SSM INSTITUTE OF ENGINEERING AND TECHNOLOGY

Sindalagundu Post, Palani Road, Dindigul – 624 002
Ph: 0451 – 2448800 – 2448899 (100 lines) Fax: 0451 – 2557755

E-mail: info@ssmiet.com, website: www.ssmiet.com
Approved by AICTE, New Delhi and Affiliated to Anna University, Chennai

IT8761 – SECURITY LABORATORY

RECORD

NAME	
REGISTER NUMBER	
YEAR	IV
SEMESTER	VII
DEPARTMENT	Computer Science and Engineering
ACADEMIC YEAR	2021-2022

SSM INSTITUTE OF ENGINEERING & TECHNOLOGY

Sindalagundu Post, Palani Road, Dindigul – 624 002
Ph: 0451 – 2448800 – 2448899 (100 lines) Fax: 0451 – 2557755
E-mail: info@ssmiet.com, website: www.ssmiet.com
Approved by AICTE, New Delhi and Affiliated to Anna University, Chennai

PRACTICAL RECORD

BONAFIDE CERTIFICATE

REGISTER NUMBER

Certified that this is the bonafide record of work done by Mr./Miss……………………..…….,

*(**Reg.No. :**)……………………... of **B.E. Computer Science and Engineering** branch in the*

***IT8761 – SECURITY LABORATORY,** as prescribed by the Anna University Chennai, for the*

*third semester during the academic year **2021-2022.***

Staff in Charge *Head of the Department*

Submitted for the University Practical Examination held on …………………………… ………

Internal Examiner *External Examiner*

SSM INSTITUTE OF ENGINEERING AND TECHNOLOGY

Dindigul- Palani Highway, Dindigul – 624 002.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISION

To grow as an eminent department with excellence in computing and research by integrating computer and information technology to develop products and services for the benefit of the society with ethical values.

MISSION

- To instill in students the urge to learn by imparting finest quality education
- To impart good attitude and integrate creativity and research orientation
- To initiate interest and equip students to design and develop intelligent products
- To inculcate the desire to serve the society with ethical values.

PROGRAM SPECIFIC OUTCOMES

A graduate of the Computer Science and Engineering Program will able to:

- PSO1: Understand the principles of basic engineering and acquire the hardware and software aspects of computer science and engineering.
- PSO2: Design and develop applications or products using various programming languages.

PROGRAM EDUCATIONAL OBJECTIVES

To produce graduates who have technical proficiency, continuous educational growth, managerial skills and perform service to the society.

PEO1: Technical Proficiency

- Advance professionally to roles of greater computer engineering responsibilities in government and private organizations, through providing solutions to challenging problems in their profession by applying computer engineering theory and principles.

PEO2: Continuous educational growth

- Engage in life-long learning through successful completion of post graduate programs in engineering and interdisciplinary areas to emerge as researchers, experts, and educators.

PEO3: Managerial Skills

- Develop and refine their knowledge to provide exposure to emerging cutting edge technologies, adequate training and opportunities to work as teams on multidisciplinary projects with effective communication skills and leadership qualities.

PEO4: Service to the society

- Establish a commitment to the society by applying technical skills and knowledge to support various service activities.

IT8761 SECURITY LABORATORY L T P C

0 0 4 2

OBJECTIVES:

- To learn different cipher techniques
- To implement the algorithms DES, RSA, MD5,SHA-1
- To use network security tools and vulnerability assessment 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. 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

TOTAL: 60 PERIODS

OUTCOMES:

Upon Completion of the course, the students will be able 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

Contents

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

AIM:

To implement a program for encrypting a plain text and decrypting a cipher text using Caesar Cipher (shift cipher) substitution technique.

ALGORITHM:

1. Create and initialize a string ALPHABET that holds the alphabet characters. The index position of the string represents the numeric representation for the corresponding characters in the string ALPHABET.
2. Read the input plain text to be encrypted and also the Caeser cipher key an integer between 0 and 25.
3. Encrypt the plain text using the Caeser cipher key and the ALPHABET string.

   a. For every character in the plain text

      i. Search the ALPHABET string for the character and assign the numeric representation of the character (plainnumeric) as the index position of the character in the ALPHABET string.

      ii. Perform encryption using ciphernumeric = (plainnumeric + Caeser cipher key) mod 26

      iii. Use ciphernumeric as the index position and get the corresponding character from the ALPHABET string as the equivalent cipher text character for the plain text character b. Print the equivalent cipher text
4. Decrypt the cipher text using the Caeser cipher key and the ALPHABET string.

   a. For every character in the cipher text

      i. Search the ALPHABET string for the character and assign the numeric representation of the character (ciphernumeric) as the index position of the character in the ALPHABET string.

      ii. Perform decryption using Plainnumeric = (ciphernumeric - Caeser cipher key) mod 26, if plainnumeric < 0 , plainnumeric = plainnumeric + 26

      iii. Use plainnumeric as the index position and get the corresponding character from the ALPHABET string as the equivalent plain text character for the cipher text character

   b. Print the equivalent plain text
5. Stop

PROGRAM:

```
import java.util.*;
import java.io.*;
public class Caesercipher
{
 public static final String ALPHABET = "abcdefghijklmnopqrstuvwxyz";
 public static String encrypt(String ptext, int cserkey)
 {
    String ctext = "";
    for (int i = 0; i < ptext.length(); i++)
    {
            int plainnumeric = ALPHABET.indexOf(ptext.charAt(i));
            int ciphernumeric = (plainnumeric+cserkey) % 26;
            char cipherchar = ALPHABET.charAt(ciphernumeric);
             ctext += cipherchar;
    }
 return ctext;
 }
 public static String decrypt(String ctext, int cserkey)
 {
    String ptext = "";
    for (int i = 0; i < ctext.length(); i++)
    {
            int ciphernumeric = ALPHABET.indexOf(ctext.charAt(i));
            int plainnumeric= (ciphernumeric-cserkey) % 26;
            if (plainnumeric < 0)
            {
                    plainnumeric = ALPHABET.length() + plainnumeric;
             }
            char plainchar = ALPHABET.charAt(plainnumeric);
                    ptext += plainchar;
     }
    return ptext;
    }
    public static void main(String[] args) throws IOException
    {
            BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
            System.out.println("Enter the PLAIN TEXT for Encryption: ");
            String plaintext = new String();
            String ciphertext = new String();
            String key;
            int cserkey;
            plaintext = br.readLine();
            System.out.println("Enter the CAESARKEY between 0 and 25:");
            key = br.readLine();
            cserkey = Integer.parseInt(key);

            System.out.println("ENCRYPTION");
            ciphertext = encrypt(plaintext,cserkey);
            System.out.println("CIPHER TEXT:"+ ciphertext);

            System.out.println("DECRYPTION");
            plaintext = decrypt(ciphertext,cserkey);
            System.out.println("PLAIN TEXT:" + plaintext);
    }
```

RESULT:

Thus, the program to implement Caesar cipher encryption technique was executed and the output was verified successfully.

Ex No:1b **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. Generate the (5 x 5) key table or matrix using the key.
   a. Fill the spaces in the table with the letters of the key without any duplication of letters.
   b. Fill the remaining spaces with the rest of the letters of the alphabet in order by having 'I' & 'J' in the same space or omitting 'Q' to reduce the alphabet to fit.
2. Remove any punctuation or characters from the plain text that are not present in the key square.
3. Identify any double letters in the plaintext and insert 'X' between the two occurrences.
4. Split the plain text into digraphs (groups of 2 letters)
5. **Encryption:** Locate the digraph letters in the key table
   a. If the letters appear on the same row of the table, replace them with the letters to their immediate right respectively (wrapping around to the left side of the row if a letter in the original pair was on the right side of the row).
   b. If the letters appear on the same column of the table, replace them with the letters immediately below respectively (wrapping around to the top side of the column if a letter in the original pair was on the bottom side of the column).
   c. If the letters are in different rows and columns, replace the pair with the letters on the same row respectively but at the other pair of corners of the rectangle defined by the original pair. The order is important – the first encrypted letter of the pair is the one that lies on the same row as the first plaintext letter.
   d. Append the letters referred from the key table using the steps 5.a, 5.b and 5.c to generate Cipher text.
6. **Decryption:** Split the cipher text into digraphs and locate the digraph letters in the key table.
   a. If the letters appear on the same row of the table, replace them with the letters to their immediate left respectively (wrapping around to the right side of the row if a letter in the original pair was on the left side of the row).
   b. If the letters appear on the same column of the table, replace them with the letters immediately above respectively (wrapping around to the bottom side of the column if a letter in the original pair was on the top side of the column).

c. If the letters are in different rows and columns, replace the pair with the letters on the same row respectively but at the other pair of corners of the rectangle defined by the original pair. The order is important – the first encrypted letter of the pair is the one that lies on the same row as the first plaintext letter.

d. Append the letters referred from the key table using the steps 6.a, 6.b and 6.c to generate Plain text.

7. Stop

PROGRAM:

```
import java.util.Scanner;
public class PlayfairCipherEncryption
{
    private String KeyWord        = new String();
    private String Key            = new String();
    private char   matrix_arr[][] = new char[5][5];

    public void setKey(String k)
    {
        String K_adjust = new String();
        boolean flag = false;
        K_adjust = K_adjust + k.charAt(0);
        for (int i = 1; i < k.length(); i++)
        {
            for (int j = 0; j < K_adjust.length(); j++)
            {
                if (k.charAt(i) == K_adjust.charAt(j))
                {
                    flag = true;
                }
            }
            if (flag == false)
                K_adjust = K_adjust + k.charAt(i);
            flag = false;
        }
        KeyWord = K_adjust;
    }
    public void KeyGen()
    {
        boolean flag = true;
        char current;
        Key = KeyWord;
        for (int i = 0; i < 26; i++)
        {
            current = (char) (i + 97);
            if (current == 'j')
                continue;
            for (int j = 0; j < KeyWord.length(); j++)
            {
                if (current == KeyWord.charAt(j))
                {
                    flag = false;
                    break;
                }
            }
            if (flag)
                Key = Key + current;
            flag = true;
        }
        System.out.println(Key);
        matrix();
    }
    private void matrix()
    {
        int counter = 0;
        for (int i = 0; i < 5; i++)
        {
            for (int j = 0; j < 5; j++)
            {
                matrix_arr[i][j] = Key.charAt(counter);
                System.out.print(matrix_arr[i][j] + " ");
                counter++;
            }
            System.out.println();
        }
    }
    private String format(String old_text)
    {
        int i = 0;
        int len = 0;
        String text = new String();
        len = old_text.length();
        for (int tmp = 0; tmp < len; tmp++)
        {
            if (old_text.charAt(tmp) == 'j')
            {
                text = text + 'i';
            }
            else
                text = text + old_text.charAt(tmp);
        }
        len = text.length();
        for (i = 0; i < len; i = i + 2)
        {
            if (text.charAt(i + 1) == text.charAt(i))
            {
                text = text.substring(0, i + 1) + 'x' + text.substring(i + 1);
            }
        }
        return text;
    }
    private String[] Divid2Pairs(String new_string)
    {
        String Original = format(new_string);
        int size = Original.length();
        if (size % 2 != 0)
        {
            size++;
            Original = Original + 'x';
        }
        String x[] = new String[size / 2];
        int counter = 0;
        for (int i = 0; i < size / 2; i++)
        {
            x[i] = Original.substring(counter, counter + 2);
            counter = counter + 2;
        }
        return x;
    }
    public int[] GetDiminsions(char letter)
    {
        int[] key = new int[2];
        if (letter == 'j')
            letter = 'i';
        for (int i = 0; i < 5; i++)
        {
            for (int j = 0; j < 5; j++)
            {
                if (matrix_arr[i][j] == letter)
                {
                    key[0] = i;
                    key[1] = j;
                    break;
                }
            }
        }
        return key;
    }

    public String encryptMessage(String Source)
    {
        String src_arr[] = Divid2Pairs(Source);
        String Code = new String();
        char one;
        char two;
        int part1[] = new int[2];
        int part2[] = new int[2];
        for (int i = 0; i < src_arr.length; i++)
        {
            one = src_arr[i].charAt(0);
            two = src_arr[i].charAt(1);
            part1 = GetDiminsions(one);
            part2 = GetDiminsions(two);
            if (part1[0] == part2[0])
        {
            if (part1[1] < 4)
                part1[1]++;
            else
                part1[1] = 0;
            if (part2[1] < 4)
                part2[1]++;
            else
                part2[1] = 0;
        }
        else if (part1[1] == part2[1])
        {
            if (part1[0] < 4)
                part1[0]++;
            else
                part1[0] = 0;
            if (part2[0] < 4)
                part2[0]++;
            else
                part2[0] = 0;
        }
        else
        {
            int temp = part1[1];
            part1[1] = part2[1];
            part2[1] = temp;
        }
        Code = Code + matrix_arr[part1[0]][part1[1]]
                + matrix_arr[part2[0]][part2[1]];
    }
    return Code;
}

public static void main(String[] args)
{
    PlayfairCipherEncryption x = new PlayfairCipherEncryption();
    Scanner sc = new Scanner(System.in);
    System.out.println("Enter a keyword:");
```

```
        String keyword = sc.next();
        x.setKey(keyword);
        x.KeyGen();
        System.out
                .println("Enter word to encrypt: (Make sure length of message is even)");
        String key_input = sc.next();
        if (key_input.length() % 2 == 0)
        {
            System.out.println("Encryption: " + x.encryptMessage(key_input));
        }
        else
        {
            System.out.println("Message length should be even");
        }
        sc.close();
    }
}
```

RESULT:

Thus, the program to implement Play Fair Cipher technique was executed and the output was verified successfully.

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

AIM:

To write a java program to encrypt and decrypt using the Hill cipher substitution technique.

ALGORITHM:

1. Get the n-by-n key matrix with vectors of length n.
2. Separate the plaintext from left to right into some number k of groups of n letters each. If you run out of letters when forming the final group, repeat the last plaintext letter or 'X' as many times as needed to fill out the final group of n letters.
3. Replace each letter by the corresponding number of its position (from 0 through m-1) in the alphabet to get k groups of n integers each.
4. **Encryption:**

   a. Reshape each of the k groups of integers into an n-row column vector called Plaintext matrix.

   b. **Cipher Text Matrix = (Plain Text Matrix * Key Matrix) mod 26.**

   c. Using step 4.b, perform matrix multiplication of Plain text matrix and Key matrix and modulus 26 to yield Cipher text matrix.

   d. Translate the Cipher text matrix to string representation by replacing each of the entries with the corresponding letter of the alphabet, after arranging all k of the resulting column vectors in order into a single vector of length k x n
5. **Decryption:**

   a. Find the inverse of the key matrix

      i. Find the determinant of the key matrix

      ii. Transpose the key matrix

      iii. Find minor matrix and then Cofactor of the key matrix

      iv. Key^{-1}= [[Det(key matrix)] $^{-1}$* Cofactor] mod 26 using modulus arithmetic

   b. Plain Text = (Cipher Text * Key^{-1}) mod 26

   c. Using step 5.b, perform matrix multiplication of Cipher text matrix and Key inverse matrix and modulus 26 to yield Plain text matrix.

   d. Translate the Plain text matrix to string representation by replacing each of the entries with the corresponding letter of the alphabet, after arranging all k of the resulting column vectors and removing any letters padded in order into a single vector of length k x n
6. Stop

PROGRAM:

```
import java.util.*;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
 public class HillCipherExample {
int[] lm;
    int[][] km;
    int[] rm;
    static int choice;
    int [][] invK;

    public void performDivision(String temp, int s)
    {
        while (temp.length() > s)
        {
            String line = temp.substring(0, s);
            temp = temp.substring(s, temp.length());
            calLineMatrix(line);
            if(choice ==1){
                multiplyLineByKey(line.length());
            }else{
                multiplyLineByInvKey(line.length());
            }
            showResult(line.length());
        }
        if (temp.length() == s)
          {

            if(choice ==1){
            calLineMatrix(temp);
            multiplyLineByKey(temp.length());
            showResult(temp.length());
            }
            else{
                calLineMatrix(temp);
                this.multiplyLineByInvKey(temp.length());
                showResult(temp.length());
            }
        }
        else if (temp.length() < s)
        {
            for (int i = temp.length(); i < s; i++)
                temp = temp + 'x';
            if(choice ==1){
            calLineMatrix(temp);
            multiplyLineByKey(temp.length());
            showResult(temp.length());
            }
            else{
                calLineMatrix(temp);
                multiplyLineByInvKey(temp.length());
                showResult(temp.length());
            }
        }
    }
    public void calKeyMatrix(String key, int len)
    {
        km = new int[len][len];
        int k = 0;
        for (int i = 0; i < len; i++)
        {
            for (int j = 0; j < len; j++)
            {
                km[i][j] = ((int) key.charAt(k)) - 97;
                k++;
            }
        }
    }
    public void calLineMatrix(String line)
    {
        lm = new int[line.length()];
        for (int i = 0; i < line.length(); i++)
        {
            lm[i] = ((int) line.charAt(i)) - 97;
        }
```

```
}
public void multiplyLineByKey(int len)
{
    rm = new int[len];
    for (int i = 0; i < len; i++)
    {
        for (int j = 0; j < len; j++)
        {
            rm[i] += km[i][j] * lm[j];
        }
        rm[i] %= 26;
    }
}
public void multiplyLineByInvKey(int len)
{
    rm = new int[len];
    for (int i = 0; i < len; i++)
    {
        for (int j = 0; j < len; j++)
        {
            rm[i] += invK[i][j] * lm[j];
        }
        rm[i] %= 26;
    }
}
public void showResult(int len)
{
    String result = "";
    for (int i = 0; i < len; i++)
    {
        result += (char) (rm[i] + 97);
    }
    System.out.print(result);
}
public int calDeterminant(int A[][], int N)
{
    int resultOfDet;
    switch (N) {
        case 1:
            resultOfDet = A[0][0];
            break;
        case 2:
            resultOfDet = A[0][0] * A[1][1] - A[1][0] * A[0][1];
            break;
        default:
            resultOfDet = 0;
            for (int j1 = 0; j1 < N; j1++)
            {
                int m[][] = new int[N - 1][N - 1];
                for (int i = 1; i < N; i++)
                {
                    int j2 = 0;
                    for (int j = 0; j < N; j++)
                    {
                        if (j == j1)
                            continue;
                        m[i - 1][j2] = A[i][j];
                        j2++;
                    }
                }
                resultOfDet += Math.pow(-1.0, 1.0 + j1 + 1.0) * A[0][j1]
                        * calDeterminant(m, N - 1);
            }   break;
    }
    return resultOfDet;
}
public void cofact(int num[][], int f)
{
    int b[][], fac[][];
    b = new int[f][f];
    fac = new int[f][f];
    int p, q, m, n, i, j;
    for (q = 0; q < f; q++)
    {
        for (p = 0; p < f; p++)
        {
            m = 0;
            n = 0;
            for (i = 0; i < f; i++)
            {
                for (j = 0; j < f; j++)
                {
                    b[i][j] = 0;
                    if (i != q && j != p)
                    {
                        b[m][n] = num[i][j];
                        if (n < (f - 2))
                            n++;
                        else
                        {
                            n = 0;
                            m++;
                        }
                    }
                }
            }
            fac[q][p] = (int) Math.pow(-1, q + p) * calDeterminant(b, f - 1);
        }
    }
    trans(fac, f);
}
void trans(int fac[][], int r)
{
    int i, j;
    int b[][], inv[][];
    b = new int[r][r];
    inv = new int[r][r];
    int d = calDeterminant(km, r);
    int mi = mi(d % 26);
    mi %= 26;
    if (mi < 0)
        mi += 26;
    for (i = 0; i < r; i++)
    {
        for (j = 0; j < r; j++)
        {
            b[i][j] = fac[j][i];
        }
    }
    for (i = 0; i < r; i++)
    {
        for (j = 0; j < r; j++)
        {
            inv[i][j] = b[i][j] % 26;
            if (inv[i][j] < 0)
                inv[i][j] += 26;
            inv[i][j] *= mi;
            inv[i][j] %= 26;
        }
    }
    //System.out.println("\nInverse key:");
    //matrixtoinvkey(inv, r);

    invK = inv;
}
public int mi(int d)
{
    int q, r1, r2, r, t1, t2, t;
    r1 = 26;
    r2 = d;
    t1 = 0;
    t2 = 1;
    while (r1 != 1 && r2 != 0)
    {
        q = r1 / r2;
        r = r1 % r2;
        t = t1 - (t2 * q);
        r1 = r2;
        r2 = r;
        t1 = t2;
        t2 = t;
    }
    return (t1 + t2);
}
public void matrixtoinvkey(int inv[][], int n)
{
    String invkey = "";
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            invkey += (char) (inv[i][j] + 97);
        }
    }
    System.out.print(invkey);
}
 public boolean check(String key, int len)
{
    calKeyMatrix(key, len);
    int d = calDeterminant(km, len);
    d = d % 26;
    if (d == 0)
    {
        System.out.println("Key is not invertible");
        return false;
    }
    else if (d % 2 == 0 || d % 13 == 0)
    {
        System.out.println("Key is not invertible");
        return false;
    }
    else
    {
        return true;
    }
}
public static void main(String args[]) throws IOException
{
    HillCipherExample obj = new HillCipherExample();
    BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
```

```
        System.out.println("Menu:\n1: Encryption\n2: Decryption");
        choice = Integer.parseInt(in.readLine());
        System.out.println("Enter the line: ");
        String line = in.readLine();
        System.out.println("Enter the key: ");
        String key = in.readLine();
        double sq = Math.sqrt(key.length());
        if (sq != (long) sq)
            System.out.println("Cannot Form a square matrix");
        else
        {
            int size = (int) sq;
            if (obj.check(key, size))
            {
                System.out.println("Result:");
                obj.cofact(obj.km, size);
                obj.performDivision(line, size);
            }
        }
    }
}
```

RESULT:

Thus, the program to implement Hill Cipher technique was executed and the output was verified successfully.

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

AIM:

To develop a java program to implement encryption and decryption using Vigenère cipher substitution technique

ALGORITHM:

1. Vigenère table 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.
2. The Key is repeated so that the total length is equal to that of the plaintext. In this way, each letter in the plaintext is shifted by the alphabet number of the corresponding letter in the key.
3. At different points in the encryption process, the cipher uses a different alphabet from one of the rows used.
4. **Encryption:** The the plaintext(P) and key(K) are added modulo 26.

   $E_i = (P_i + K_i) \bmod 26$
5. **Decryption:** Subtract the key from the Encrypted (Cipher) text and perform modulo 26 to arrive back at the original, plaintext value

   $D_i = (E_i - K_i + 26) \bmod 26$
6. Stop

PROGRAM:

```
public class VigenereCipher
{
   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)
    {
        String key = "VIGENERECIPHER";
        String message = "Beware the Jabberwock, my son! The jaws that bite, the claws that
catch!";
        String encryptedMsg = encrypt(message, key);
        System.out.println("String: " + message);
        System.out.println("Encrypted message: " + encryptedMsg);
        System.out.println("Decrypted message: " + decrypt(encryptedMsg, key));
    }
}
```

RESULT:

Thus, the program to implement Vigenere Cipher technique was executed and the output was verified successfully.

Ex No:2a **Date:**	**RAIL FENCE TRANSPOSITION TECHNIQUE**

AIM:

To develop a java program for implementing encryption and decryption using rail fence transposition technique.

ALGORITHM:

1. Generate numerical key from the word key by the characters of the word in alphabetical order.
2. Encryption
   i. The plain text is written in the matrix form, where the column of the matrix is number of characters in the word key and row of the matrix is to accommodate the characters of the plain text and the space left after the plain text characters in the last row is filled with any character. (eg. x or z)
   ii. The cipher text is generated by reading the characters column by column in the order specified in the numerical key.
3. Decryption
   i. The characters in the cipher text are filled in the matrix of same order used for encryption, but in the order specified in the key. The characters from cipher text equal to the number of rows in matrix are taken and filled in the matrix column based on the order specified in the key.
   ii. The plain text is generated from cipher text by reading the characters from the matrix row by row.
4. Stop

PROGRAM:

```
import java.util.*;
class RailFenceBasic
{
 int depth;
String Encryption(String plainText,int depth)throws Exception
{
        int r=depth,len=plainText.length();
        int c=len/depth;
         char mat[][]=new char[r][c];
          int k=0;
 String cipherText="";

 for(int i=0;i< c;i++)
 {
  for(int j=0;j< r;j++)
  {
   if(k!=len)
    mat[j][i]=plainText.charAt(k++);
   else
    mat[j][i]='X';
  }
 }
 for(int i=0;i< r;i++)
 {
  for(int j=0;j< c;j++)
  {
   cipherText+=mat[i][j];
  }
 }
 return cipherText;
}
String Decryption(String cipherText,int depth)throws Exception
{
 int r=depth,len=cipherText.length();
 int c=len/depth;
 char mat[][]=new char[r][c];
 int k=0;

 String plainText="";
 for(int i=0;i< r;i++)
 {
  for(int j=0;j< c;j++)
  {
   mat[i][j]=cipherText.charAt(k++);
  }
 }
 for(int i=0;i< c;i++)
```

```
           {
            for(int j=0;j< r;j++)
            {
             plainText+=mat[j][i];
            }
           }
           return plainText;
          }
         }
public class RailFence
{
         public static void main(String args[])throws Exception
         {
           RailFenceBasic rf=new RailFenceBasic();
                        Scanner scn=new Scanner(System.in);
                        int depth;
                        String plainText,cipherText,decryptedText;
                        System.out.println("Enter plain text:");
                        plainText=scn.nextLine();
                        System.out.println("Enter depth for Encryption:");
                        depth=scn.nextInt();
                        cipherText=rf.Encryption(plainText,depth);
                        System.out.println("Encrypted text is:\n"+cipherText);
                        decryptedText=rf.Decryption(cipherText, depth);
           System.out.println("Decrypted text is:\n"+decryptedText);
          }
         }
```

RESULT:

Thus, the program for Rail fence cipher was executed and verified successfully.

Ex No:2b Date:	**ROW AND COLUMNAR TRANSPOSITION TECHNIQUE**

AIM:

To write a java program to implement encryption and decryption using rail fence transposition technique.

ALGORITHM:

1. The message is written out in rows of a fixed length.
2. Read out again column by column, and the columns are chosen in some scrambled order.
3. Width of the rows and the permutation of the columns are usually defined by a keyword.
4. Finally, the message is read off in columns, in the order specified by the keyword.

PROGRAM:

```
import java.util.*;
import java.io.*;
import java.lang.*;
public class columnarTranspose
{
    public static void main(String[] args)

{
        Scanner scan = new Scanner(System.in);
        String line = System.getProperty("line.separator");
        scan.useDelimiter(line);
        System.out.print("1. Encryt 2.Decrypt : ");
        int option = scan.nextInt();
        switch (option)
{
            case 1:
                System.out.print("Enter String:");
                String text = scan.next();
                System.out.print("Enter Key:");
                String key = scan.next();
                System.out.println(encryptCT(key, text).toUpperCase());
                break;
            case 2:
                System.out.print("Enter Encrypted String:");
            text = scan.next();
            System.out.print("Enter Key:");
            key = scan.next();
            System.out.println(decryptCT(key, text));
            break;
        default:
            break;
      }
   }
   public static String encryptCT(String key, String text)
{
      int[] arrange = arrangeKey(key);
      int lenkey = arrange.length;
      int lentext = text.length();
      int row = (int) Math.ceil((double) lentext / lenkey);
      char[][] grid = new char[row][lenkey];
      int z = 0;
      for (int x = 0; x < row; x++) {
         for (int y = 0; y < lenkey; y++) {
            if (lentext == z) {
               // at random alpha for trailing null grid
               grid[x][y] = RandomAlpha();
               z--;
            } else {
               grid[x][y] = text.charAt(z);
            }

            z++;
         }
      }
      String enc = "";
      for (int x = 0; x < lenkey; x++) {
         for (int y = 0; y < lenkey; y++) {
            if (x == arrange[y]) {
               for (int a = 0; a < row; a++) {
                  enc = enc + grid[a][y];
               }
            }
          }
      }
      return enc;
   }
   public static String decryptCT(String key, String text)
{
      int[] arrange = arrangeKey(key);
      int lenkey = arrange.length;
      int lentext = text.length();
      int row = (int) Math.ceil((double) lentext / lenkey);
      String regex = "(?<=\\G.{" + row + "})";
      String[] get = text.split(regex);
      char[][] grid = new char[row][lenkey];
      for (int x = 0; x < lenkey; x++) {
          for (int y = 0; y < lenkey; y++) {
            if (arrange[x] == y) {
               for (int z = 0; z < row; z++) {
                  grid[z][y] = get[arrange[y]].charAt(z);
               }
            }
          }
      }
      String dec = "";
      for (int x = 0; x < row; x++) {
          for (int y = 0; y < lenkey; y++) {
            dec = dec + grid[x][y];
          }
      }
      return dec;
   }
   public static char RandomAlpha()
{
      //generate random alpha for null space
      Random r = new Random();
      return (char)(r.nextInt(26) + 'a');
   }
   public static int[] arrangeKey(String key)
{
      //arrange position of grid
      String[] keys = key.split("");
      Arrays.sort(keys);
      int[] num = new int[key.length()];
      for (int x = 0; x < keys.length; x++) {
          for (int y = 0; y < key.length(); y++) {
            if (keys[x].equals(key.charAt(y) + "")) {
               num[y] = x;
               break;
            }
          }
      }
      return num;
   }
}
```

RESULT:

Thus, the program for row and columnar transposition was executed and verified successfully.

Ex No:3 **Date:**	**DES ALGORITHM**

AIM:

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

ALGORITHM:

1. **Process the key.**
   i. Get a 64-bit key from the user.
   ii. Calculate the key schedule.
2. Perform the following permutation on the 64-bit key. The parity bits are discarded, reducing the key to 56 bits. Bit 1 of the permuted block is bit 57 of the original key, bit 2 is bit 49, and so on with bit 56 being bit 4 of the original key.
3. Split the permuted key into two halves. The first 28 bits are called C[0] and the last 28 bits are called D[0].
4. 3. Calculate the 16 subkeys. Start with i = 1.

   1. Perform one or two circular left shifts on both C[i-1] and D[i-1] to get C[i] and D[i], respectively. The number of shifts per iteration are given in the table below.

   Iteration # 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

   Left Shifts 1 1 2 2 2 2 2 2 1 2 2 2 2 2 2 1

   2. Permute the concatenation C[i]D[i] as indicated below. This will yield K[i], which is 48 bits long.

   3. Loop back to 1.ii.c.1 until K[16] has been calculated.

2 Process a 64-bit data block.

i. Get a 64-bit data block. If the block is shorter than 64 bits, it should be padded as appropriate for the application.

ii. Perform the initial permutation on the data block.

iii. Split the block into two halves. The first 32 bits are called L[0], and the last 32 bits are called R[0].

iv. Apply the 16 subkeys to the data block. Start with i = 1.

a. Expand the 32-bit R[i-1] into 48 bits according to the bit-selection function Expansion (E)

b. Exclusive-or E(R[i-1]) with K[i].

c. Break E(R[i-1]) xor K[i] into eight 6-bit blocks. Bits 1-6 are B[1], bits 7-12 are B[2], and so on with bits 43-48 being B[8].

d. Substitute the values found in the S-boxes for all B[j]. Start with j = 1. All values in the S-boxes should be considered 4 bits wide.

i. Take the 1st and 6th bits of B[j] together as a 2-bit value (call it m) indicating the row in S[j] to look in for the substitution.

ii. Take the 2nd through 5th bits of B[j] together as a 4-bit value(call it n) indicating the column in S[j] to find the substitution.

iii. Replace B[j] with S[j][m][n].

iv. Loop back to 2.iv.d.i until all 8 blocks have been replaced.

e. Permute the concatenation of B[1] through B[8]

f. Exclusive-or the resulting value with L[i-1]. Thus, all together, your R[i] = L[i-1] xor P(S[1](B[1])...S[8](B[8])), where B[j] is a 6-bit block of E(R[i-1]) xor K[i]. (The function for R[i] is written as, R[i] = L[i-1] xor f(R[i-1], K[i]).)

g. L[i] = R[i-1].

h. Loop back to 2.iv.a until K[16] has been applied.

v. Perform the final permutation on the block R[16]L[16].

3.Decryption : Use the keys K[i] in reverse order. That is, instead of applying K[1] for the first iteration, apply K[16], and then K[15] for the second, on down to K[1]

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,encryptedData,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();
intnum = r.nextInt(10000);
String knum = String.valueOf(num);
byte[] knumb = knum.getBytes();
skey=getRawKey(knumb);
skeyString = new String(skey);
System.out.println("DES Symmetric key = "+skeyString);
}
catch(Exception e) {
System.out.println(e);
}
}
private static byte[] getRawKey(byte[] seed) throws Exception {
KeyGeneratorkgen = KeyGenerator.getInstance("DES");
SecureRandomsr = SecureRandom.getInstance("SHA1PRNG");
sr.setSeed(seed);
kgen.init(56, sr);
SecretKeyskey = kgen.generateKey();
raw = skey.getEncoded();
return raw;
}
private static byte[] encrypt(byte[] raw, byte[] clear) throws Exception {
SecretKeySpecskeySpec = new SecretKeySpec(raw, "DES");
Cipher cipher = Cipher.getInstance("DES");
cipher.init(Cipher.ENCRYPT_MODE, skeySpec);
byte[] encrypted = cipher.doFinal(clear);
return encrypted;
}
```

```
private static byte[] decrypt(byte[] raw, byte[] encrypted) throws Exception {
SecretKeySpecskeySpec = new SecretKeySpec(raw, "DES");
Cipher cipher = Cipher.getInstance("DES");
cipher.init(Cipher.DECRYPT_MODE, skeySpec);
byte[] decrypted = cipher.doFinal(encrypted);
return decrypted;
}
public static void main(String args[]) {
DES des = new DES();
}
}
```

RESULT:

Thus, the program to implement DES algorithm was executed and verified successfully.

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

AIM:

To develop a java program to implement Advanced Encryption Standard for encryption and decryption.

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) {
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);
}
}
```

RESULT:

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

Ex No:5 Date:	**RSA**

AIM:

To implement RSA algorithm using Html and JavaScript.

ALGORITHM:

1. Choose two prime number p and q.
2. Compute the value of n and p
3. Find the totient of n, ϕ(n) ϕ(n)=(p-1)(q-1)
4. Find the value of e (public key), 1 < e < ϕ(n), and such that e and ϕ(n) share no divisors other than 1 (e and ϕ(n) are relatively prime)
5. Determine d (using modular arithmetic) which satisfies the congruence relation

   $d^e \equiv 1 \pmod{\phi(n)}$.
6. Do the encryption and decryption
   a. Encryption is given as,

      $c = t^e \bmod n$

   b. Decryption is given as,

      $t = c^d \bmod n$
7. Stop.

PROGRAM:

```
<html>
<head>
<title>RSA Encryption</title>
<meta name="viewport" content="width=device-width, initialscale=1.0">
</head>
<body>
<h1 style="text-align: center;">RSA Algorithm</h1>
<h2 style="text-align: center;">Implemented Using HTML & Javascript</h2>
<hr>
<table class="center">
<tr>
<td>Enter P:</td>
<td><input type="number" value="53" id="p"></td>
</tr>
<tr>
<td>Enter Q :</td>
<td><input type="number" value="59" id="q"></p>
```

```
</td>
</tr>
<tr>
<td>Enter the Message:<br>[A=1, B=2,...]</td>
<td><input type="number" value="89" id="msg"></p>
</td>
</tr>
<tr>
<td>Public Key(N):</td>
<td>
<p id="publickey(N)"></p>
</td>
</tr>
<tr>
<td>Exponent(e):</td>
<td>
<p id="exponent(e)"></p>
</td>
</tr>
<tr>
<td>Private Key(d):</td>
<td>
<p id="privatekey(d)"></p>
</td>
</tr>
<tr>
<td>Cipher Text(c):</td>
<td>
<p id="ciphertext(ct)"></p>
</td>
</tr>
<tr>
   <td><button onclick="RSA();">Apply RSA</button></td>
   </tr>
   </table>
   </body>
   <style>
      .center {
```

```
  margin-left: auto;
  margin-right: auto;
}
    </style>
    <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(N)').innerHTML = n;
document.getElementById('exponent(e)').innerHTML = e;
document.getElementById('privatekey(d)').innerHTML = d;
document.getElementById('ciphertext(ct)').innerHTML = ct;
}
</script>
</html>
```

RESULT:

Thus, the program to implement RSA algorithm was executed and verified successfully.

Ex No:6 **Date:**	**DIFFIE-HELLMAN KEY EXCHANGE**

AIM:

To write a java program to implement Diffie-Hellman Key Exchange algorithm.

ALGORITHM:

1. Global Public Elements: Let q be a prime number and 〈 where 〈 < q and 〈 is a primitive root of q.
2. User A Key Generation:
   i. Select private XA where XA < q
   ii. ii. Calculate public YA where YA = 〈 XA mod q
3. User B Key Generation:
   i. Select private XB where XB < q
   ii. iCalculate public YB where YB = 〈 XB mod q
4. Calculation of Secret Key by User A: K = (YB) XA mod q
5. Calculation of Secret Key by User B: K = (YA) XB mod q

PROGRAM:

```
import java.util.*;
class DiffieHellman
{
public static void main(String args[])
{
Scanner sc=new Scanner(System.in);
 System.out.println("Enter the value of modulo(p)");
int p=sc.nextInt();
System.out.println("Enter the root of "+p);
 int g=sc.nextInt();
System.out.println("select 1st secret no of Alice");
int a=sc.nextInt();
 System.out.println("select 2nd secret no of Bob");
int b=sc.nextInt();
int A = (int)Math.pow(g,a)%p;
int B = (int)Math.pow(g,b)%p;
int S_A = (int)Math.pow(B,a)%p;
int S_B =(int)Math.pow(A,b)%p;
if(S_A==S_B) {
 System.out.println("They shared a secret key is = "+S_A); }
else { System.out.println("Alice and Bob cannot exchange information with each other");
}
}
}
```

RESULT:

Thus, the program to implement Diffie-Hellman Key Exchange algorithm was executed and verified successfully.

Ex No:7 Date:	**IMPLEMENT SECURE HASH ALGORITHM (SHA)**

AIM:

To write a java program to implement secure hash Algorithm (SHA).

ALGORITHM:

1. Append Padding Bits: Message is “padded” with a 1 and as many 0’s as necessary to bring the message length to 64 bits less than an even multiple of 512.
2. Append Length: 64 bits are appended to the end of the padded message. These bits hold the binary format of 64 bits indicating the length of the original message.
3. Prepare Processing Functions: SHA1 requires 80 processing functions defined as:

   f(t;B,C,D) = (B AND C) OR ((NOT B) AND D) (0 <= t <= 19)

   f(t;B,C,D) = B XOR C XOR D (20 <= t <= 39)

   f(t;B,C,D) = (B AND C) OR (B AND D) OR (C AND D) (40 <= t<=59)

   f(t;B,C,D) = B XOR C XOR D (60 <= t <= 79)
4. Prepare Processing Constants: SHA1 requires 80 processing constant words defined as:

   K(t) = 0x5A827999 (0 <= t <= 19)

   K(t) = 0x6ED9EBA1 (20 <= t <= 39)

   K(t) = 0x8F1BBCDC (40 <= t <= 59)

   K(t) = 0xCA62C1D6 (60 <= t <= 79)
5. Initialize Buffers: SHA1 requires 160 bits or 5 buffers of words (32 bits):

   H0 = 0x67452301 H1 = 0xEFCDAB89

   H2 = 0x98BADCFE H3 = 0x10325476

   H4 = 0xC3D2E1F0
6. Processing Message in 512-bit blocks (L blocks in total message)

   i. This is the main task of SHA1 algorithm which loops through the padded and appended message in 512-bit blocks.

   ii. Input and predefined functions: M[1, 2, ..., L]: Blocks of the padded and appended message f(0;B,C,D), f(1,B,C,D), ..., f(79,B,C,D): 80

   Processing Functions

   K(0), K(1), ..., K(79): 80 Processing Constant Words

   H0, H1, H2, H3, H4, H5: 5 Word buffers with initial values
7. For loop on k = 1 to L

   1. (W(0),W(1),...,W(15)) = M[k] /* Divide M[k] into 16 words */
8. For t = 16 to 79 do:

   W(t) = (W(t-3) XOR W(t-8) XOR W(t-14) XOR W(t-16)) <<< 1

   A = H0, B = H1, C = H2, D = H3, E = H4

```
    For t = 0 to 79 do:
        TEMP = A<<<5 + f(t;B,C,D) + E + W(t) + K(t)
        E = D, D = C, C = B<<<30, B = A, A = TEMP
    End of for loop
    H0 = H0 + A, H1 = H1 + B, H2 = H2 + C, H3 = H3 + D, H4 = H4 + E
9.  End of for loop
```

PROGRAM:

```
import java.security.*;
public class SHA1 {
public static void main(String[] a) {
try {
MessageDigest md = MessageDigest.getInstance("SHA1");
String input = "ssm";
md.update(input.getBytes());
byte[] output = md.digest();
System.out.println();
System.out.println("SHA1(\""+input+"\") = " +bytesToHex(output));
input = "iet";
md.update(input.getBytes());
output = md.digest();
System.out.println();
System.out.println("SHA1(\""+input+"\") = " +bytesToHex(output));
input = "security";
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);
}
}
public 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 (int j=0; j<b.length; j++) {
buf.append(hexDigit[(b[j] >> 4) & 0x0f]);
buf.append(hexDigit[b[j] & 0x0f]); }
return buf.toString(); }
}
```

RESULT:

Thus, the program to implement Secure Hash Algorithm was executed and verified successfully.

Ex No:8 Date:	**IMPLEMENT DIGITAL SIGNATURE SCHEME**

AIM:

To write a java program to implement the SIGNATURE SCHEME - Digital Signature Standard.

ALGORITHM:

1. Choose a prime number q, which is called the prime divisor.
2. Choose another primer number p, such that p-1 mod q = 0. p is called the prime modulus.
3. Choose an integer g, such that 1 < g < p, g**q mod p = 1 and g = h**((p–1)/q) mod p. q is also called g's multiplicative order modulo p.
4. Choose an integer, such that 0 < x < q.
5. Compute y as g**x mod p.
6. Package the public key as {p,q,g,y}, {p,q,g,x}.
7. Generate the message digest h, using a hash algorithm like SHA1.
8. Generate a random number k, such that 0 < k < q.
9. Compute r as (g**k mod p) mod q. If r = 0, select a different k.
10. Compute i, such that k*i mod q = 1. i is called the modular multiplicative inverse of k modulo q.
11. Compute s = i*(h+r*x) mod q. If s = 0, select a different k.
12. Package the digital signature as {r,s}.
13. Generate the message digest h, using the same hash algorithm.
14. 1Compute w, such that s*w mod q = 1. w is called the modular multiplicative inverse of s modulo q.
15. Compute u1 = h*w mod q. Compute u2 = r*w mod q.
16. Compute v = (((g**u1)*(y**u2)) mod p) mod q.
17. If v == r, the digital signature is valid.

PROGRAM:

```
import java.util.*;
import java.math.BigInteger;
public class dsaAlg
{
final static BigInteger one = new BigInteger("1");
final static BigInteger zero = new BigInteger("0");
/* incrementally tries for next prime */
public static BigInteger getNextPrime(String ans)
{
```

```
BigInteger test = new BigInteger(ans);
while (!test.isProbablePrime(99))
{
test = test.add(one);
}
return test;
}
/* finds largest prime factor of n */
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;
}
/* finds a generator mod p */
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();
/* establish the global public key components */
BigInteger p = getNextPrime("10600"); /* approximate prime */
BigInteger q = findQ(p.subtract(one));
BigInteger g = getGen(p,q,randObj);
/* public key components */
System.out.println("Digital Signature Algorithm");
System.out.println("global public key components are:");
System.out.println("p is: " + p);
System.out.println("q is: " + q);
System.out.println("g is: " + g);
/* find the private key */
BigInteger x = new BigInteger(q.bitLength(), randObj);
x = x.mod(q);
/* corresponding public key */
BigInteger y = g.modPow(x,p);
/* random value message */
BigInteger k = new BigInteger(q.bitLength(), randObj);
k = k.mod(q);
/* randomly generated hash value and digital signature */
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);
/* secret information */
System.out.println("secret information are:");
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("Generating digital signature:");
System.out.println("r is : " + r);
System.out.println("s is : " + s);
/*verify the digital signature */
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("verifying digital signature (checkpoints):");
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("success: digital signature is verified! " + r);
}
else
{
System.out.println("error: incorrect digital signature");
}
}
}
```

RESULT:

Thus, the program to implement Digital Signature was developed and executed successfully.

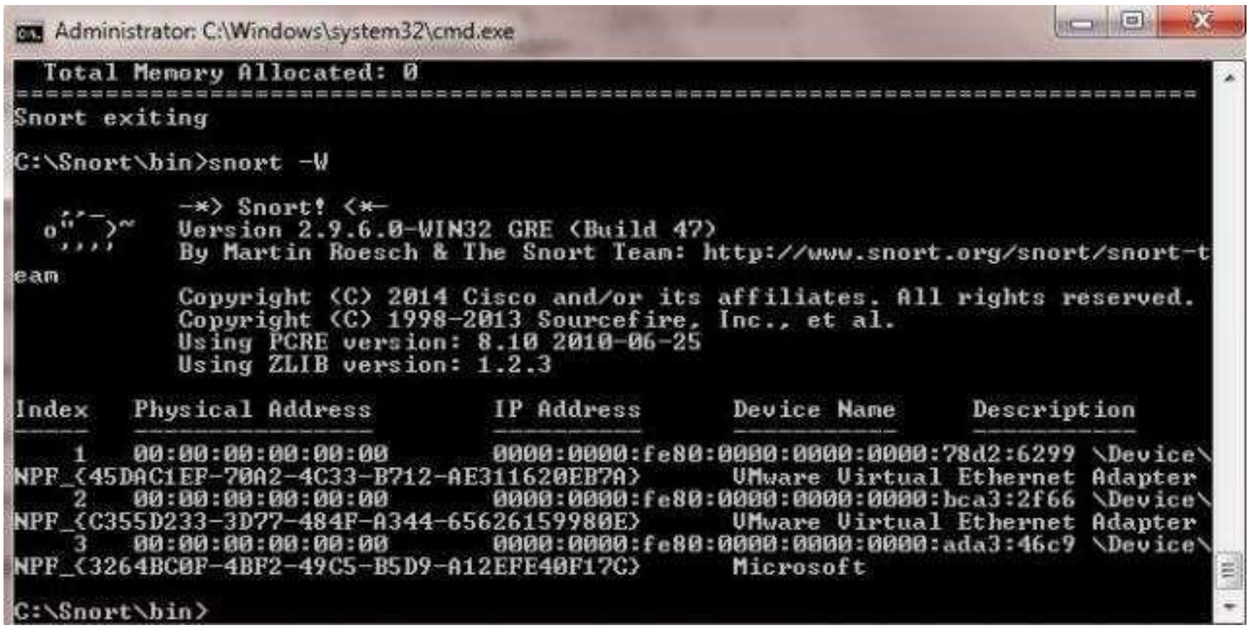
Ex No:9 Date:	**DEMONSTRATE INTRUSION DETECTION SYSTEM (IDs) USING ANY TOOL (SNORT OR ANY OTHER S/W)**

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, 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



```
  Total Memory Allocated: 0
===============================================================================
Snort exiting

C:\Snort\bin>snort -W

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.6.0-WIN32 GRE (Build 47)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/snort/snort-t
eam
           Copyright (C) 2014 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using PCRE version: 8.10 2010-06-25
           Using ZLIB version: 1.2.3

Index   Physical Address        IP Address        Device Name      Description
-----   ----------------        ----------        -----------      -----------
    1   00:00:00:00:00:00       0000:0000:fe80:0000:0000:0000:78d2:6299 \Device\
NPF_{45DAC1EF-70A2-4C33-B712-AE311620EB7A}        VMware Virtual Ethernet Adapter
    2   00:00:00:00:00:00       0000:0000:fe80:0000:0000:0000:bca3:2f66 \Device\
NPF_{C355D233-3D77-484F-A344-65626159980E}        VMware Virtual Ethernet Adapter
    3   00:00:00:00:00:00       0000:0000:fe80:0000:0000:0000:ada3:46c9 \Device\
NPF_{3264BC0F-4BF2-49C5-B5D9-A12EFE40F17C}        Microsoft

C:\Snort\bin>
```

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.

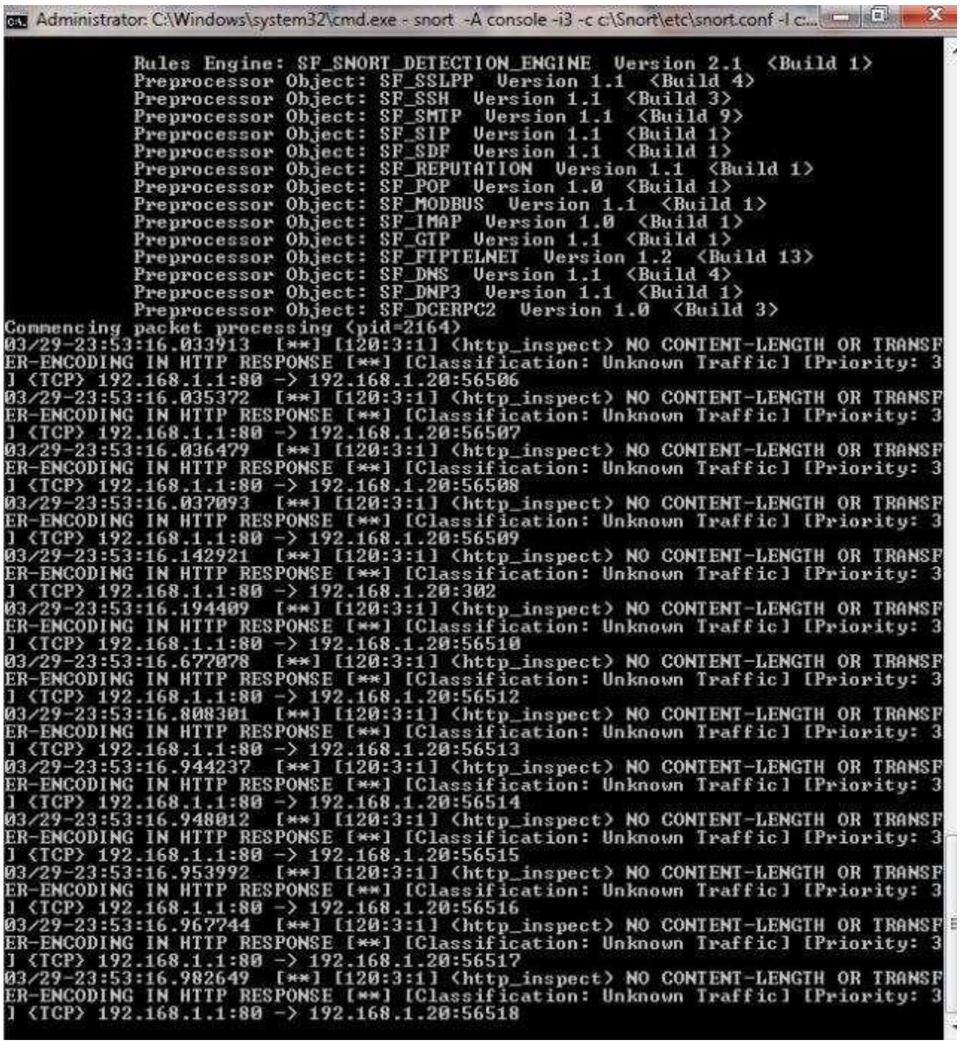
    To generate Log files in ASCII mode, you can use following command while running snort in IDS mode:

    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 –



```
Administrator: C:\Windows\system32\cmd.exe - snort  -A console -i3 -c c:\Snort\etc\snort.conf -l c...
           Rules Engine: SF_SNORT_DETECTION_ENGINE  Version 2.1  <Build 1>
           Preprocessor Object: SF_SSLPP  Version 1.1  <Build 4>
           Preprocessor Object: SF_SSH  Version 1.1  <Build 3>
           Preprocessor Object: SF_SMTP  Version 1.1  <Build 9>
           Preprocessor Object: SF_SIP  Version 1.1  <Build 1>
           Preprocessor Object: SF_SDF  Version 1.1  <Build 1>
           Preprocessor Object: SF_REPUTATION  Version 1.1  <Build 1>
           Preprocessor Object: SF_POP  Version 1.0  <Build 1>
           Preprocessor Object: SF_MODBUS  Version 1.1  <Build 1>
           Preprocessor Object: SF_IMAP  Version 1.0  <Build 1>
           Preprocessor Object: SF_GTP  Version 1.1  <Build 1>
           Preprocessor Object: SF_FTPTELNET  Version 1.2  <Build 13>
           Preprocessor Object: SF_DNS  Version 1.1  <Build 4>
           Preprocessor Object: SF_DNP3  Version 1.1  <Build 1>
           Preprocessor Object: SF_DCERPC2  Version 1.0  <Build 3>
Commencing packet processing (pid=2164)
03/29-23:53:16.033913  [**] [120:3:1] (http_inspect) NO CONTENT-LENGTH OR TRANSF
ER-ENCODING IN HTTP RESPONSE [**] [Classification: Unknown Traffic] [Priority: 3
] {TCP} 192.168.1.1:80 -> 192.168.1.20:56506
03/29-23:53:16.035372  [**] [120:3:1] (http_inspect) NO CONTENT-LENGTH OR TRANSF
ER-ENCODING IN HTTP RESPONSE [**] [Classification: Unknown Traffic] [Priority: 3
] {TCP} 192.168.1.1:80 -> 192.168.1.20:56507
03/29-23:53:16.036479  [**] [120:3:1] (http_inspect) NO CONTENT-LENGTH OR TRANSF
ER-ENCODING IN HTTP RESPONSE [**] [Classification: Unknown Traffic] [Priority: 3
] {TCP} 192.168.1.1:80 -> 192.168.1.20:56508
03/29-23:53:16.037093  [**] [120:3:1] (http_inspect) NO CONTENT-LENGTH OR TRANSF
ER-ENCODING IN HTTP RESPONSE [**] [Classification: Unknown Traffic] [Priority: 3
] {TCP} 192.168.1.1:80 -> 192.168.1.20:56509
03/29-23:53:16.142921  [**] [120:3:1] (http_inspect) NO CONTENT-LENGTH OR TRANSF
ER-ENCODING IN HTTP RESPONSE [**] [Classification: Unknown Traffic] [Priority: 3
] {TCP} 192.168.1.1:80 -> 192.168.1.20:302
03/29-23:53:16.194409  [**] [120:3:1] (http_inspect) NO CONTENT-LENGTH OR TRANSF
ER-ENCODING IN HTTP RESPONSE [**] [Classification: Unknown Traffic] [Priority: 3
] {TCP} 192.168.1.1:80 -> 192.168.1.20:56510
03/29-23:53:16.677078  [**] [120:3:1] (http_inspect) NO CONTENT-LENGTH OR TRANSF
ER-ENCODING IN HTTP RESPONSE [**] [Classification: Unknown Traffic] [Priority: 3
] {TCP} 192.168.1.1:80 -> 192.168.1.20:56512
03/29-23:53:16.808301  [**] [120:3:1] (http_inspect) NO CONTENT-LENGTH OR TRANSF
ER-ENCODING IN HTTP RESPONSE [**] [Classification: Unknown Traffic] [Priority: 3
] {TCP} 192.168.1.1:80 -> 192.168.1.20:56513
03/29-23:53:16.944237  [**] [120:3:1] (http_inspect) NO CONTENT-LENGTH OR TRANSF
ER-ENCODING IN HTTP RESPONSE [**] [Classification: Unknown Traffic] [Priority: 3
] {TCP} 192.168.1.1:80 -> 192.168.1.20:56514
03/29-23:53:16.948012  [**] [120:3:1] (http_inspect) NO CONTENT-LENGTH OR TRANSF
ER-ENCODING IN HTTP RESPONSE [**] [Classification: Unknown Traffic] [Priority: 3
] {TCP} 192.168.1.1:80 -> 192.168.1.20:56515
03/29-23:53:16.953992  [**] [120:3:1] (http_inspect) NO CONTENT-LENGTH OR TRANSF
ER-ENCODING IN HTTP RESPONSE [**] [Classification: Unknown Traffic] [Priority: 3
] {TCP} 192.168.1.1:80 -> 192.168.1.20:56516
03/29-23:53:16.967744  [**] [120:3:1] (http_inspect) NO CONTENT-LENGTH OR TRANSF
ER-ENCODING IN HTTP RESPONSE [**] [Classification: Unknown Traffic] [Priority: 3
] {TCP} 192.168.1.1:80 -> 192.168.1.20:56517
03/29-23:53:16.982649  [**] [120:3:1] (http_inspect) NO CONTENT-LENGTH OR TRANSF
ER-ENCODING IN HTTP RESPONSE [**] [Classification: Unknown Traffic] [Priority: 3
] {TCP} 192.168.1.1:80 -> 192.168.1.20:56518
```

RESULT:

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

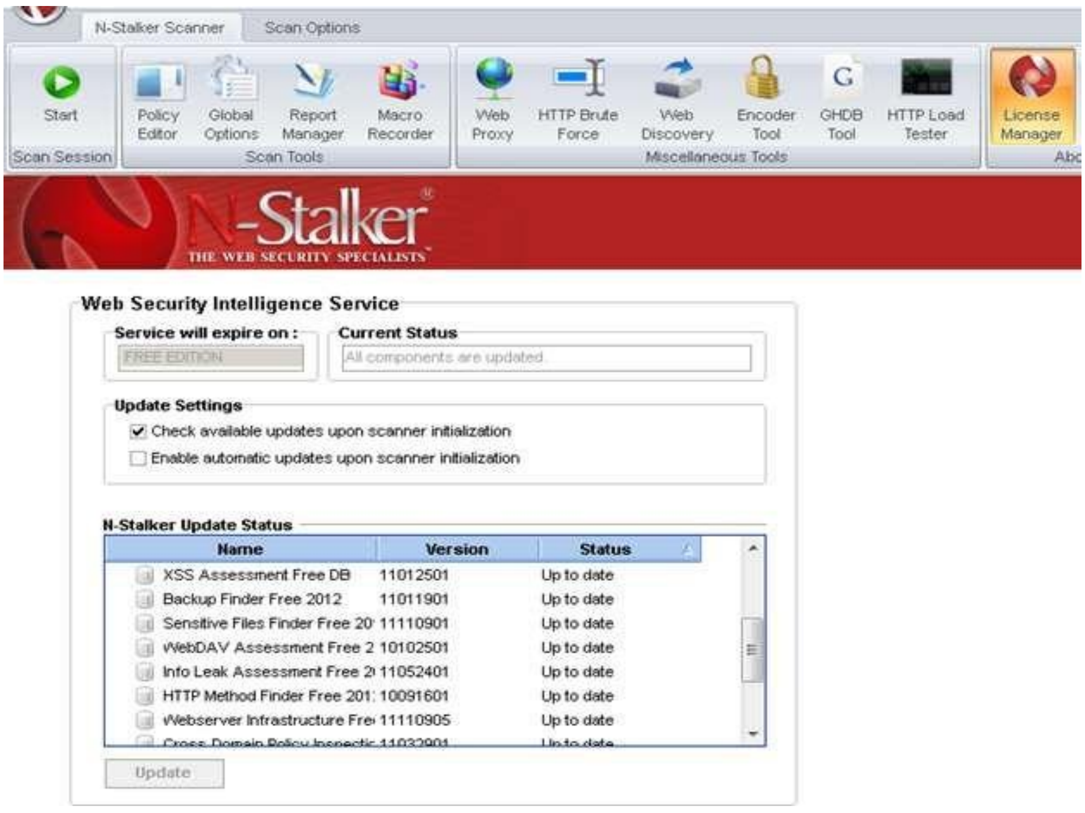
Ex No:10 **Date:**	**AUTOMATED ATTACK AND PENETRATION TOOLS** **EXPLORING N-STALKER, A VULNERABILITY ASSESSMENT TOOL**

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.



Web Security Intelligence Service

Service will expire on : FREE EDITION

Current Status: All components are updated

Update Settings

✓ Check available updates upon scanner initialization

☐ Enable automatic updates upon scanner initialization

N-Stalker Update Status

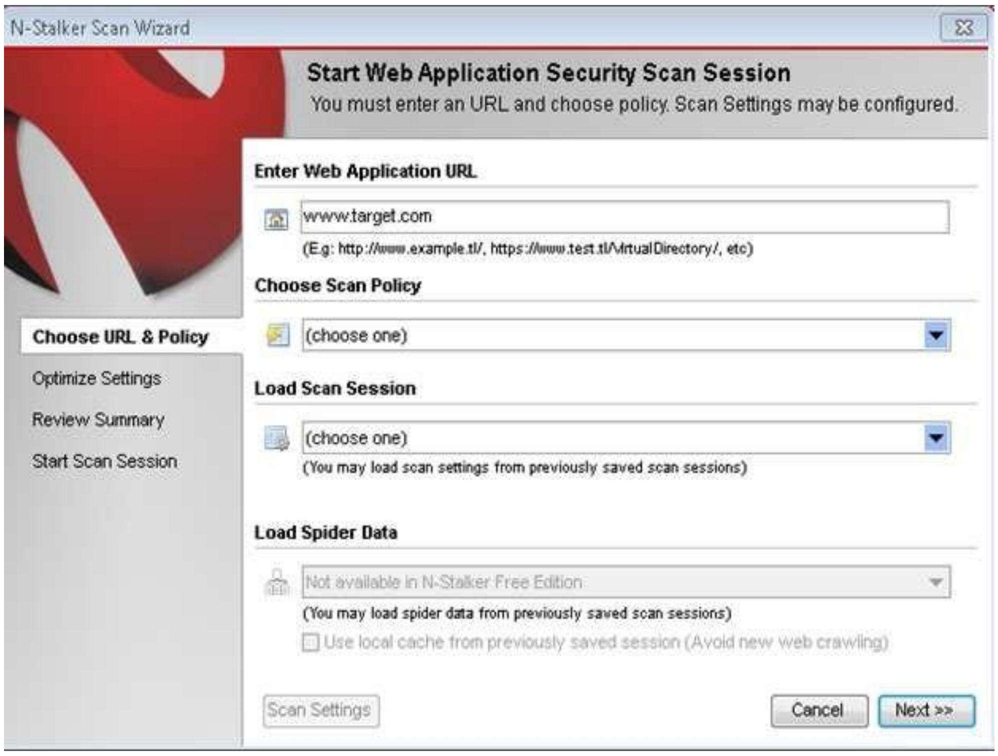
Name	Version	Status
XSS Assessment Free DB	11012501	Up to date
Backup Finder Free 2012	11011901	Up to date
Sensitive Files Finder Free 20	11110901	Up to date
WebDAV Assessment Free 2	10102501	Up to date
Info Leak Assessment Free 2	11052401	Up to date
HTTP Method Finder Free 201	10091601	Up to date
Webserver Infrastructure Fre	11110905	Up to date
Cross Domain Policy Inspectic	11032901	Up to date

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

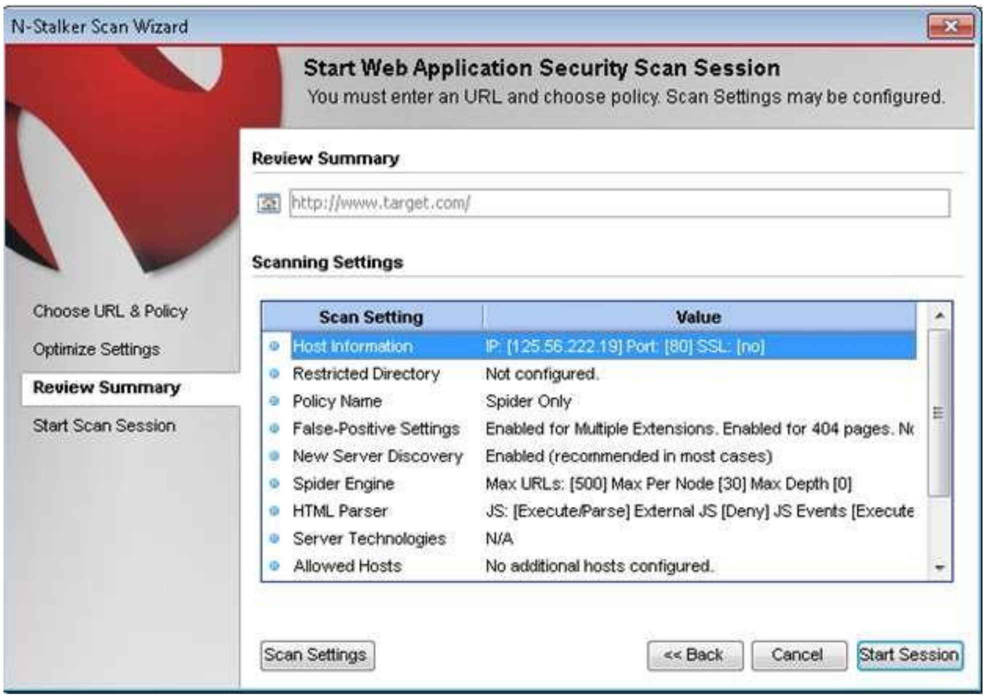
Start Web Application Security Scan Session
You must enter an URL and choose policy. Scan Settings may be configured.

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
Not available in N-Stalker Free Edition
(You may load spider data from previously saved scan sessions)
Use local cache from previously saved session (Avoid new web crawling)



N-Stalker Scan Wizard

Start Web Application Security Scan Session

You must enter an URL and choose policy. Scan Settings may be configured.

Choose URL & Policy
Optimize Settings
Review Summary
Start Scan Session

Review Summary

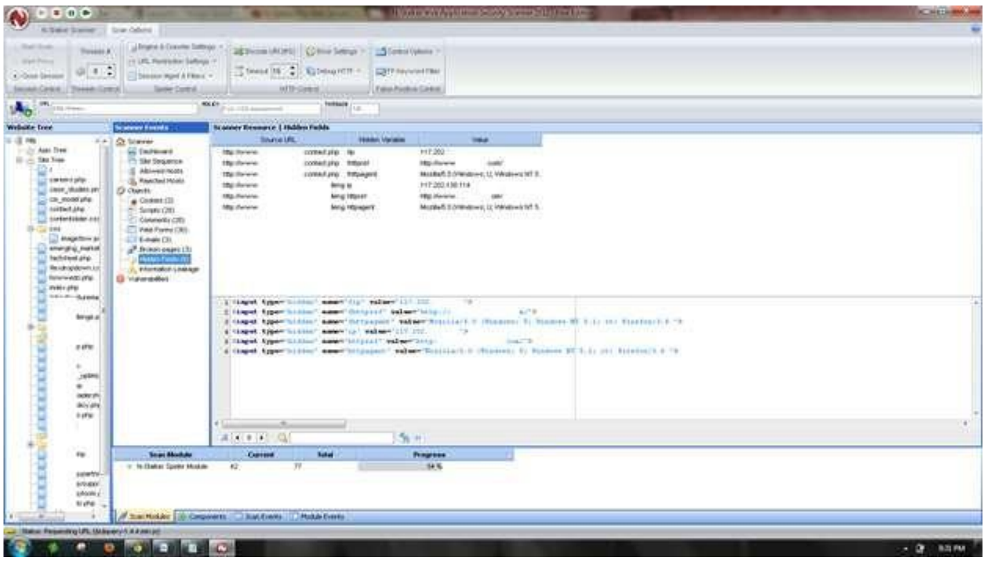
http://www.target.com/

Scanning Settings

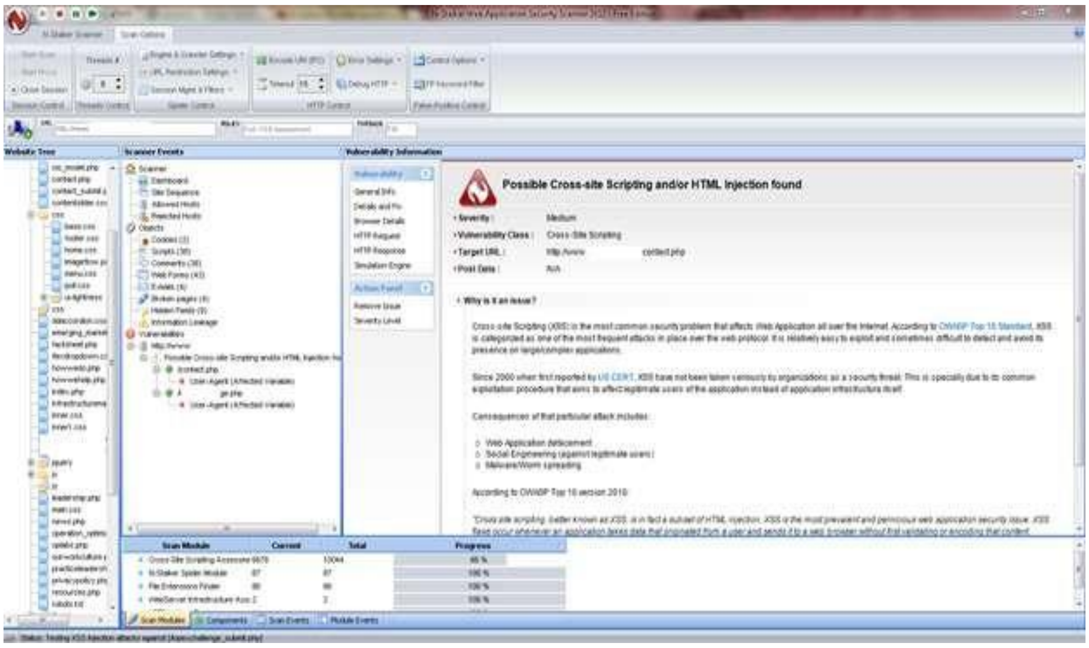
Scan Setting	Value
Host Information	IP: [125.56.222.19] Port: [80] SSL: [no]
Restricted Directory	Not configured.
Policy Name	Spider Only
False-Positive Settings	Enabled for Multiple Extensions. Enabled for 404 pages. N
New Server Discovery	Enabled (recommended in most cases)
Spider Engine	Max URLs: [500] Max Per Node [30] Max Depth [0]
HTML Parser	JS: [Execute/Parse] External JS [Deny] JS Events [Execute
Server Technologies	N/A
Allowed Hosts	No additional hosts configured.

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?



Possible Cross-site Scripting and/or HTML Injection found

Severity : Medium
Vulnerability Class : Cross-Site Scripting
Target URL : http://www contact.php
Post Data : N/A

Why is it an issue?

Cross-site Scripting (XSS) is the most common security problem that affects Web Application all over the Internet. According to OWASP Top 10 Standard, XSS is categorized as one of the most frequent attacks in place over the web protocol. It is relatively easy to exploit and sometimes difficult to detect and avoid its presence on large/complex applications.

Since 2000 when first reported by US CERT, XSS have not been taken seriously by organizations as a security threat. This is specially due to its common exploitation procedure that aims to affect legitimate users of the application instead of application infrastructure itself.

Consequences of that particular attack includes:

- Web Application defacement
- Social Engineering (against legitimate users)
- Malware/Worm spreading

According to OWASP Top 10 version 2010:

"Cross site scripting, better known as XSS, is in fact a subset of HTML injection. XSS is the most prevalent and pernicious web application security issue. XSS flaws occur whenever an application takes data that originated from a user and sends it to a web browser without first validating or encoding that content.

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:11a **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
```

RESULT:

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

Ex No:11b Date:	DEFEATING MALWARE - ROOTKIT HUNTER

AIM:

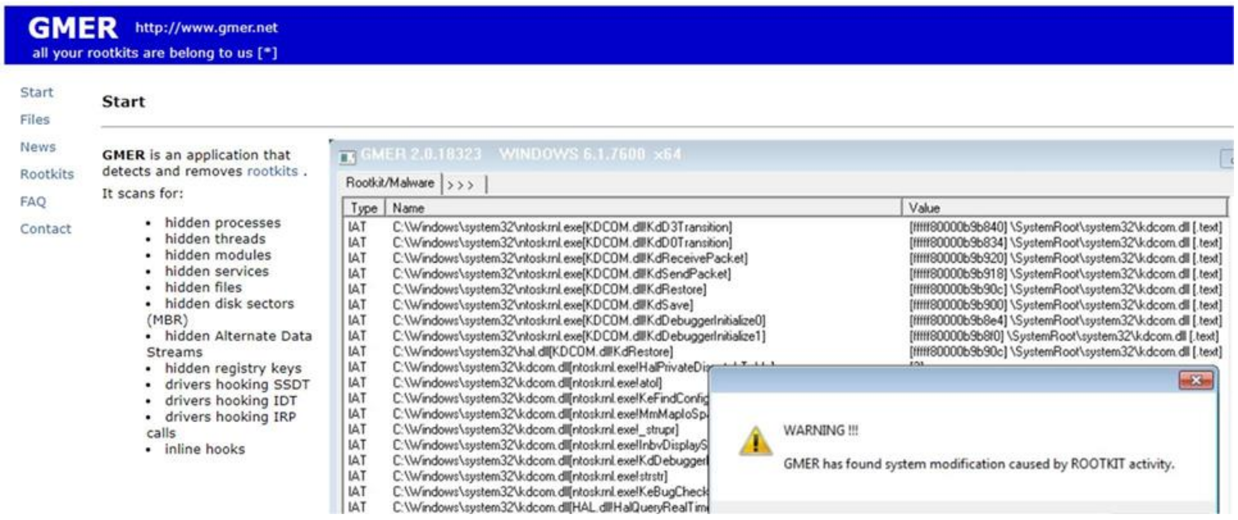
To install a rootkit hunter and find the malwares in a computer.

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.



Start

GMER is an application that detects and removes rootkits .
It scans for:

- hidden processes
- hidden threads
- hidden modules
- hidden services
- hidden files
- hidden disk sectors (MBR)
- hidden Alternate Data Streams
- hidden registry keys
- drivers hooking SSDT
- drivers hooking IDT
- drivers hooking IRP calls
- inline hooks

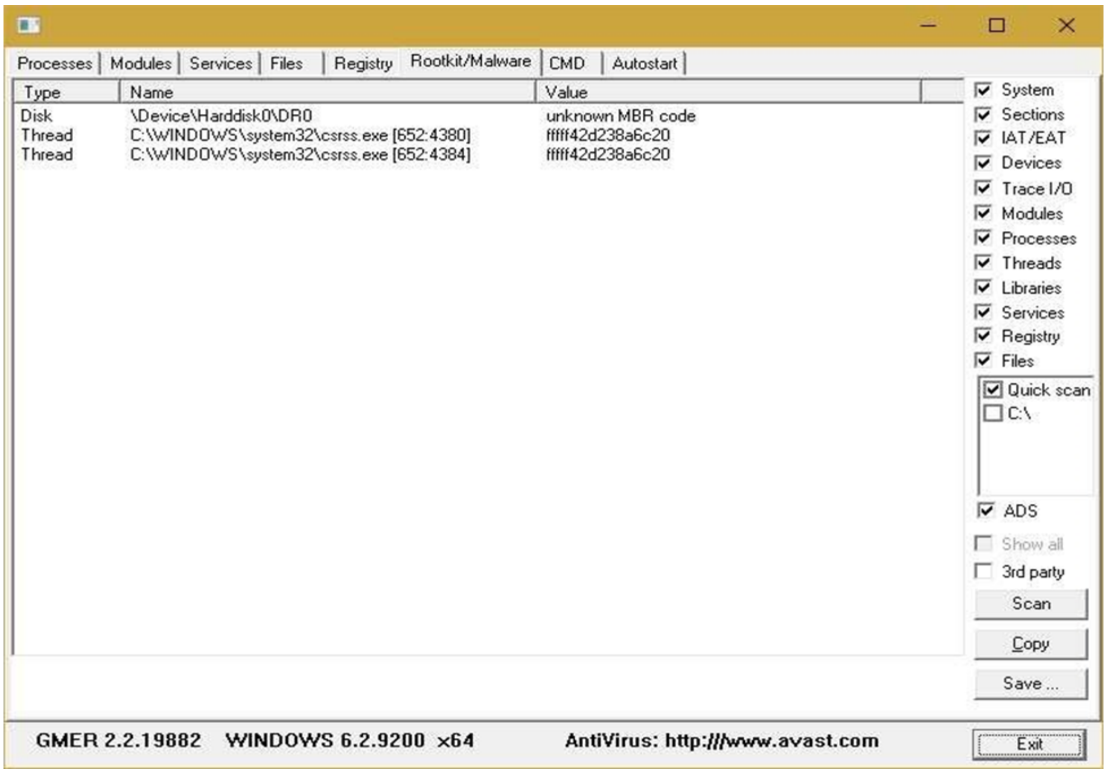
```
Type  Name                                                              Value
IAT   C:\Windows\system32\ntoskrnl.exe[KDCOM.dll!KdD3Transition]          [fffff80000b9b840] \SystemRoot\system32\kdcom.dll [.text]
IAT   C:\Windows\system32\ntoskrnl.exe[KDCOM.dll!KdD0Transition]          [fffff80000b9b834] \SystemRoot\system32\kdcom.dll [.text]
IAT   C:\Windows\system32\ntoskrnl.exe[KDCOM.dll!KdReceivePacket]         [fffff80000b9b920] \SystemRoot\system32\kdcom.dll [.text]
IAT   C:\Windows\system32\ntoskrnl.exe[KDCOM.dll!KdSendPacket]            [fffff80000b9b918] \SystemRoot\system32\kdcom.dll [.text]
IAT   C:\Windows\system32\ntoskrnl.exe[KDCOM.dll!KdRestore]               [fffff80000b9b90c] \SystemRoot\system32\kdcom.dll [.text]
IAT   C:\Windows\system32\ntoskrnl.exe[KDCOM.dll!KdSave]                  [fffff80000b9b900] \SystemRoot\system32\kdcom.dll [.text]
IAT   C:\Windows\system32\ntoskrnl.exe[KDCOM.dll!KdDebuggerInitialize0]   [fffff80000b9b8e4] \SystemRoot\system32\kdcom.dll [.text]
IAT   C:\Windows\system32\ntoskrnl.exe[KDCOM.dll!KdDebuggerInitialize1]   [fffff80000b9b8f0] \SystemRoot\system32\kdcom.dll [.text]
IAT   C:\Windows\system32\hal.dll[KDCOM.dll!KdRestore]                    [fffff80000b9b90c] \SystemRoot\system32\kdcom.dll [.text]
IAT   C:\Windows\system32\kdcom.dll[ntoskrnl.exe!HalPrivateDi
IAT   C:\Windows\system32\kdcom.dll[ntoskrnl.exe!atol]
IAT   C:\Windows\system32\kdcom.dll[ntoskrnl.exe!KeFindConfig
IAT   C:\Windows\system32\kdcom.dll[ntoskrnl.exe!MmMapIoSp
IAT   C:\Windows\system32\kdcom.dll[ntoskrnl.exe!_strupr]
IAT   C:\Windows\system32\kdcom.dll[ntoskrnl.exe!InbvDisplayS
IAT   C:\Windows\system32\kdcom.dll[ntoskrnl.exe!KdDebugger
IAT   C:\Windows\system32\kdcom.dll[ntoskrnl.exe!strstr]
IAT   C:\Windows\system32\kdcom.dll[ntoskrnl.exe!KeBugCheck
IAT   C:\Windows\system32\kdcom.dll[HAL.dll!HalQueryRealTim
```

```
WARNING !!!

GMER has found system modification caused by ROOTKIT activity.
```

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.


Processes | Modules | Services | Files | Registry | Rootkit/Malware | CMD | Autostart

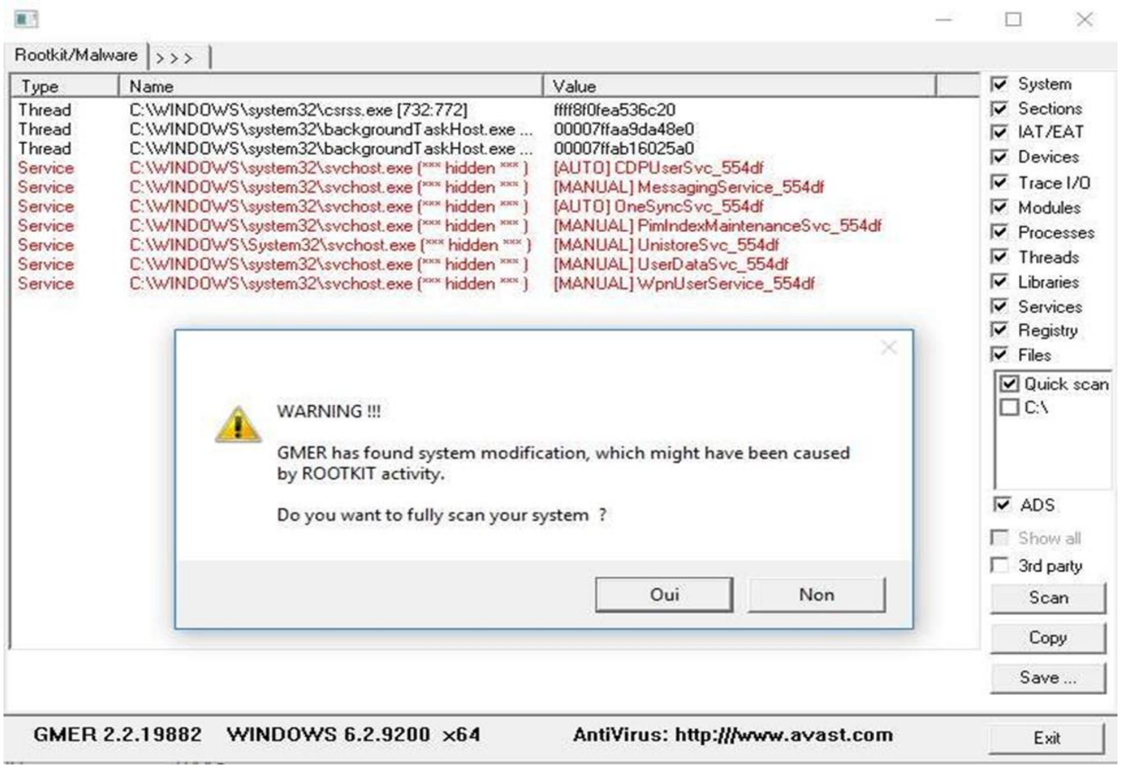
```
Type    Name                                          Value
Disk    \Device\Harddisk0\DR0                         unknown MBR code
Thread  C:\WINDOWS\system32\csrss.exe [652:4380]      fffff42d238a6c20
Thread  C:\WINDOWS\system32\csrss.exe [652:4384]      fffff42d238a6c20
```

[✓] System [✓] Sections [✓] IAT/EAT [✓] Devices [✓] Trace I/O [✓] Modules [✓] Processes [✓] Threads [✓] Libraries [✓] Services [✓] Registry [✓] Files [✓] Quick scan [ ] C:\ [✓] ADS [ ] Show all [ ] 3rd party

GMER 2.2.19882 WINDOWS 6.2.9200 x64 AntiVirus: http:///www.avast.com

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



```
Type     Name                                                   Value
Thread   C:\WINDOWS\system32\csrss.exe [732:772]                ffff8f0fea536c20
Thread   C:\WINDOWS\system32\backgroundTaskHost.exe ...         00007ffaa9da48e0
Thread   C:\WINDOWS\system32\backgroundTaskHost.exe ...         00007ffab16025a0
Service  C:\WINDOWS\system32\svchost.exe (*** hidden *** )      [AUTO] CDPUserSvc_554df
Service  C:\WINDOWS\system32\svchost.exe (*** hidden *** )      [MANUAL] MessagingService_554df
Service  C:\WINDOWS\system32\svchost.exe (*** hidden *** )      [AUTO] OneSyncSvc_554df
Service  C:\WINDOWS\system32\svchost.exe (*** hidden *** )      [MANUAL] PimIndexMaintenanceSvc_554df
Service  C:\WINDOWS\System32\svchost.exe (*** hidden *** )      [MANUAL] UnistoreSvc_554df
Service  C:\WINDOWS\system32\svchost.exe (*** hidden *** )      [MANUAL] UserDataSvc_554df
Service  C:\WINDOWS\system32\svchost.exe (*** hidden *** )      [MANUAL] WpnUserService_554df
```

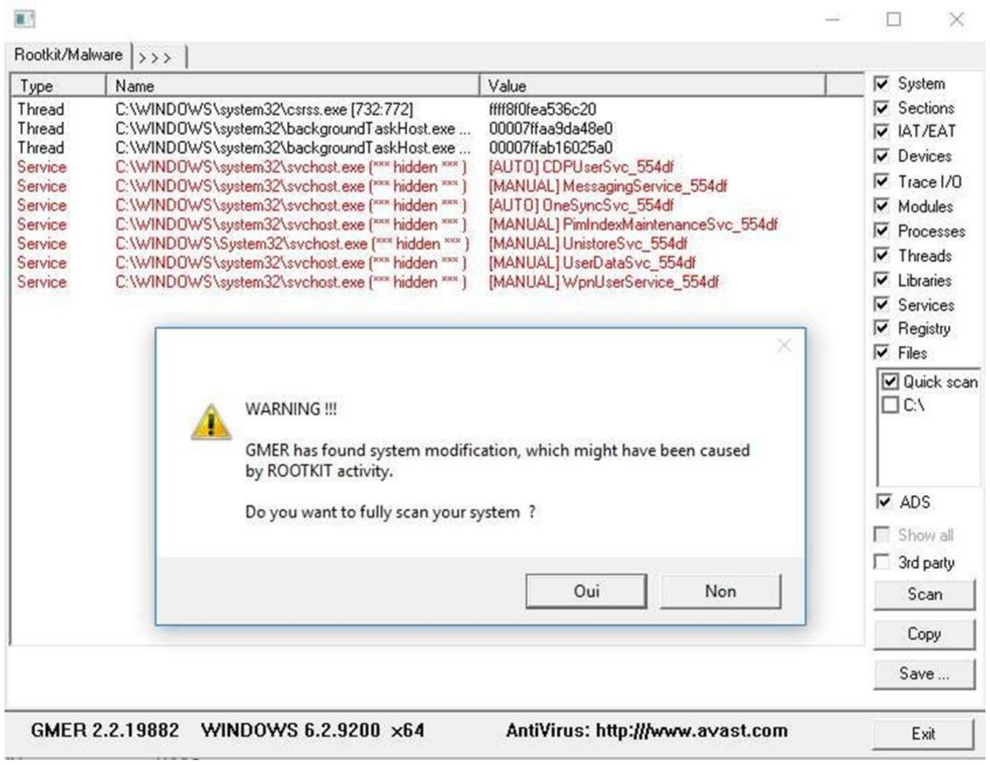
WARNING !!!

GMER has found system modification, which might have been caused by ROOTKIT activity.

Do you want to fully scan your system ?

[✓] System [✓] Sections [✓] IAT/EAT [✓] Devices [✓] Trace I/O [✓] Modules [✓] Processes [✓] Threads [✓] Libraries [✓] Services [✓] Registry [✓] Files [✓] Quick scan [ ] C:\ [✓] ADS [ ] Show all [ ] 3rd party

GMER 2.2.19882 WINDOWS 6.2.9200 x64 AntiVirus: http:///www.avast.com



Type	Name	Value
Thread	C:\WINDOWS\system32\csrss.exe [732:772]	ffff8f0fea536c20
Thread	C:\WINDOWS\system32\backgroundTaskHost.exe ...	00007ffaa9da48e0
Thread	C:\WINDOWS\system32\backgroundTaskHost.exe ...	00007ffab16025a0
Service	C:\WINDOWS\system32\svchost.exe (*** hidden ***)	[AUTO] CDPUserSvc_554df
Service	C:\WINDOWS\system32\svchost.exe (*** hidden ***)	[MANUAL] MessagingService_554df
Service	C:\WINDOWS\system32\svchost.exe (*** hidden ***)	[AUTO] OneSyncSvc_554df
Service	C:\WINDOWS\system32\svchost.exe (*** hidden ***)	[MANUAL] PimIndexMaintenanceSvc_554df
Service	C:\WINDOWS\System32\svchost.exe (*** hidden ***)	[MANUAL] UnistoreSvc_554df
Service	C:\WINDOWS\system32\svchost.exe (*** hidden ***)	[MANUAL] UserDataSvc_554df
Service	C:\WINDOWS\system32\svchost.exe (*** hidden ***)	[MANUAL] WpnUserService_554df

WARNING !!!

GMER has found system modification, which might have been caused by ROOTKIT activity.

Do you want to fully scan your system ?

GMER 2.2.19882 WINDOWS 6.2.9200 x64 AntiVirus: http://www.avast.com

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.