date :

AIM:

To implement the signature scheme - Digital Signature Standard.

ALGORITHM:

1. Declare the class and required variables.
2. Create the object for the class in the main program.
3. Access the member functions using the objects.
4. Implement the SIGNATURE SCHEME - Digital Signature Standard.
5. It uses a hash function.
6. The hash code is provided as input to a signature function along with a random number K generated for the particular signature.
7. The signature function also depends on the sender,,s private key.
8. The signature consists of two components.
9. The hash code of the incoming message is generated.
10. The hash code and signature are given as input to a verification function.

PROGRAM:

```
import java.util.*;
import java.math.BigInteger;
class dsaAlg {
final static BigInteger one = new BigInteger("1");
final static BigInteger zero = new BigInteger("0");
public static BigInteger getNextPrime(String ans)
{
BigInteger test = new BigInteger(ans);
while (!test.isProbablePrime(99))
e:
{
test = test.add(one);
}
return test;
}
public static BigInteger findQ(BigInteger n)
{
BigInteger start = new BigInteger("2");
while (!n.isProbablePrime(99))
{
while (!((n.mod(start)).equals(zero)))
{
start = start.add(one);
}
n = n.divide(start);
}
return n;
}
```

```
public static BigInteger getGen(BigInteger p, BigInteger q,
Random r)
{
BigInteger h = new BigInteger(p.bitLength(), r);
h = h.mod(p);
return h.modPow((p.subtract(one)).divide(q), p);
}
public static void main (String[] args) throws
java.lang.Exception
{
Random randObj = new Random();
BigInteger p = getNextPrime("10600"); /* approximate
prime */
BigInteger q = findQ(p.subtract(one));
BigInteger g = getGen(p,q,randObj);
System.out.println(" \n simulation of Digital Signature Algorithm \n");
System.out.println(" \n global public key components are:\n");
System.out.println("\np is: " + p);
System.out.println("\nq is: " + q);
System.out.println("\ng is: " + g);
BigInteger x = new BigInteger(q.bitLength(), randObj);
x = x.mod(q);
BigInteger y = g.modPow(x,p);
BigInteger k = new BigInteger(q.bitLength(), randObj);
k = k.mod(q);
BigInteger r = (g.modPow(k,p)).mod(q);
BigInteger hashVal = new BigInteger(p.bitLength(),
randObj);
BigInteger kInv = k.modInverse(q);
BigInteger s = kInv.multiply(hashVal.add(x.multiply(r)));
s = s.mod(q);
System.out.println("\nsecret information are:\n");
System.out.println("x (private) is:" + x);
System.out.println("k (secret) is: " + k);
System.out.println("y (public) is: " + y);
System.out.println("h (rndhash) is: " + hashVal);
System.out.println("\n generating digital signature:\n");
System.out.println("r is : " + r);
System.out.println("s is : " + s);
BigInteger w = s.modInverse(q);
BigInteger u1 = (hashVal.multiply(w)).mod(q);
BigInteger u2 = (r.multiply(w)).mod(q);
BigInteger v = (g.modPow(u1,p)).multiply(y.modPow(u2,p));
v = (v.mod(p)).mod(q);
System.out.println("\nverifying digital signature (checkpoints)\n:");
System.out.println("w is : " + w);
System.out.println("u1 is : " + u1);
System.out.println("u2 is : " + u2);
System.out.println("v is : " + v);
if (v.equals(r))
```

```
{
System.out.println("\nsuccess: digital signature is verified!\n " + r);
}
else
{
System.out.println("\n error: incorrect digital signature\n ");
}
}
}
```

OUTPUT:

```
C:\Security Lab New\programs>javac dsaAlg.java
C:\Security Lab New\programs>java dsaAlg
simulation of Digital Signature Algorithm
global public key components are:
p is: 10601
q is: 53
g is: 6089
secret information are:
x (private) is:6
k (secret) is: 3
y (public) is: 1356
h (rndhash) is: 12619
generating digital signature:
r is : 2
s is :41
verifying digital signature (checkpoints):
w is : 22
u1 is : 4
u2 is : 44
v is : 2
success: digital signature is verified!
 2
```

RESULT:

Thus the Digital Signature Standard Signature Scheme has been implemented andexecuted successfully.

Ex. No. :8 **INTRUSION DETECTION SYSTEM (IDS)**
Date:

AIM:

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

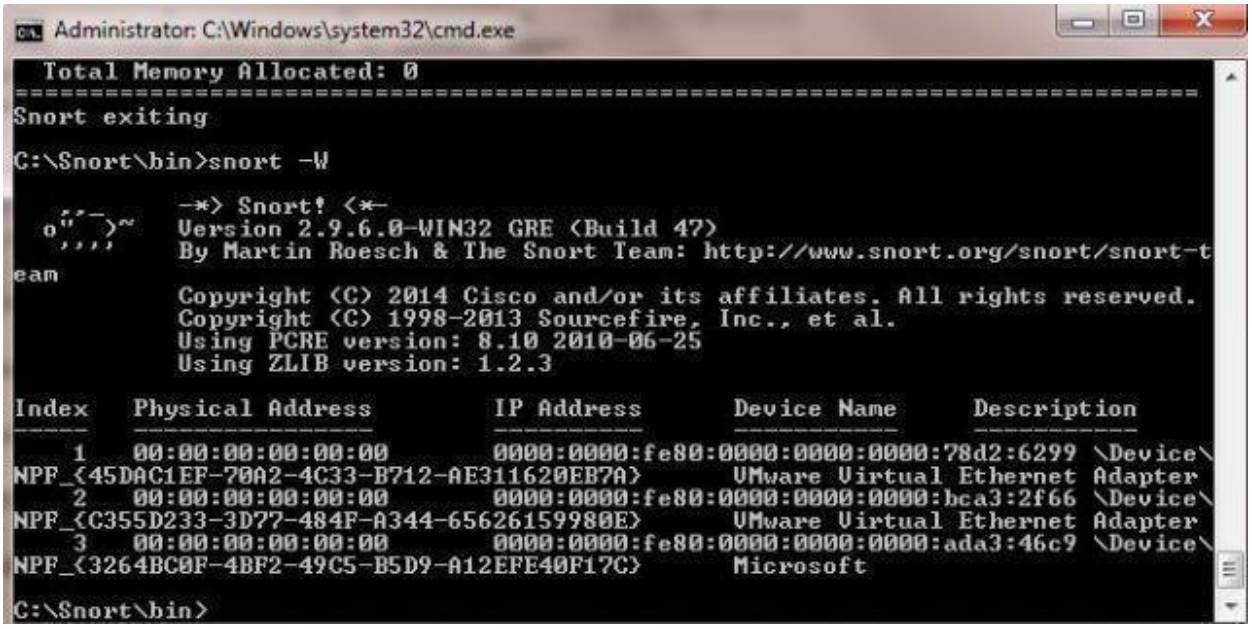
STEPS ON CONFIGURING AND INTRUSION DETECTION:

1. Download Snort from the Snort.org website. (http://www.snort.org/snort-downloads)
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, itis 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 toyour 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 ofthe "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)

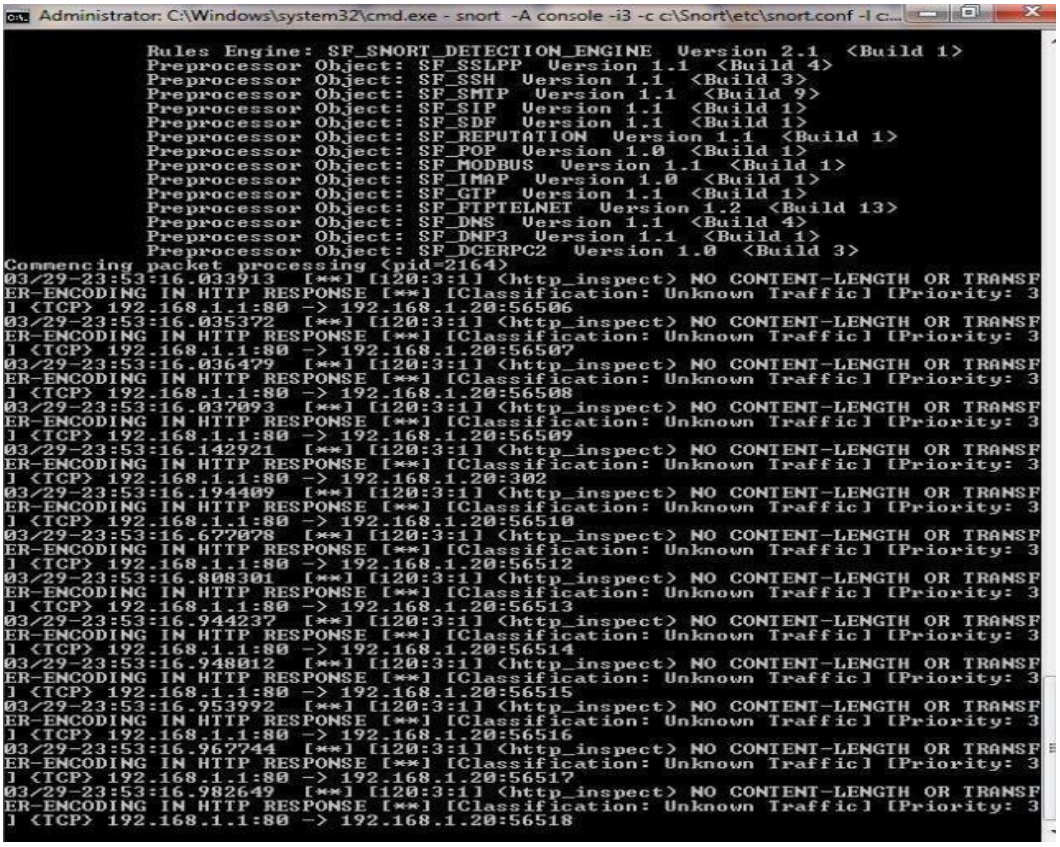
Ifa log is created, select the appropriate program to open it. You can use WordPard or NotePad++ to read the file.

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 using the Open Source Intrusion Detection Tool Snort.

Ex.No. : 9 **EXPLORING N-STALKER**
Date :

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 toolalso 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.

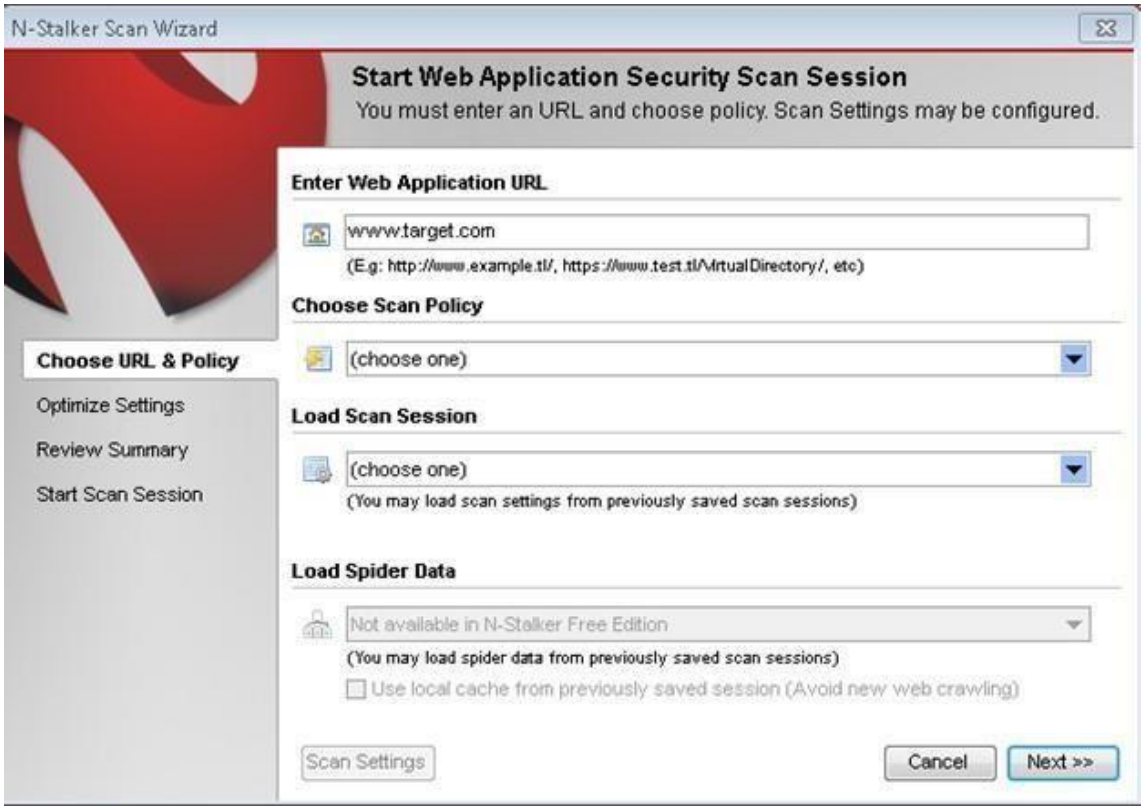
In scan policy, you can select from the four options,

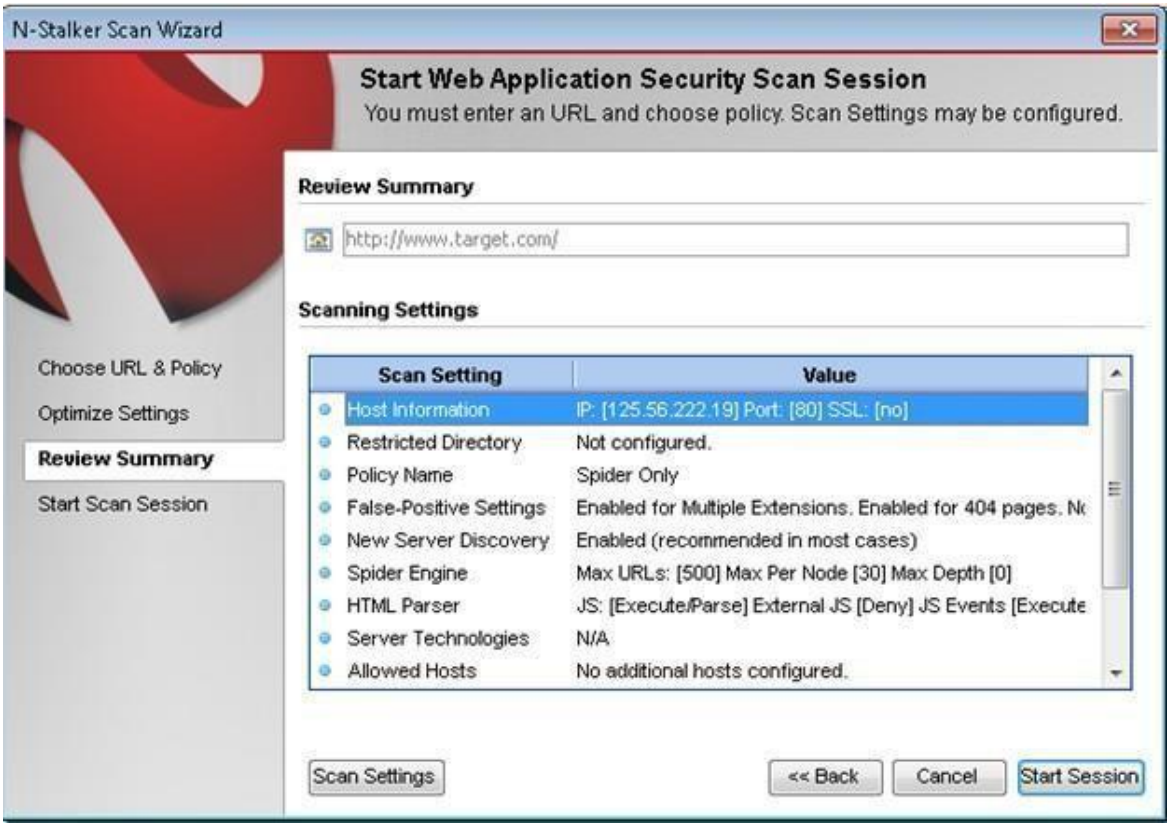
- Manual test which will crawl the website and will be waiting for manual attacks.
- full xss assessment
- owasp policy

- Web server infrastructure analysis.

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

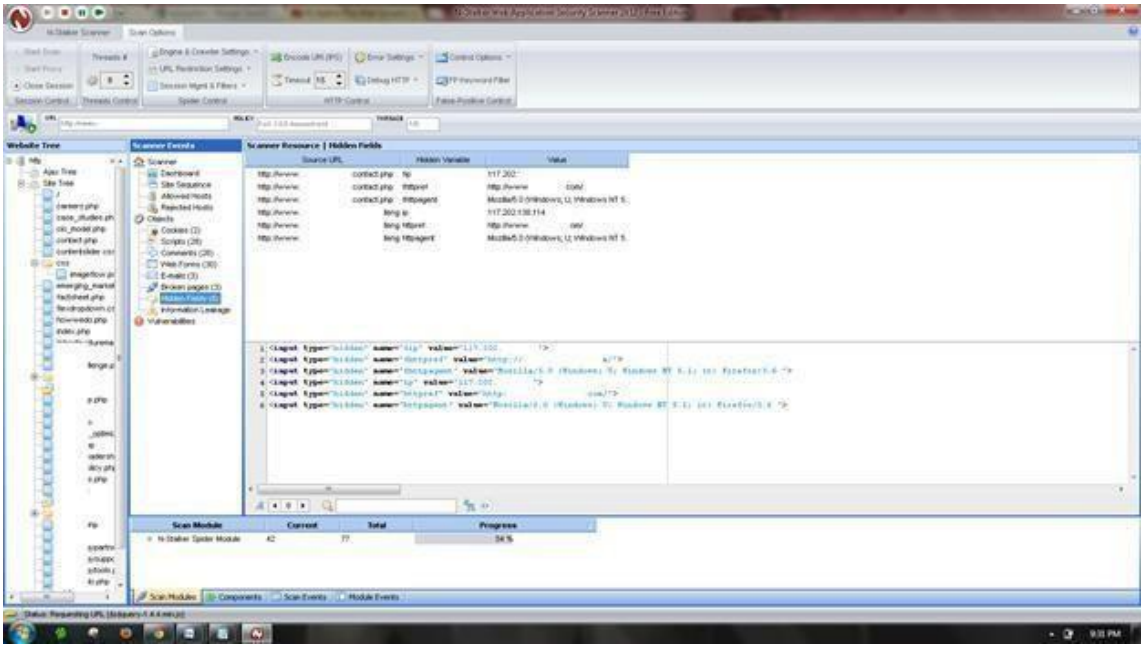
In review option, you can get all the information like host information, technologies used, policy name, etc.



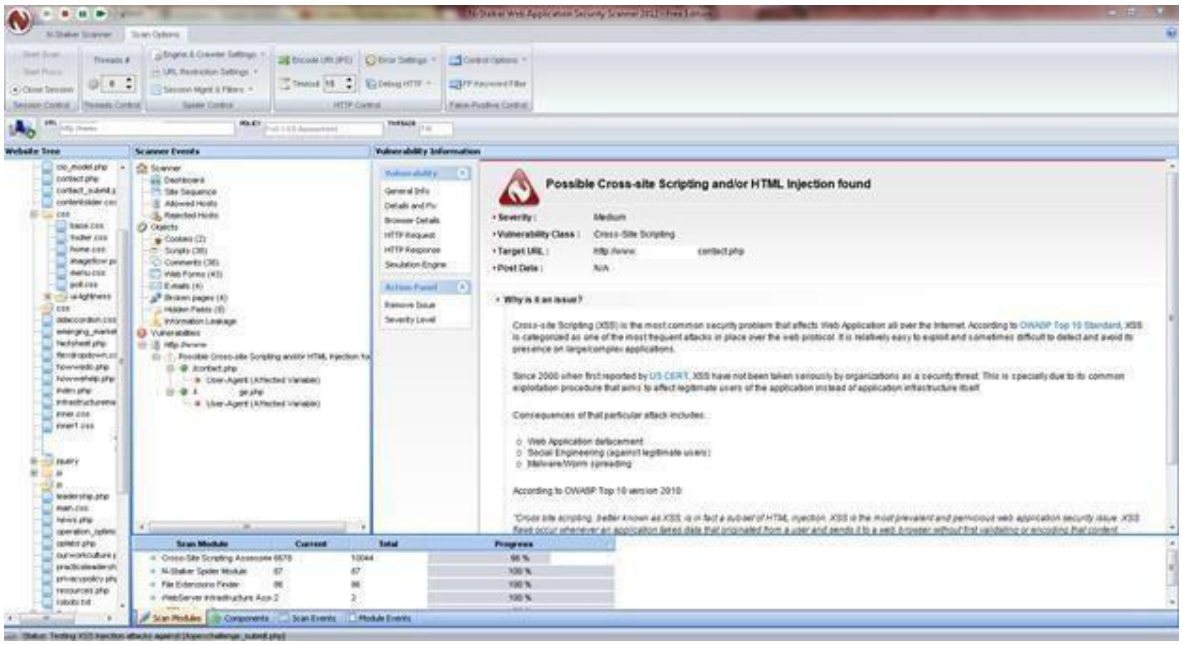


Once done, start the session and start the scan.

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.: 10.a **DEFEATING MALWARE - BUILDING TROJANS**
Date :

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. : 10.b **DEFEATING MALWARE - ROOTKIT HUNTER**
Date :

AIM:

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

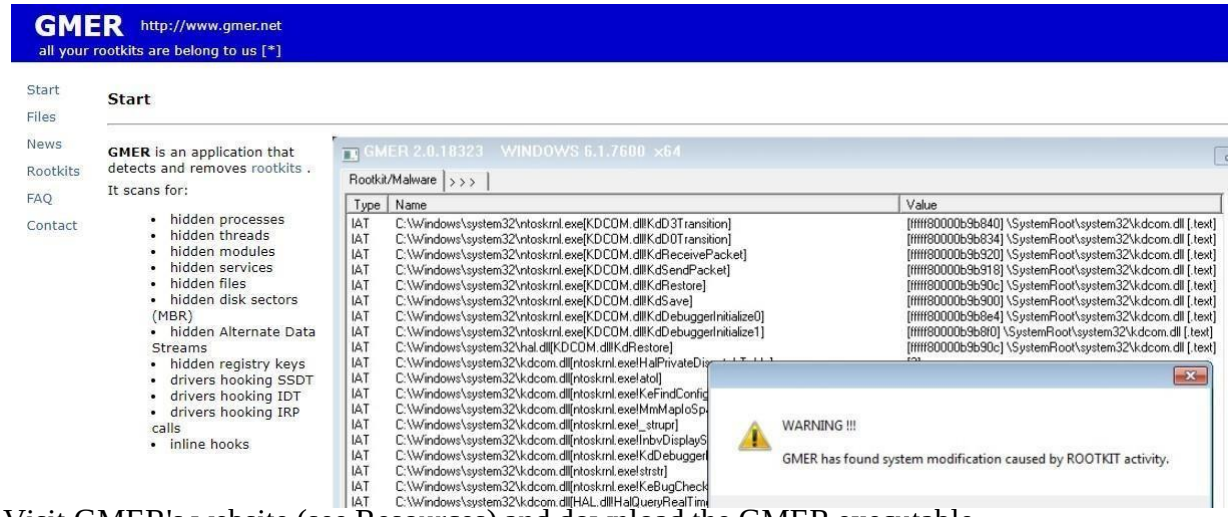
ROOTKIT HUNTER:

- 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:

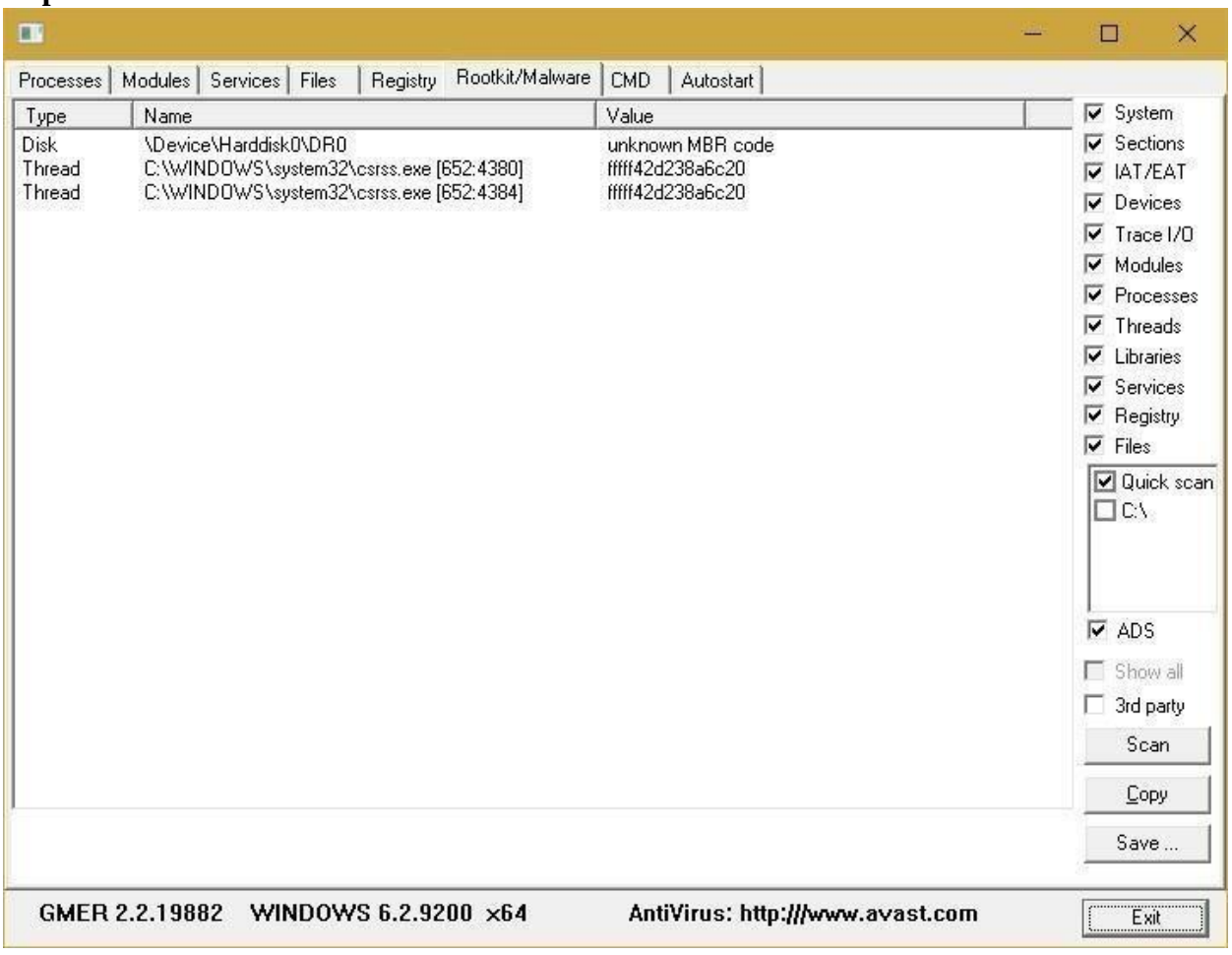
- GMER is a software tool written by a Polish researcher Przemysław Gmerek, for detecting and removing rootkits.
- It runs on Microsoft Windows and has support for Windows NT, 2000, XP, Vista, 7, 8 and 10. With version 2.0.18327 full support for Windows x64 is added.

Step 1



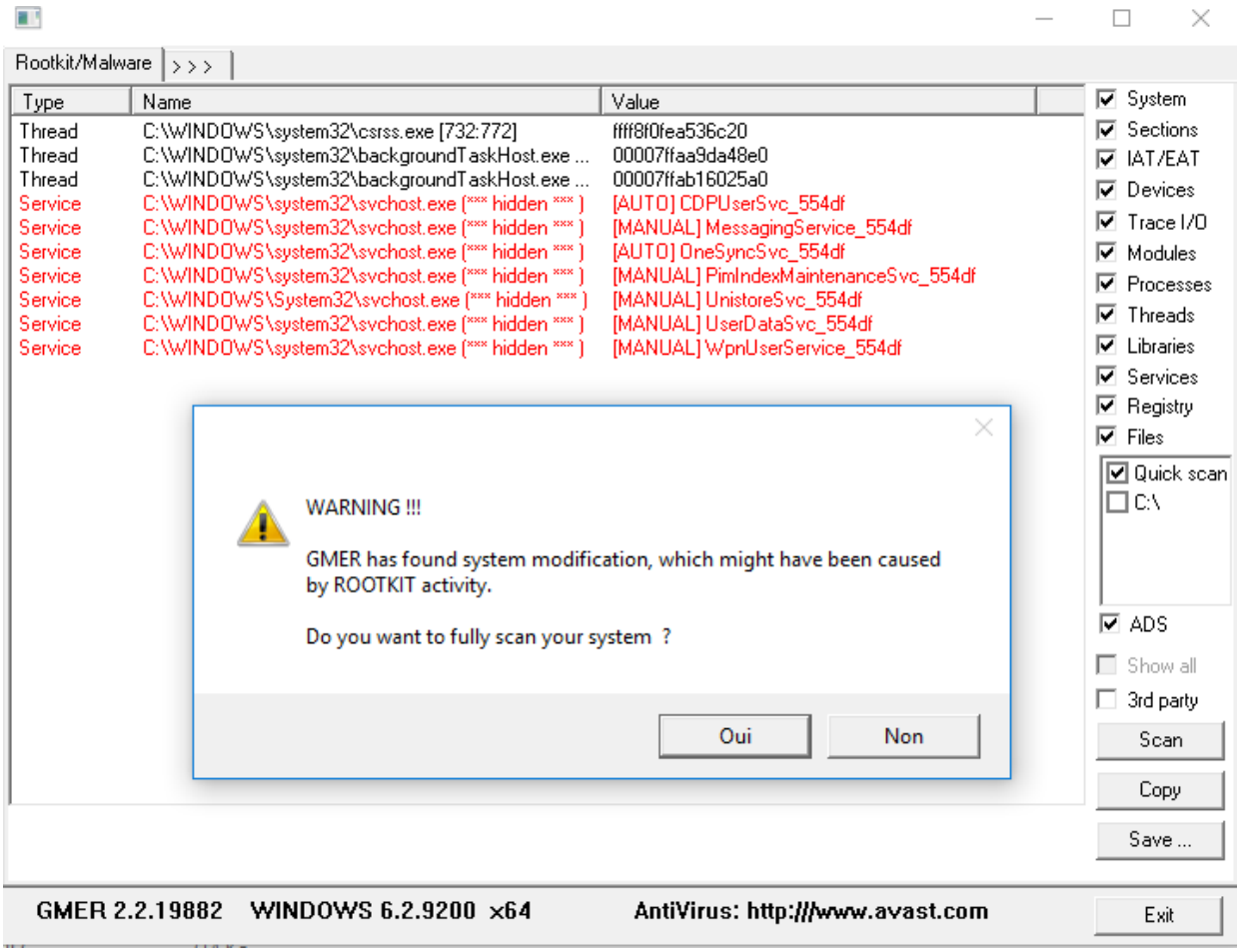
Visit GMER's website (see Resources) and download the GMER executable.
Click the "Download EXE" button to download the program with a random file name, as some rootkits will close "gmer.exe" before you can open it.

Step 2



Double-click the icon for the program.
Click the "Scan" button in the lower-right corner ofthe 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:

A rootkit hunter software tool *gmer* has been installed and the rootkits have been detected.

EX.No.: 11	TRIPLE DES

AIM:

To implement the TRIPLE DES in java.

ALGORITHM:

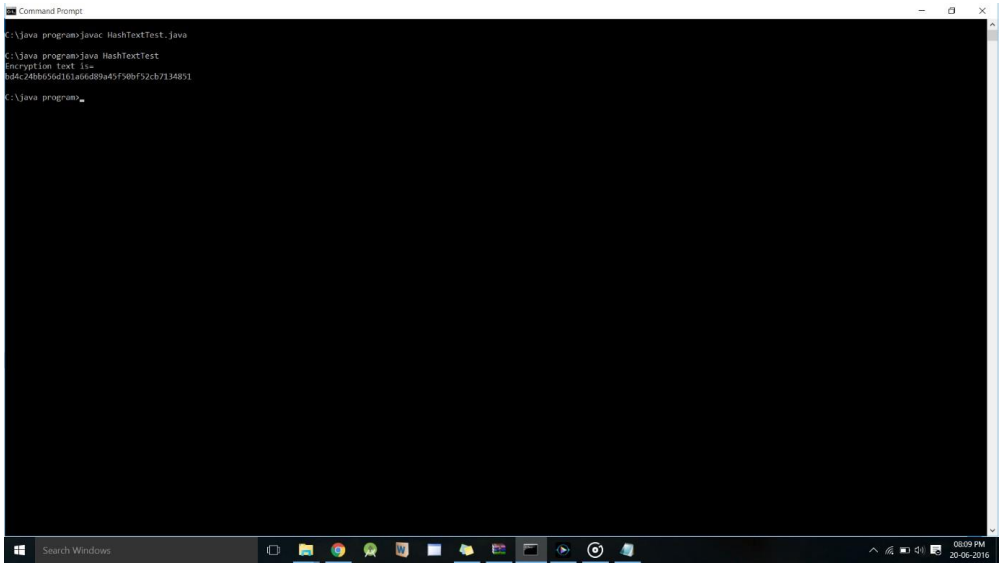
1. Start the program.
2. Encrypt the plaintext blocks using single DES with key K_1.
3. Now decrypt the output of step 1 using single DES with key K_2.
4. Finally, encrypt the output of step 2 using single DES with key K_3.
5. The output of step 3 is the ciphertext.
6. Decryption of a ciphertext is a reverse process. User first decrypt using K_3, then encrypt with K_2, and finally decrypt with K_1.
7. Stop the program.

PROGRAM:

```
import java.util.Arrays; import
javax.crypto.Cipher;
import  javax.crypto.SecretKey;
import  javax.crypto.spec.SecretKeySpec;
import org.apache.commons.codec.binary.Base64; public
class TripleDESTest
{
        public static void main(String[] args) throws Exception
    {
       String text = "textToEncrypt";
       String codedtext = new TripleDESTest()._encrypt(text,"SecretKey");
       String decodedtext = new TripleDESTest()._decrypt(codedtext,"SecretKey");
            System.out.println(codedtext + " ---> " + decodedtext);
      }
     private String _encrypt(String message, String secretKey) throws Exception
     {
      MessageDigest md = MessageDigest.getInstance("SHA-1");
            byte[] digestOfPassword = md.digest(secretKey.getBytes("utf-8"));
            byte[] keyBytes = Arrays.copyOf(digestOfPassword, 24);

            SecretKey key = new SecretKeySpec(keyBytes, "DESede");Cipher
            cipher = Cipher.getInstance("DESede");
            cipher.init(Cipher.ENCRYPT_MODE, key);

            byte[] plainTextBytes = message.getBytes("utf-8");byte[]
       buf = cipher.doFinal(plainTextBytes);
```

```
        byte [] base64Bytes = Base64.encodeBase64(buf);
        String base64EncryptedString = new String(base64Bytes);

        return base64EncryptedString;
    }
    private String _decrypt(String encryptedText, String secretKey) throws Exception { byte[]
        message = Base64.decodeBase64(encryptedText.getBytes("utf-8"));
            MessageDigest md = MessageDigest.getInstance("SHA-1");
            byte[] digestOfPassword = md.digest(secretKey.getBytes("utf-8"));
            byte[] keyBytes = Arrays.copyOf(digestOfPassword, 24); SecretKey
            key = new SecretKeySpec(keyBytes, "DESede");

            Cipher decipher = Cipher.getInstance("DESede");
            decipher.init(Cipher.DECRYPT_MODE, key);

            byte[] plainText = decipher.doFinal(message);

            return new String(plainText, "UTF-8");
    }
}
```

OUTPUT:



RESULT:

Thus the program to implement the TRIPLE DES in java has been executed and the outputwas verified successfully.