INDEX PAGE

S.NO	DATE	TITLE	MARKS	SIGNATURE
1 A		Ceaser Cipher		
1 B		Play Fair Cipher		
1 C		Hill Cipher		
1 D		VignereCipher		
2 A		Rail Fence Cipher		
2 B		Column Transformation Cipher		
3		Data Encryption Standard (DES) Algorithm		
4		AES Algorithm		
5		RSA Algorithm		
6		Diffie-Hellman Key Exchange		
7		SHA- 1 algorithm		
8		Digital Signature Standard		
9		Demonstrate IDS Using Any Snort Tool		
10		Exploring N Stalker , a Vulnerability , Assessment Tool		
11		Defeating Malware - Building Trojans		
12		Defeating Malware Rootkit Hunter		

EX.NO : 01(A)

CEASER CIPHER

DATE :

AIM:

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

ALGORITHMS:

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

$$En(x) = (x + n) \bmod 26$$

5. Decryption is performed similarly,

$$Dn(x)=(x - n) \bmod 26$$

PROGRAM:

CaearCipher.

```
#include <stdio.h>
#include <stdlib.h>
```

```
// Function to perform Caesar Cipher encryption
void caesarEncrypt(char *text, int key) {
    for (int i = 0; text[i] != '\0'; i++) {
        char c = text[i];
        // Check if the character is an uppercase letter
        if (c >= 'A' && c <= 'Z') {
            text[i] = ((c - 'A' + key) % 26 + 26) % 26 + 'A';
        }
        // Check if the character is a lowercase letter
        else if (c >= 'a' && c <= 'z') {
            text[i] = ((c - 'a' + key) % 26 + 26) % 26 + 'a';
        }
        // Ignore non-alphabetic characters
    }
}

// Function to perform Caesar Cipher decryption
void caesarDecrypt(char *text, int key) {
    // Decryption is the same as encryption with a negative key
    caesarEncrypt(text, -key);
}

int main() {
    char message[100]; // Declare a character array to store the message
    int key;

    printf("Enter the message to encrypt: ");
    fgets(message, sizeof(message), stdin); // Read input from the user
    printf("Enter the Caesar Cipher key (an integer): ");
    scanf("%d", &key); // Read the key from the user
    // Encrypt the message using the Caesar Cipher
    caesarEncrypt(message, key);
    printf("Encrypted Message: %s", message);
    // Decrypt the message back to the original
    caesarDecrypt(message, key);
    printf("Decrypted Message: %s", message);
    return 0;
}
```

OUTPUT:

```
 Simulating Caesar Cipher
 ---------------------------

 Input : Anna University
 Encrypted Message : Dqqd Xqlyhuvlwb
 Decrypted Message : Anna University
```

RESULT :

EX.NO :1(B) **PLAY FAIR CIPHER**
DATE :

AIM:

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

ALGORITHM DESCRIPTION:

The Playfair cipher uses a 5 by 5 table containing a key word or phrase. To generate the key table, first fill the spaces in the table with the letters of the keyword, then fill the remaining spaces with the rest of the letters of the alphabet in order (usually omitting "Q" to reduce the alphabet to fit; other versions put both "I" and "J" in the same space). The key can be written in the top rows of the table, from left to right, or in some other pattern, such as a spiral beginning in the upper-left-hand corner and ending in the centre.

The keyword together with the conventions for filling in the 5 by 5 table constitutes the cipher key. To encrypt a message, one would break the message into digrams (groups of 2 letters) such that, for example, "HelloWorld" becomes "HE LL OW OR LD", and map them out on the key table. Then apply the following 4 rules, to each pair of letters in the plaintext:

1. If both letters are the same (or only one letter is left), add an "X" after the first letter. Encrypt the new pair and continue. Some variants of Playfair use "Q" instead of "X", but any letter, itself uncommon as a repeated pair, will do.
2. If the letters appear on the same row of your 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).
3. If the letters appear on the same column of your 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).
4. If the letters are not on the same row or column, replace them 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 letter of the encrypted pair is the one that lies on the same row as the first letter of the plaintext pair.

To decrypt, use the INVERSE (opposite) of the last 3 rules, and the 1st as-is (dropping any extra "X"s, or "Q"s that do not make sense in the final message when finished).

PROGRAM :

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define SIZE 30

// Function to convert the string to lowercase
void toLowerCase(char plain[], int ps)
{
        int i;
        for (i = 0; i < ps; i++) {
                if (plain[i] > 64 && plain[i] < 91)
```

```
                        plain[i] += 32;
        }
}

// Function to remove all spaces in a string
int removeSpaces(char* plain, int ps)
{
        int i, count = 0;
        for (i = 0; i < ps; i++)
                if (plain[i] != ' ')
                        plain[count++] = plain[i];
        plain[count] = '\0';
        return count;
}

// Function to generate the 5x5 key square
void generateKeyTable(char key[], int ks, char keyT[5][5])
{
        int i, j, k, flag = 0, *dicty;

        // a 26 character hashmap
        // to store count of the alphabet
        dicty = (int*)calloc(26, sizeof(int));
        for (i = 0; i < ks; i++) {
                if (key[i] != 'j')
                        dicty[key[i] - 97] = 2;
        }

        dicty['j' - 97] = 1;

        i = 0;
        j = 0;

        for (k = 0; k < ks; k++) {
                if (dicty[key[k] - 97] == 2) {
                        dicty[key[k] - 97] -= 1;
                        keyT[i][j] = key[k];
                        j++;
                        if (j == 5) {
                                i++;
                                j = 0;
                        }
                }
        }

        for (k = 0; k < 26; k++) {
                if (dicty[k] == 0) {
                        keyT[i][j] = (char)(k + 97);
```

```
				j++;
				if (j == 5) {
					i++;
					j = 0;
				}
			}
		}
}

// Function to search for the characters of a digraph
// in the key square and return their position
void search(char keyT[5][5], char a, char b, int arr[])
{
		int i, j;

		if (a == 'j')
			a = 'i';
		else if (b == 'j')
			b = 'i';

		for (i = 0; i < 5; i++) {

			for (j = 0; j < 5; j++) {

				if (keyT[i][j] == a) {
					arr[0] = i;
					arr[1] = j;
				}
				else if (keyT[i][j] == b) {
					arr[2] = i;
					arr[3] = j;
				}
			}
		}
}

// Function to find the modulus with 5
int mod5(int a)
{
		return (a % 5);
}

// Function to make the plain text length to be even
int prepare(char str[], int ptrs)
{
		if (ptrs % 2 != 0) {
			str[ptrs++] = 'z';
			str[ptrs] = '\0';
```

```
        }
        return ptrs;
}

// Function for performing the encryption
void encrypt(char str[], char keyT[5][5], int ps)
{
        int i, a[4];

        for (i = 0; i < ps; i += 2) {

                search(keyT, str[i], str[i + 1], a);

                if (a[0] == a[2]) {
                        str[i] = keyT[a[0]][mod5(a[1] + 1)];
                        str[i + 1] = keyT[a[0]][mod5(a[3] + 1)];
                }
                else if (a[1] == a[3]) {
                        str[i] = keyT[mod5(a[0] + 1)][a[1]];
                        str[i + 1] = keyT[mod5(a[2] + 1)][a[1]];
                }
                else {
                        str[i] = keyT[a[0]][a[3]];
                        str[i + 1] = keyT[a[2]][a[1]];
                }
        }
}

// Function to encrypt using Playfair Cipher
void encryptByPlayfairCipher(char str[], char key[])
{
        char ps, ks, keyT[5][5];

        // Key
        ks = strlen(key);
        ks = removeSpaces(key, ks);
        toLowerCase(key, ks);

        // Plaintext
        ps = strlen(str);
        toLowerCase(str, ps);
        ps = removeSpaces(str, ps);

        ps = prepare(str, ps);

        generateKeyTable(key, ks, keyT);

        encrypt(str, keyT, ps);
```

```
}

// Driver code
int main()
{
        char str[SIZE], key[SIZE];

        // Key to be encrypted
        strcpy(key, "Monarchy");
        printf("Key text: %s\n", key);

        // Plaintext to be encrypted
        strcpy(str, "instruments");
        printf("Plain text: %s\n", str);

        // encrypt using Playfair Cipher
        encryptByPlayfairCipher(str, key);

        printf("Cipher text: %s\n", str);

        return 0;
}
```

Output:

```
Key text: Monarchy

Plain text: instruments

Cipher text: gatlmzclrqtx
```

<u>**RESULT:**</u>

EX.NO :1(C) **HILL CIPHER**
DATE :

AIM:

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

ALGORITHM DESCRIPTION:

The Hill cipher is a substitution cipher invented by Lester S. Hill in 1929. Each letter is represented by a number modulo 26. To encrypt a message, each block of n letters is multiplied by an invertible n × n matrix, again modulus 26.
To decrypt the message, each block is multiplied by the inverse of the matrix used for encryption.
The matrix used for encryption is the cipher key, and it should be chosen randomly from the set of invertible n × n matrices (modulo 26).
The cipher can, be adapted to an alphabet with any number of letters. All arithmetic just needs to be done modulo the number of letters instead of modulo 26.

PROGRAM:

```
#include <stdio.h>
#include <string.h>
int keymat[3][3] = { { 1, 2, 1 }, { 2, 3, 2 }, { 2, 2, 1 } };
int invkeymat[3][3] = { { -1, 0, 1 }, { 2, -1, 0 }, { -2, 2, -1 } };
char key[] = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
char encode(char a, char b, char c) {
char ret[4];
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];
ret[0] = key[x % 26];
ret[1] = key[y % 26];
ret[2] = key[z % 26];
ret[3] = '\0';
return ret;
}
char decode(char a, char b, char c) {
char ret[4];
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];ret[0] =
key[(x % 26 < 0) ? (26 + x % 26) : (x % 26)];
ret[1] = key[(y % 26 < 0) ? (26 + y % 26) : (y % 26)];
ret[2] = key[(z % 26 < 0) ? (26 + z % 26) : (z % 26)];
ret[3] = '\0';
return ret;
}
int main() {
char msg[1000];
char enc[1000] = "";
char dec[1000] = "";
int n;
strcpy(msg, "SecurityLaboratory");
printf("Simulation of Hill Cipher\n");
printf("Input message : %s\n", msg);
for (int i = 0; i < strlen(msg); i++) {
msg[i] = toupper(msg[i]);
}
// Remove spaces
n = strlen(msg) % 3;
// Append padding text X
if (n != 0) {
for (int i = 1; i <= (3 - n); i++) {
strcat(msg, "X");
}
}
printf("Padded message : %s\n", msg);
for (int i = 0; i < strlen(msg); i += 3) {
char a = msg[i];
char b = msg[i + 1];
char c = msg[i + 2];
strcat(enc, encode(a, b, c));
}
printf("Encoded message : %s\n", enc);
for (int i = 0; i < strlen(enc); i += 3) {
char a = enc[i];
char b = enc[i + 1];
char c = enc[i + 2];
strcat(dec, decode(a, b, c));
}
printf("Decoded message : %s\n", dec);
return 0;
}
```

OUTPUT:

```
 Simulating Hill Cipher
 -----------------------------------

 Input Message : SecurityLaboratory
 Padded Message : SECURITYLABORATORY
 Encrypted Message : EACSDKLCAEFQDUKSXU
 Decrypted Message : SECURITYLABORATORY
```

RESULT:

EX.NO : 1(D) **VIGENERE CIPHER**
DATE :

AIM:

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

ALGORITHM DESCRIPTION:

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. It is a simple form of polyalphabetic substitution.To encrypt, a table of alphabets can be used, termed a Vigenere square, or Vigenere table. 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. At different points in the encryption process, the cipher uses a different alphabet from one of the rows used. The alphabet at each point depends on a repeating keyword.

PROGRAM:

```
  #include<stdio.h>
  #include<string.h>
  //FunctiontoperformVigenereencryption
 voidvigenereEncrypt(char*text,constchar*key){
 inttextLen= strlen(text);
 intkeyLen=strlen(key);
 for(inti =0;i< textLen;i++){
 charc =text[i];
 if(c>='A'&&c<='Z'){
 //Encryptuppercaseletters
 text[i]=((c-'A'+key[i%keyLen]-'A')%26)+'A';
 }else if(c>='a'&&c<='z'){
 //Encryptlowercaseletters
 text[i]=((c-'a'+key[i%keyLen]-'A')%26)+'a';
 }
 }
 }
 //FunctiontoperformVigeneredecryption
 voidvigenereDecrypt(char*text,constchar*key){
 inttextLen= strlen(text);
 intkeyLen=strlen(key);

 for(inti =0;i< textLen;i++){
charc =text[i];
if(c>='A'&&c<='Z'){
 //Decryptuppercaseletters
 text[i]=((c-'A'-(key[i% keyLen]-'A') +26) %26)+ 'A';
 }else if(c>='a'&&c<='z'){
 //Decryptlowercaseletters
 text[i]=((c-'a'-(key[i% keyLen]-'A') +26) %26)+ 'a';
 }
 }
 }
 intmain(){
 constchar *key="KEY";//Replacewithyourdesired key
 char message[]= "Thisisasecretmessage.";//Replace withyourmessage
 //Encrypt themessage
 vigenereEncrypt(message,key);
 printf("EncryptedMessage:%s\n",message);
 //Decrypt themessage backtotheoriginal
 vigenereDecrypt(message,key);
 printf("DecryptedMessage:%s\n",message);
 Return 0;
```

OUTPUT :

```
Simulating Vigenere Cipher
---------------------------

Input Message : SecurityLaboratory
Encrypted Message : NMIYEMKCNIQVVROWXC
Decrypted Message : SECURITYLABORATORY
```

RESULT:

EX.NO : 2(A) **RAIL FENCE CIPHER**
DATE :

AIM:

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

ALGORITHM DESCRIPTION:

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. When we reach the top rail, the message is written downwards again until the whole plaintext is written out. The message is then read off in rows.

PROGRAM:

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
main()
{
 int i,j,len,rails,count,code[100][1000];
    char str[1000];
    printf("Enter a Secret Message\n");
    gets(str);
    len=strlen(str);
 printf("Enter number of rails\n");
 scanf("%d",&rails);
 for(i=0;i<rails;i++)
 {
  for(j=0;j<len;j++)
  {
   code[i][j]=0;
  }
 }
count=0;
j=0;
while(j<len)
{
if(count%2==0)
{
 for(i=0;i<rails;i++)
 {
  //strcpy(code[i][j],str[j]);
  code[i][j]=(int)str[j];
  j++;
 }

}
else
{
 for(i=rails-2;i>0;i--)
 {
  code[i][j]=(int)str[j];
 j++;
 }
}

count++;
}

for(i=0;i<rails;i++)
{
 for(j=0;j<len;j++)
 {
  if(code[i][j]!=0)
  printf("%c",code[i][j]);
 }

}
printf("\n");
}
```

<u>OUTPUT:</u>

```
Enter a Secret Message
wearediscovered
Enter number of rails
2
waeicvrderdsoee
```

<u>RESULT:</u>

EX NO-2 B COLUMN TRANSFORMATION CIPHER

Date:

AIM

To develop a program to implement the working of transposition cipher using column transformation cipher algorithm.

ALGORITHM

- Start the program.
- Encryption and Decryption is carried out.

➢ ENCRYPTION
- Get the input plain text from the user.
- Store the alternative digits in two separate arrays.
- Print the two arrays continuously to get the encrypted cipher text.

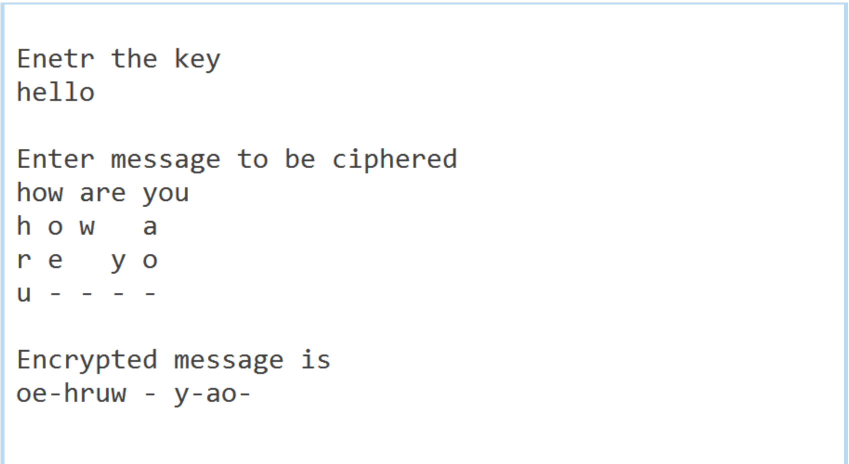
➢ DECRYPTION
- Get one character from the first array and print it.
- Get the same index value character from the array and print it.
- Continue printing the characters from the two arrays alternatively.
- When concatenated the plaintext can be obtained.

CODING

```
#include<stdio.h>
#include<string.h>

void cipher(int i,int c);
int findMin();
void makeArray(int,int);

char arr[22][22],darr[22][22],emessage[111],retmessage[111],key[55];
char temp[55],temp2[55];
int k=0;

int main()
{
  char *message,*dmessage;

  int i,j,klen,emlen,flag=0;
  int r,c,index,min,rows;
  clrscr();

  printf("Enetr the key\n");
  fflush(stdin);
  scanf(key);

  printf("\nEnter message to be ciphered\n");
  fflush(stdin);
  gets(message);
  strcpy(temp,key);
  klen=strlen(key);

  k=0;
  for(i=0; ;i++)
  {
    if(flag==1)
    break;

    for(j=0;key[j]!=NULL;j++)
    {
      if(message[k]==NULL)
       {
         flag=1;
         arr[i][j]='-';
       }
       else
       {
       arr[i][j]=message[k++];
       }
     }
  }
  r=i;
  c=j;

  for(i=0;i<r;i++)
  {
    for(j=0;j<c;j++)
    {
    printf("%c ",arr[i][j]);
    }
    printf("\n");
  }

k=0;

  for(i=0;i<klen;i++)
  {
   index=findMin();
   cipher(index,r);
  }

   emessage[k]='\0';
   printf("\nEncrypted message is\n");
   for(i=0;emessage[i]!=NULL;i++)
   printf("%c",emessage[i]);

   printf("\n\n");
   //deciphering

   emlen=strlen(emessage);
   //emlen is length of encrypted message

   strcpy(temp,key);
   rows=emlen/klen;
   //rows is no of row of the array to made from ciphered message
   rows;
   j=0;


   for(i=0,k=1;emessage[i]!=NULL;i++,k++)
    {
      //printf("\nEmlen=%d",emlen);
      temp2[j++]=emessage[i];
      if((k%rows)==0)
       {
       temp2[j]='\0';
       index=findMin();
       makeArray(index,rows);
       j=0;
       }
    }

    printf("\nArray Retrieved is\n");

     k=0;
     for(i=0;i<r;i++)
     {
    for(j=0;j<c;j++)
    {
    printf("%c ",darr[i][j]);
    //retrieving message
    retmessage[k++]=darr[i][j];

    }
    printf("\n");
     }
       retmessage[k]='\0';

     printf("\nMessage retrieved is\n");

     for(i=0;retmessage[i]!=NULL;i++)
     printf("%c",retmessage[i]);

  getch();
  return(0);
}

void cipher(int i,int r)
{
  int j;
  for(j=0;j<r;j++)
  {
    {
    emessage[k++]=arr[j][i];
    }
  }
 // emessage[k]='\0';
}
```

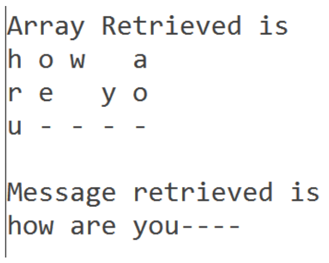
```
void makeArray(int col,int row)
{
 int i,j;

    for(i=0;i<row;i++)
       {
       darr[i][col]=temp2[i];
       }
}

int findMin()
{
  int i,j,min,index;

       min=temp[0];
       index=0;
       for(j=0;temp[j]!=NULL;j++)
       {
    if(temp[j]<min)
      {
      min=temp[j];
      index=j;
      }
       }

      temp[index]=123;
       return(index);
}
```

OUTPUT



```
Enetr the key
hello

Enter message to be ciphered
how are you
h o w   a
r e   y o
u - - - -

Encrypted message is
oe-hruw - y-ao-
```

Encrypted Message using Transposition Cipher



```
Array Retrieved is
h o w   a
r e   y o
u - - - -

Message retrieved is
how are you----
```

Decrypted Message using Transposition Cipher

<u>RESULT:</u>

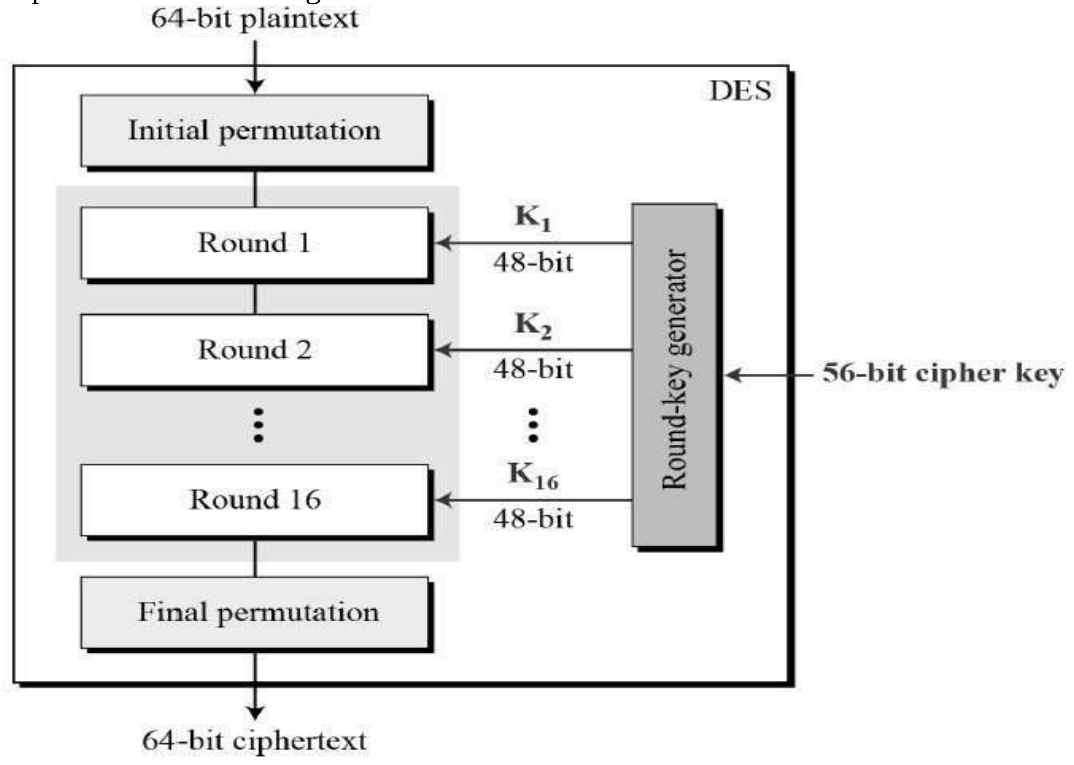
EX.NO : 3 **DATA ENCRYPTION STANDARD (DES)**
DATE :

AIM:

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

ALGORITHM DESCRIPTION:

The Data Encryption Standard (DES) is a symmetric-key block cipher published by the National Institute of Standards and Technology (NIST). DES is an implementation of a Feistel Cipher. It uses 16 round Feistel structure. The block size is 64-bit. Though, key length is 64-bit, DES has an effective key length of 56 bits, since 8 of the 64 bits of the key are not used by the encryption algorithm (function as check bits only). General Structure of DES is depicted in the following illustration –

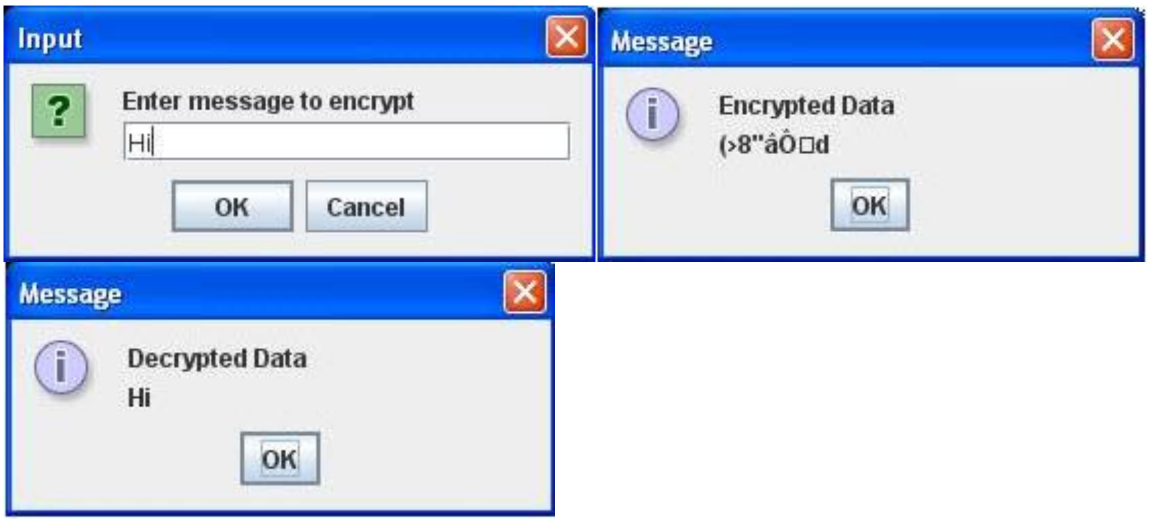


PROGRAM:

```
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();
                        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 Symmetric key = "+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 {
SecretKeySpec skeySpec = 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 {
SecretKeySpec skeySpec = 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();
            }
}
```

OUTPUT



```
Enter message to encrypt
Hi

Encrypted Data
(›8"âÔ□d

Decrypted Data
Hi
```

RESULT:

Ex. No : 4 ADVANCED ENCRYPTION STANDARD (DES) ALGORITHM
DATE : (URL ENCRYPTION)

AIM:

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

ALGORITHM:

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

PROGRAM:

AES.java

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

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

public class AES {

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

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

   public static String encrypt(String strToEncrypt, String secret) {
      try {
         setKey(secret);
         Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
         cipher.init(Cipher.ENCRYPT_MODE, secretKey);
         return
Base64.getEncoder().encodeToString(cipher.doFinal(strToEncrypt.getBytes("UTF-8")));
```

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

        String originalString = "www.annauniv.edu";
        String encryptedString = AES.encrypt(originalString, secretKey);
        String decryptedString = AES.decrypt(encryptedString, secretKey);

        System.out.println("URL Encryption Using AES Algorithm\n------------");
        System.out.println("Original URL : " + originalString);
        System.out.println("Encrypted URL : " + encryptedString);
        System.out.println("Decrypted URL : " + decryptedString);
    }
}
```

OUTPUT:

```
URL Encryption Using AES Algorithm
------------------------------------------------
Original URL : www.annauniv.edu
Encrypted URL : vibpFJW6Cvs5Y+L7t4N6YWWe07+JzS1d3CU2h3mEvEg=
Decrypted URL : www.annauniv.edu
```

RESULT:

EX.NO : 5 **RSA ALGORITHM**
DATE :

AIM:

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

ALGORITHM:

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

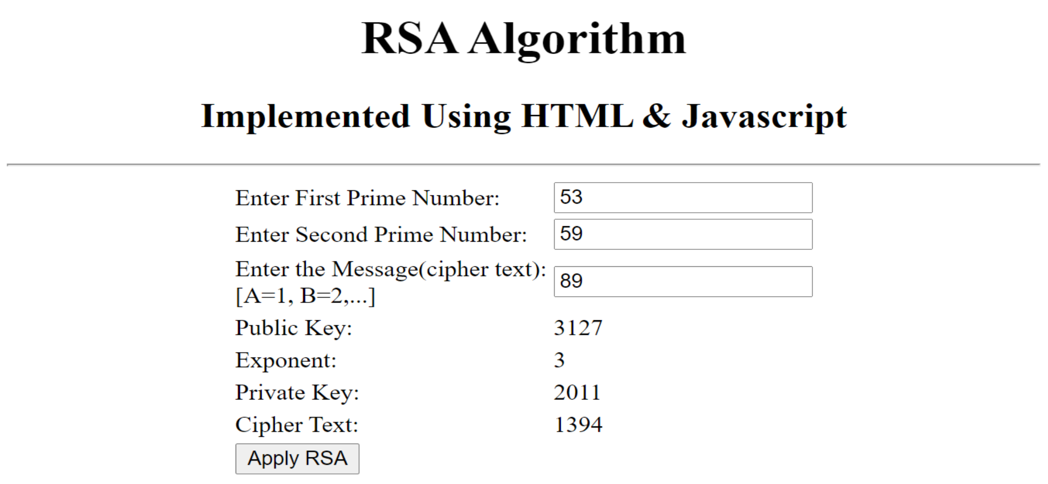
PROGRAM:

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:



```
                    RSA Algorithm

          Implemented Using HTML & Javascript

Enter First Prime Number:           53
Enter Second Prime Number:          59
Enter the Message(cipher text):     89
[A=1, B=2,...]
Public Key:                         3127
Exponent:                           3
Private Key:                        2011
Cipher Text:                        1394
```

RESULT:

EX.NO : 6 DIFFIEE HELLMAN KEY EXCHANGE ALGORITHM
DATE :
Aim :

Develop a program to implement Diffie Hellman Key Exchange Algorithm for encryption and Decryption.

Algorithm Description:

Diffie–Hellman key exchange (D–H) is a specific method of securely exchanging cryptographic keys over a public channel and was one of the first public-key protocols. The Diffie–Hellman key exchange method allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure channel. This key can then be used to encrypt subsequent communications using a symmetric key cipher.

Algorithm:

1. Global Public Elements:
   Let q be a prime number and α where $\alpha < q$ and α is a primitive root of q.
2. User A Key Generation:
   Select private X_A where $X_A < q$
   Calculate public Y_A where $Y_A = \alpha^{XA} \bmod q$
3. User B Key Generation:
   Select private X_B where $X_B < q$
   Calculate public Y_B where $Y_B = \alpha^{XB} \bmod q$
4. Calculation of Secret Key by User A
   $K = (Y_B)^{XA} \bmod q$
5. Calculation of Secret Key by User B:

PROGRAM:

DiffieHellman.java

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

```
            System.out.println("Error: Shared Secrets does not Match");
    }
}
```

OUTPUT:

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

RESULT:

EX.NO : 7 **SECURE HASH FUNCTION (SHA)**
DATE :

AIM:

Develop a program to implement Secure Hash Algorithm (SHA-1)

SECURED HASH ALGORITHM-1 (SHA-1):

Step 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 fewer than an even multiple of 512.

Step 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.

Step 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)

Step 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)

Step 5: Initialize Buffers….

SHA1 requires 160 bits or 5 buffers of words (32 bits):

H0 = 0x67452301
H1 = 0xEFCDAB89
H2 = 0x98BADCFE
H3 = 0x10325476
H4 = 0xC3D2E1F0

Step 6: Processing Message in 512-bit blocks (L blocks in total message)….

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

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

Step 6: Pseudo Code….

```
For loop on k = 1 to L
    (W(0),W(1),...,W(15)) = M[k] /* Divide M[k] into 16 words */
    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
        End of for loop
Output:
H0, H1, H2, H3, H4, H5: Word buffers with final message digest
```

PROGRAM

```java
import java.security.*;

public class SHA1 {

public static void main(String[] a) {

try {

MessageDigest md = MessageDigest.getInstance("SHA1");

System.out.println("Message digest object info: ");

System.out.println(" Algorithm = " +md.getAlgorithm());

System.out.println(" Provider = " +md.getProvider());

System.out.println(" ToString = " +md.toString());

 String input = "";

md.update(input.getBytes());

byte[] output = md.digest();

System.out.println();

System.out.println("SHA1(\""+input+"\") = " +bytesToHex(output));

input = "abc";

md.update(input.getBytes());

output = md.digest();

System.out.println();

System.out.println("SHA1(\""+input+"\") = " +bytesToHex(output));

input = "abcdefghijklmnopqrstuvwxyz";

md.update(input.getBytes());

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

OUTPUT:

```
C:\Program Files\Java\jdk1.6.0_20\bin>javac SHA1.java
C:\Program Files\Java\jdk1.6.0_20\bin>java SHA1
Message digest object info:
 Algorithm = SHA1
 Provider = SUN version 1.6
 ToString = SHA1 Message Digest from SUN, <initialized>
SHA1("") = DA39A3EE5E6B4B0D3255BFEF95601890AFD80709
SHA1("abc") = A9993E364706816ABA3E25717850C26C9CD0D89D
SHA1("abcdefghijklmnopqrstuvwxyz") =
32D10C7B8CF96570CA04CE37F2A19D84240D3A89
```

RESULT:

EX.NO : 8 **DIGITAL SIGNATURE STANDARD**
DATE :
AIM:

To write a C program to implement the signature scheme named digital signature standard (Euclidean Algorithm).

ALGORITHM:

STEP-1: Alice and Bob are investigating a forgery case of x and y.
STEP-2: X had document signed by him but he says he did not sign that document digitally.
STEP-3: Alice reads the two prime numbers p and a.
STEP-4: He chooses a random co-primes alpha and beta and the x's original signature x.
STEP-5: With these values, he applies it to the elliptic curve cryptographic equation to obtain y
STEP-6: Comparing this 'y' with actual y's document, Alice concludes that y is a forgery.

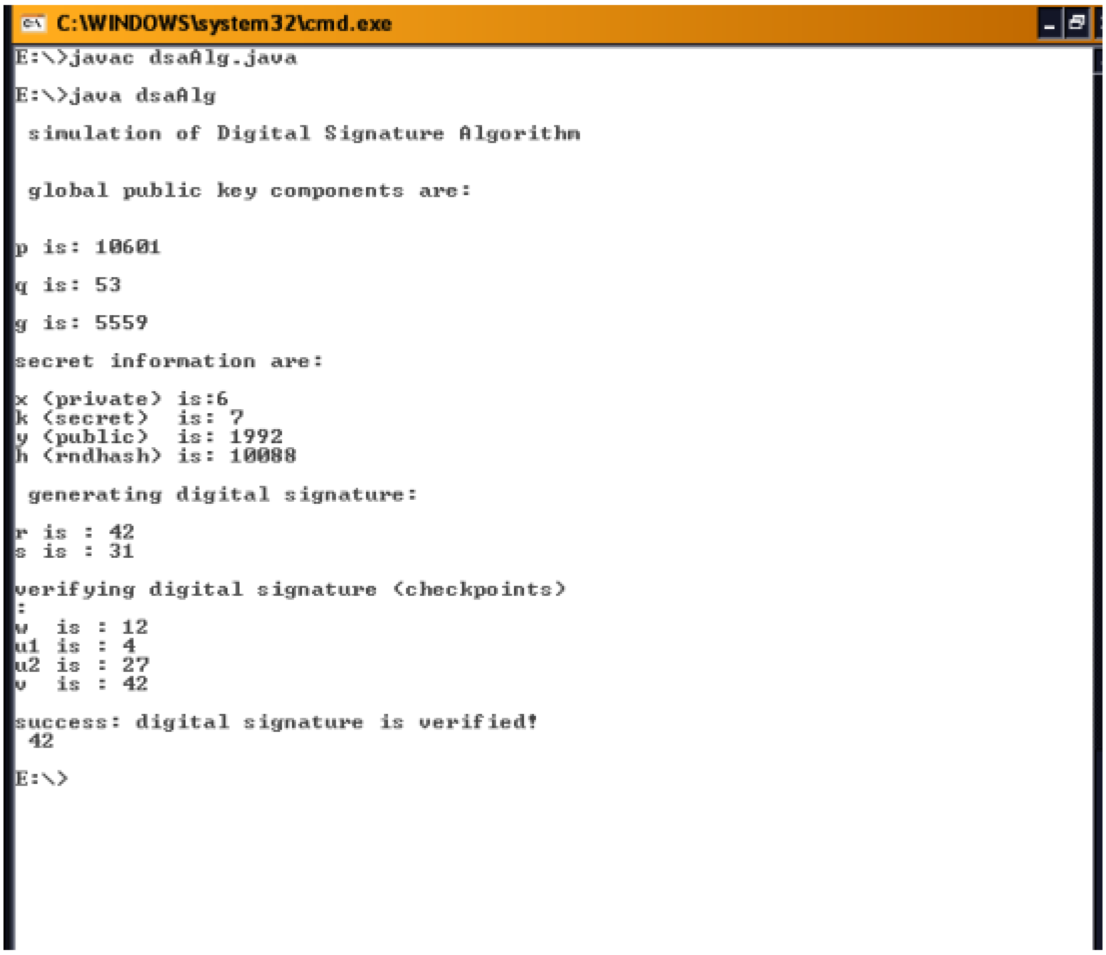
PROGRAM: (Digital Signature Standard)

```
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:



```
E:\>javac dsaAlg.java

E:\>java dsaAlg

 simulation of Digital Signature Algorithm


 global public key components are:


p is: 10601

q is: 53

g is: 5559

secret information are:

x (private) is:6
k (secret)  is: 7
y (public)  is: 1992
h (rndhash) is: 10088

 generating digital signature:

r is : 42
s is : 31

verifying digital signature (checkpoints)
:
w  is : 12
u1 is : 4
u2 is : 27
v  is : 42

success: digital signature is verified!
 42

E:\>
```

<u>RESULT:</u>

Ex. No : 9 DEMONSTRATION OF 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:

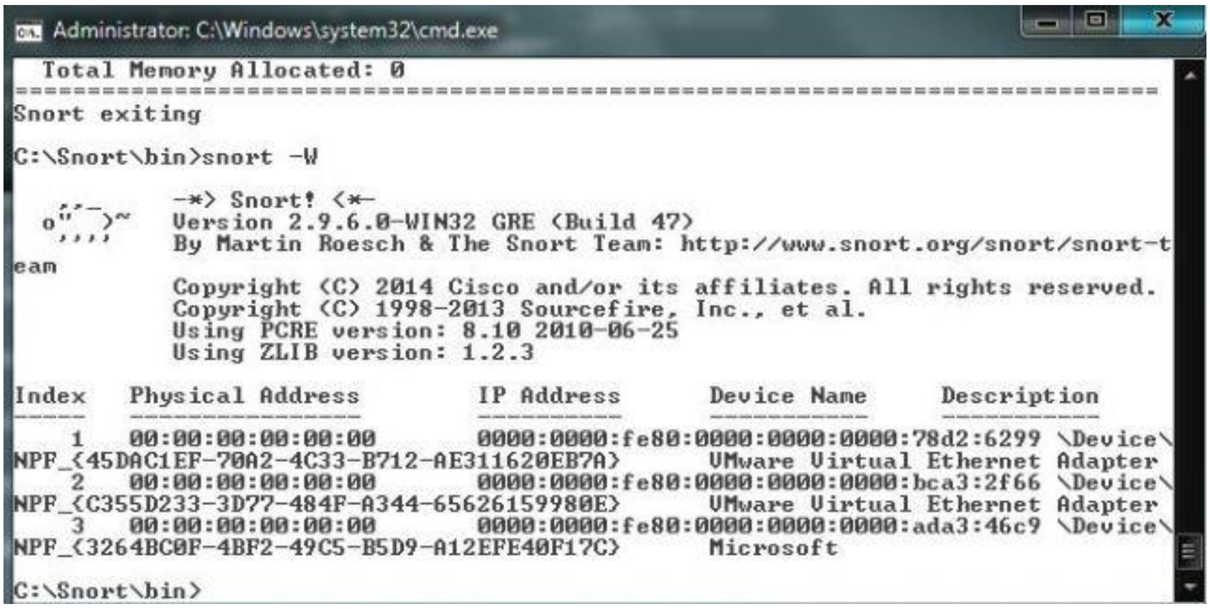
snort -dev -i 3

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


Administrator: C:\Windows\system32\cmd.exe

```
 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:

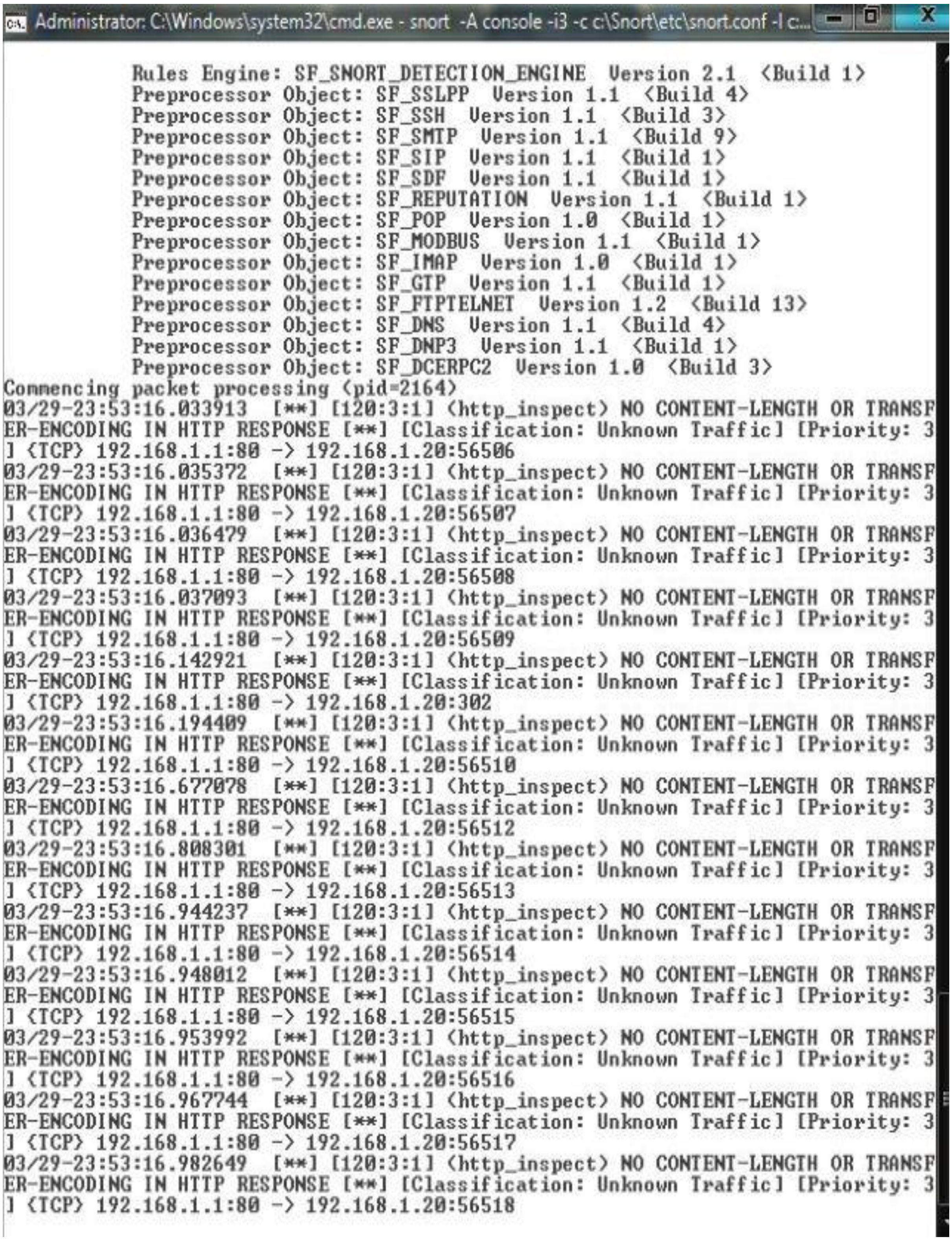
```
snort -c c:\snort\etc\snort.conf -l c:\snort\log -i 3
```
(Note: 3 is used for my interface card)

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:

EX. NO : 10 **EXPLORING N-STALKER, A VULNERABILITY ASSESSMENT**
DATE : **TOOL**

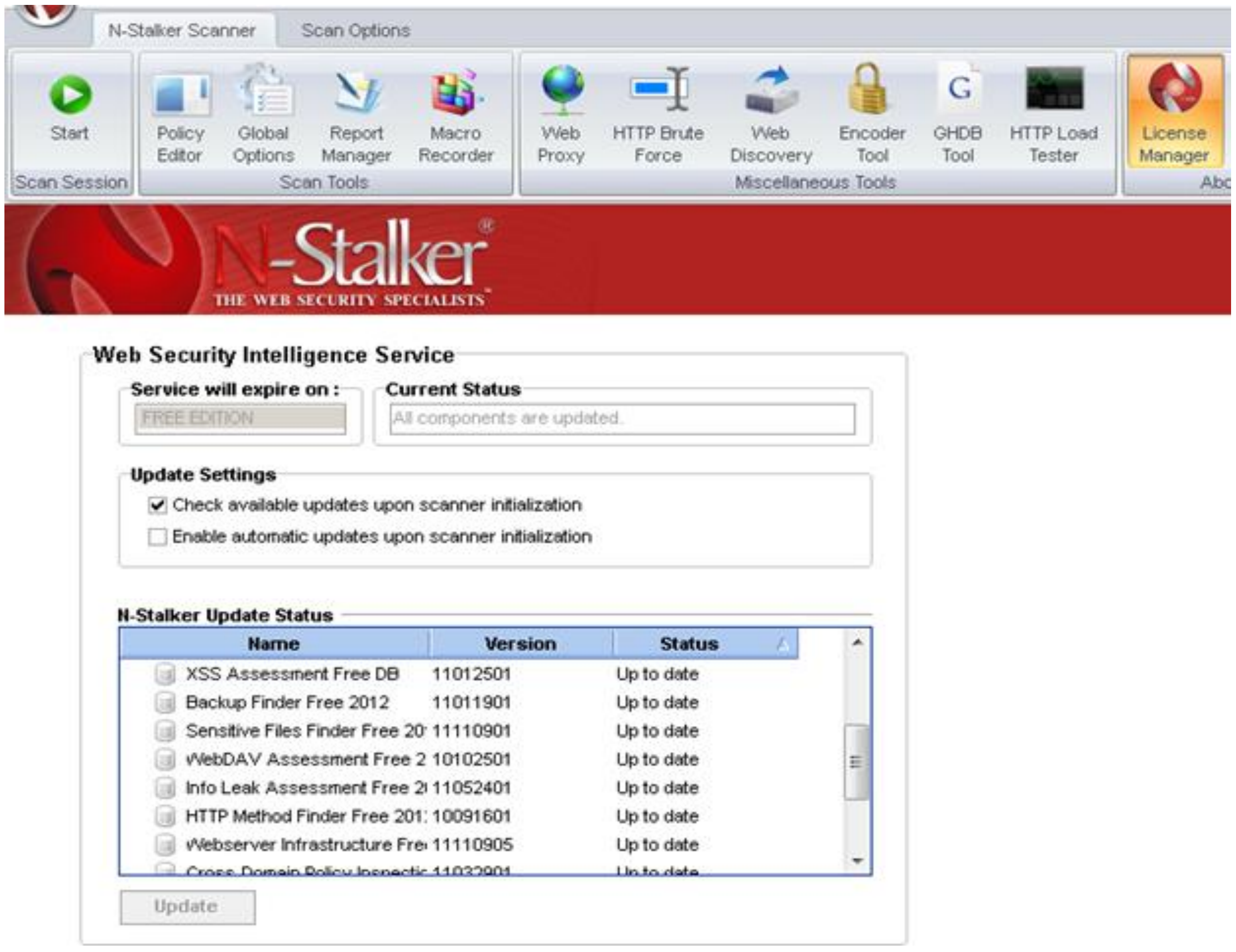
AIM:

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

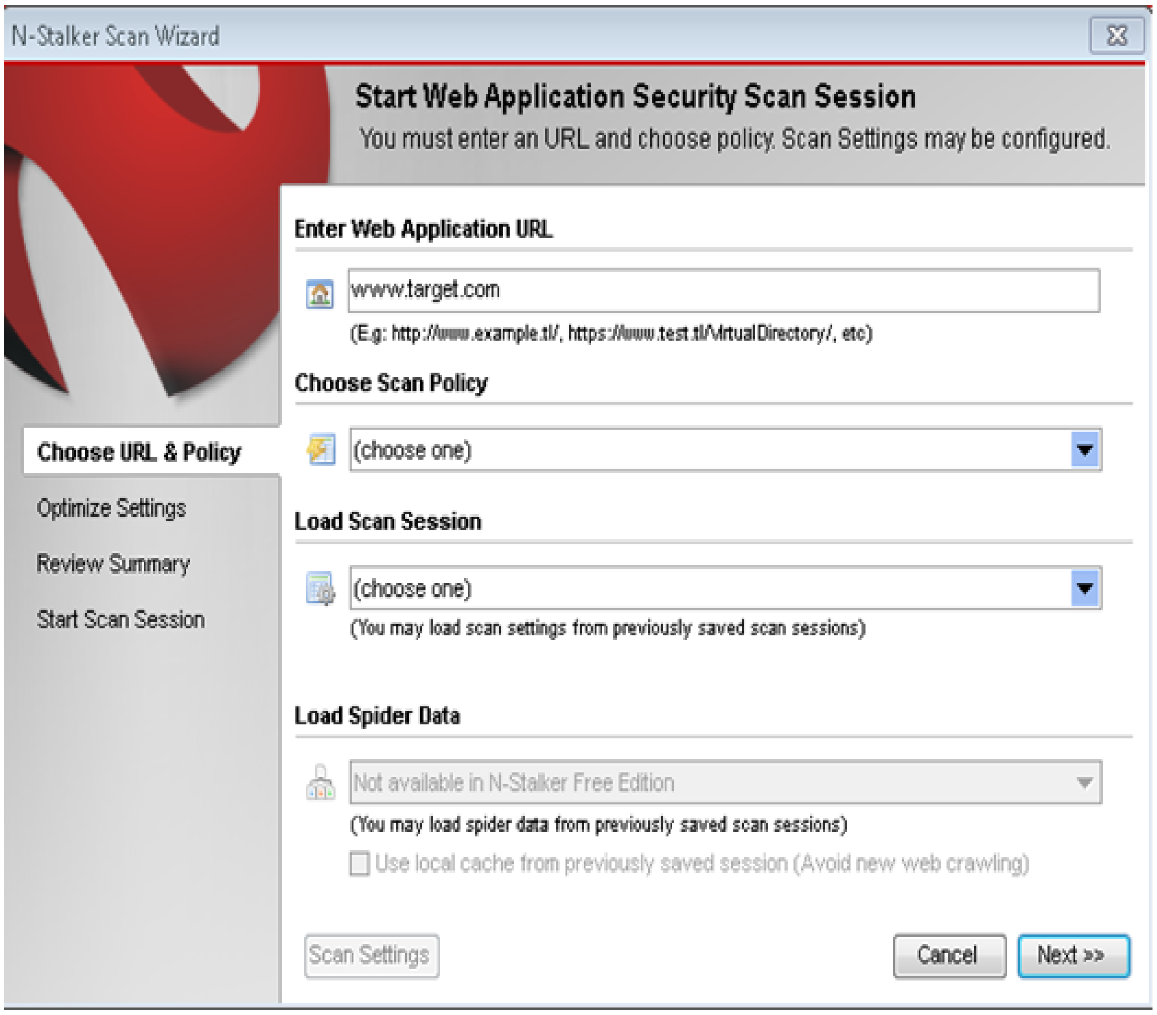
Now goto “Scan Session”, enter the target URL.

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)

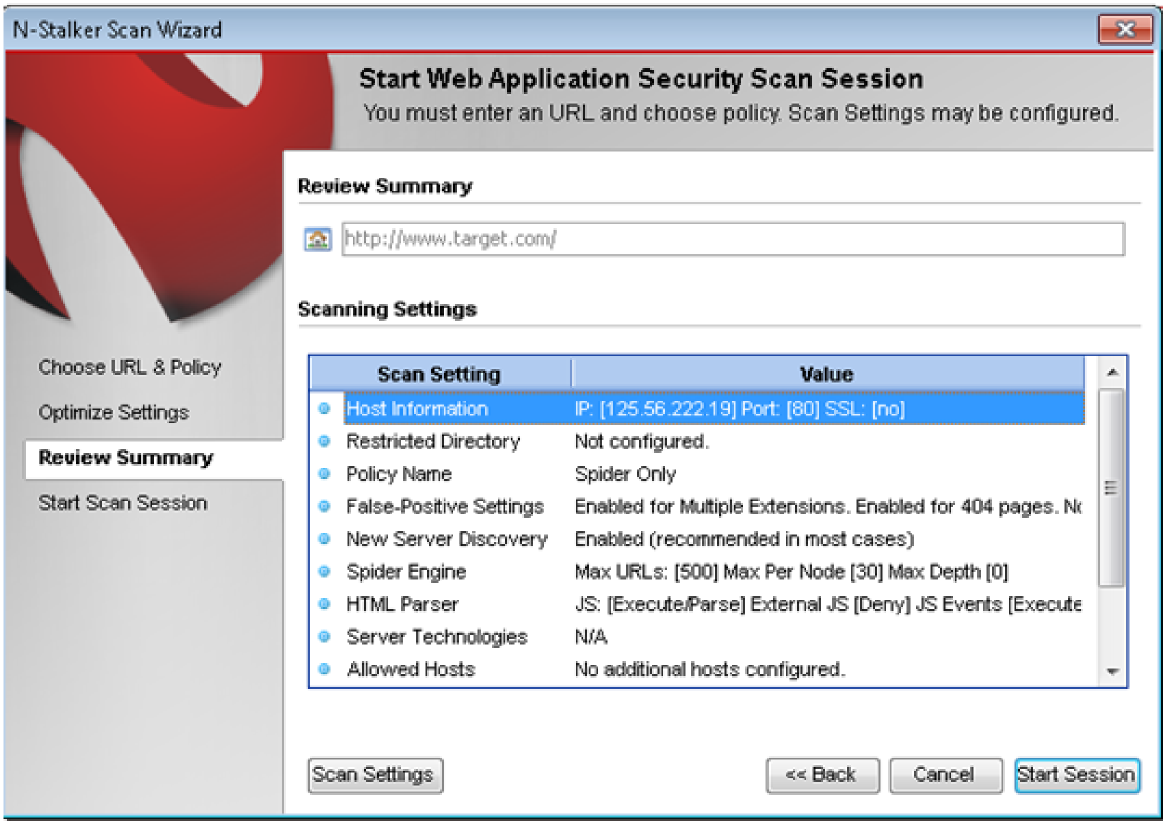
(You may load scan settings from previously saved scan sessions)

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/

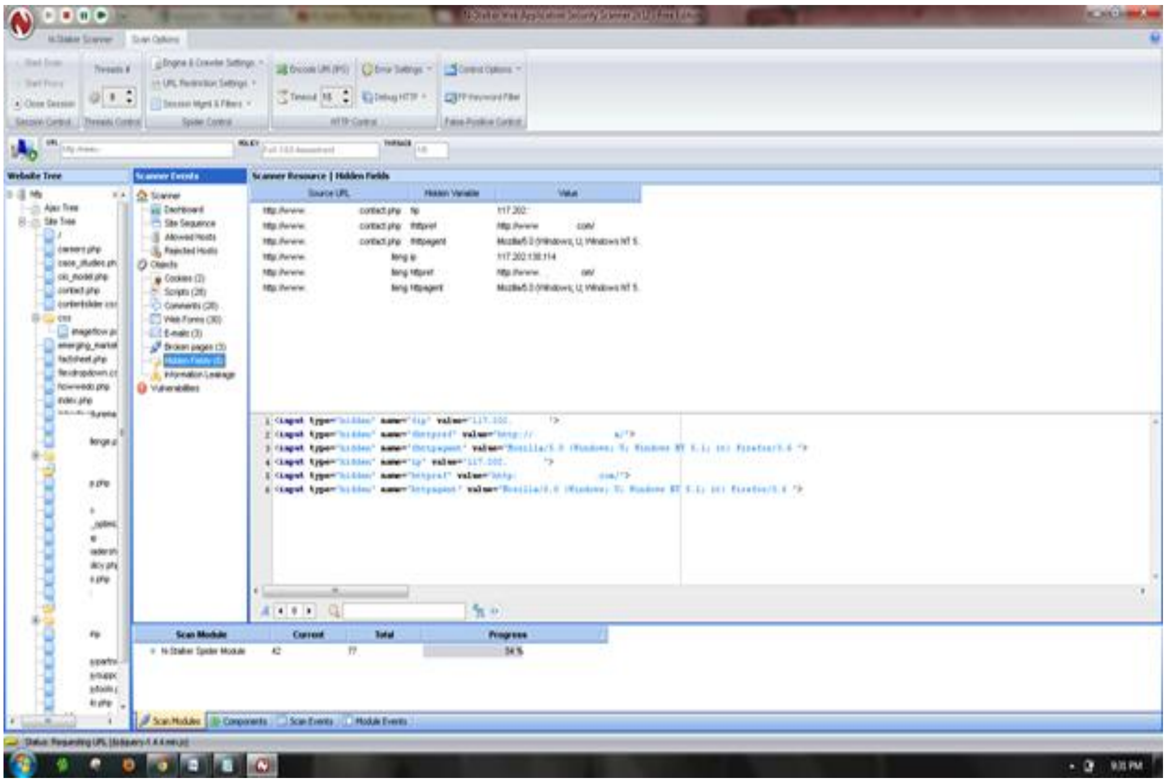
Scanning Settings

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.

Scan Settings · << Back · Cancel · Start Session

Once done, start the session and start the scan.

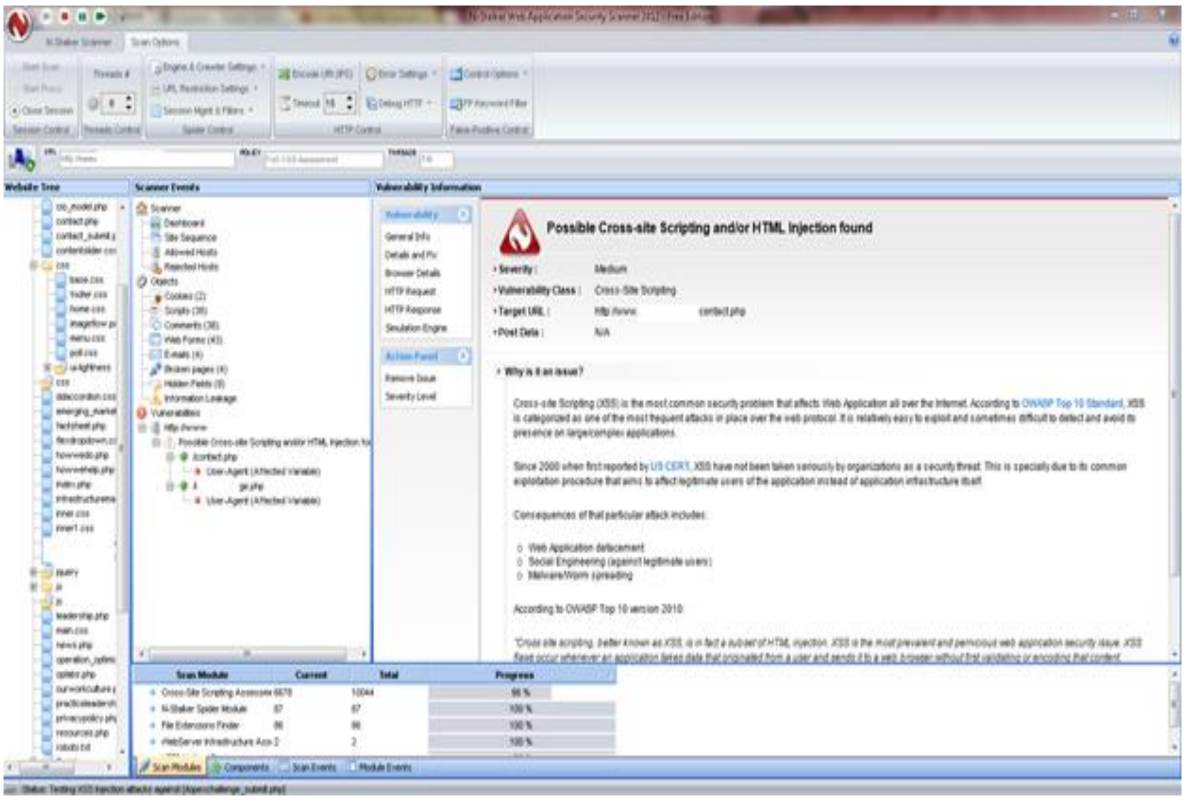
The scanner will crawl the whole website and will show the scripts, broken pages, hidden fields, information leakage, web forms related information which helps to analyze further.



Scanner Resource | Hidden Fields

Source URL	Hidden Variable	Value
http://www. contact.php	fip	117.202.
http://www. contact.php	fhttpref	http://www. .com/
http://www. contact.php	fhttpagent	Mozilla/5.0 (Windows; U; Windows NT 5.
http://www. fping	ip	117.202.138.114
http://www. fping	httpref	http://www. .com/
http://www. fping	httpagent	Mozilla/5.0 (Windows; U; Windows NT 5.

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.

Scan Module	Current	Total	Progress
Cross-Site Scripting Assessme	6678	10044	66 %
N-Stalker Spider Module	67	67	100 %
File Extensions Finder	86	86	100 %
WebServer Infrastructure Ass	2	2	100 %

Status: Testing XSS Injection attacks against [/open-challenge_submit.php]

RESULT:

EX. NO : 11
DATE :

DEFEATING MALWARE - BUILDING TROJANS

AIM:

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

PROCEDURE:

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

TROJAN:

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

- Trojans are generally spread by some form of social engineering, for example where a user is duped into executing an email attachment disguised to appear not suspicious, (e.g., a routine form to be filled in), or by clicking on some fake advertisement on social media or anywhere else.
- Although their payload can be anything, many modern forms act as a backdoor, contacting a controller which can then have unauthorized access to the affected computer.
- Trojans may allow an attacker to access users' personal information such as banking information, passwords, or personal identity.

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

CODE:

Trojan.bat

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

OUTPUT

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

RESULT:

EX. NO : 12
DATE :

INSTALL ROOTKITS AND STUDY VARIETY OF OPTIONS

AIM:

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

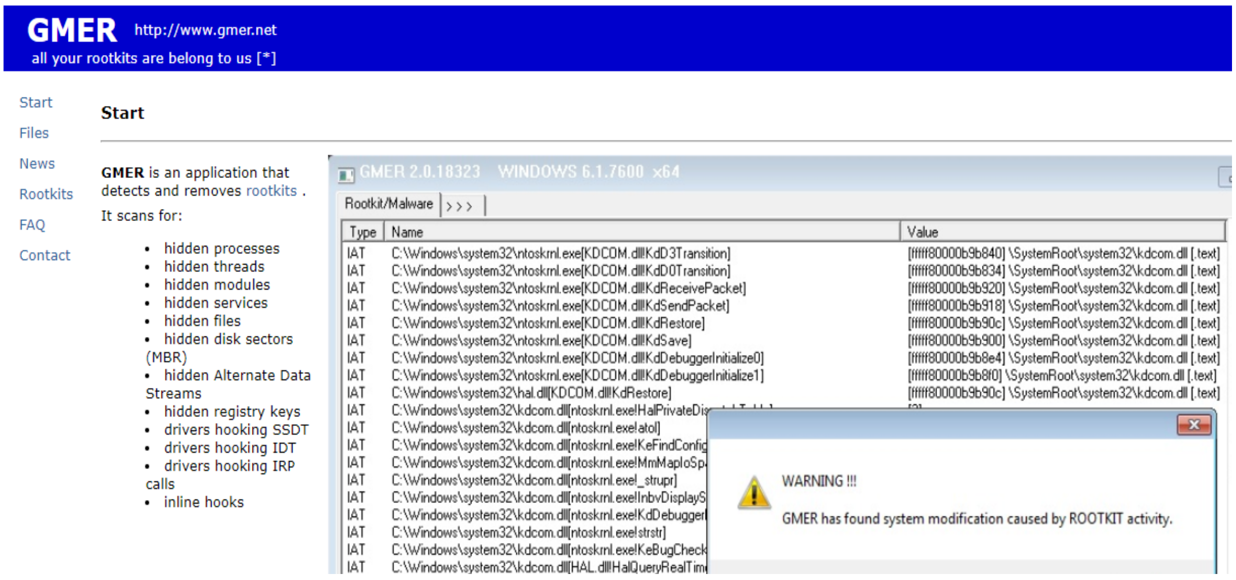
ROOTKIT HUNTER:

- 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


GMER
http://www.gmer.net
all your rootkits are belong to us [*]
Start
Files
News
Rootkits
FAQ
Contact
GMER 2.0.18323 WINDOWS 6.1.7600 x64
Rootkit/Malware
>>>

Start

GMER is an application that detects and removes rootkits .

It scans for:

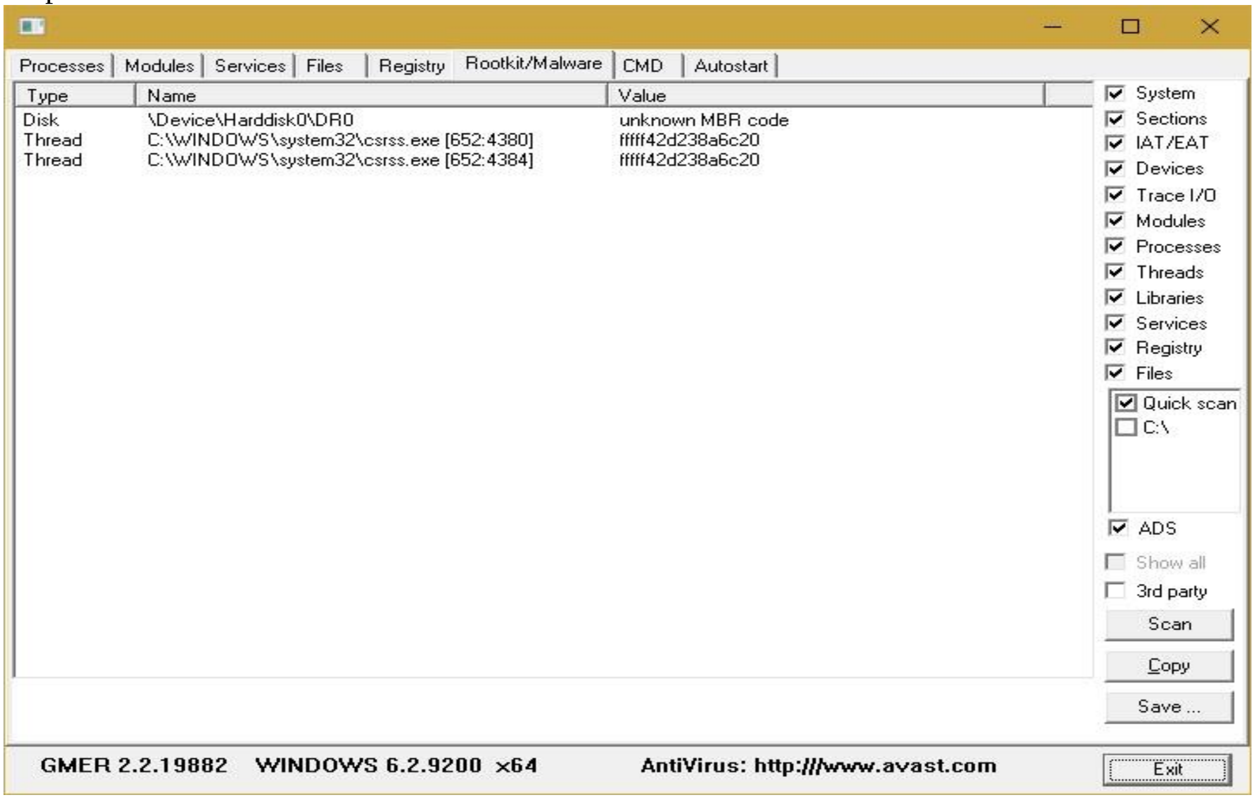
- hidden processes
- hidden threads
- hidden modules
- hidden services
- hidden files
- hidden disk sectors (MBR)
- hidden Alternate Data Streams
- hidden registry keys
- drivers hooking SSDT
- drivers hooking IDT
- drivers hooking IRP calls
- inline hooks

```
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!HalPrivateDis
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.

Step 2



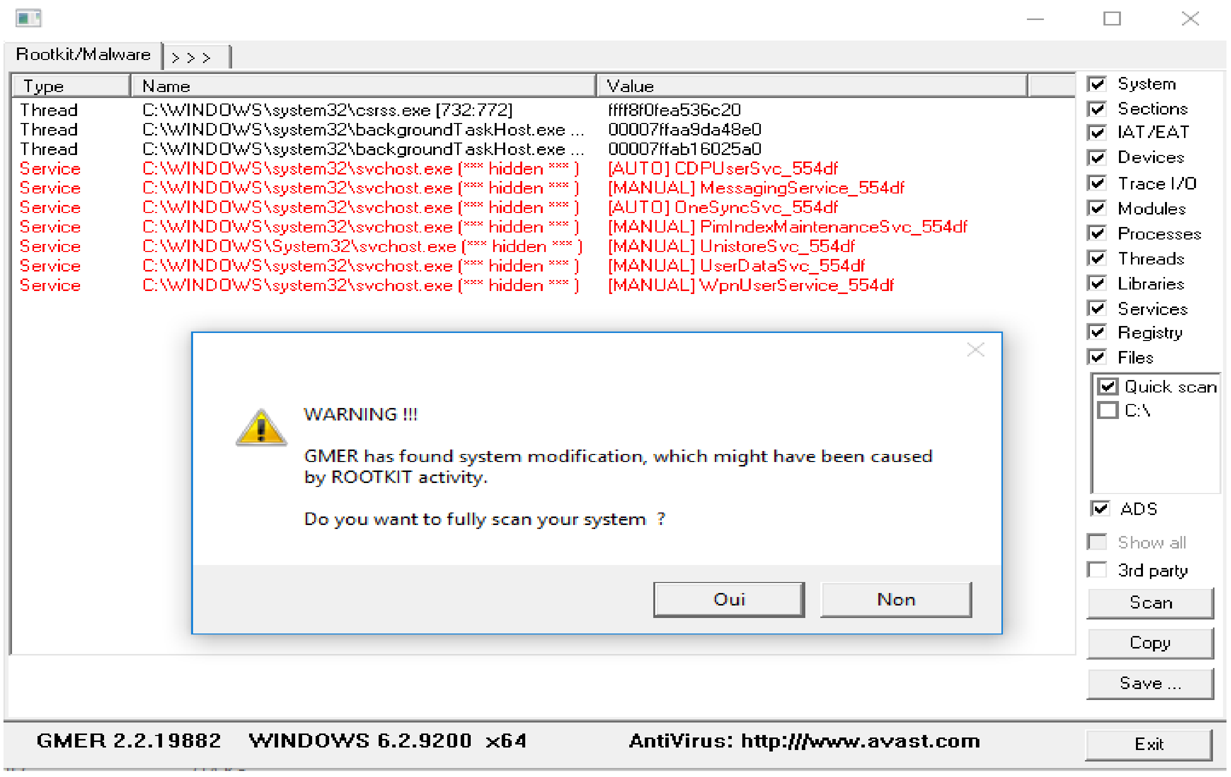
Processes | Modules | Services | Files | Registry | Rootkit/Malware | CMD | Autostart

```
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.
Step 3:



```
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

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: