**Branch**: Information Technology & Computer Science Engineering

**Subject**: Security Laboratory Manual

**Authors:**

**Ms.Sangeethapriya. J**

**Ms. Sheelavathi. A**

**Ms.Muthukarupaee. K**

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Exercise No**. | **Name of the Exercise** | **Page No** |
| 1a | Implementation of **Ceasercipher** Substitution Techniques. | **4** |
| 1b | **PlayFaircipher** Substitution Techniques implementation. | **6** |
| 1c | **Hillcipher** Substitution Techniques implementation. | **13** |
| 1d | **Vigenerecipher** Substitution Techniques implementation. | **16** |
| 2a | **RailFence** transposition Techniques implementation. | **18** |
| 2b | **Row & Column Transformation** transposition Technique for encryption and decryption. | **21** |
| 3 | Implementation of **DES** algorithm. | **24** |
| 4 | Implementation of **AES** algorithm. | **26** |
| 5 | Implementation of **RSA** algorithm using HTML & JavaScript. | **29** |
| 6 | Implementation of **Diffie-Hellman** key exchange algorithm for a given input. | **33** |
| 7 | Compute the message digest of a text using **SHA-1** algorithm. | **35** |
| 8 | Implementation of the **Signature Scheme**-Digital Signature Standard. | **37** |
| -9 | Demonstrate **intrusion detection system**(ids) using Snort or any other s/w. | **39** |
| 10 | Automated **Attack and Penetration Tools** Exploring N-Stalker, Vulnerability Assessment Tool. | **43** |
| 11a | Defeating Malware - **Building Trojans** | **47** |
| 11b | **Rootkit Hunter** | **49** |
| 12 | Setup a **honey pot** and monitor the honeypot on network | **53** |

|  |  |
| --- | --- |
| **Ex. No : 1(a)**  **Date :** | **Python Program for Encryption and Decryption Using Ceaser Cipher** |

**AIM:**

To write a python program to encrypt and decrypt the given input message by using Ceaser Cipher encryption technique

**ALGORITHM:**

1. Each letter in the plaintext is substituted in the Ceaser Cipher by a letter that is a predetermined number of positions farther down the alphabet.

2. With a left shift of 3, for instance, D would become A, E would become B, and so on.

3. Modular arithmetic can also be used to express the encryption by first converting the letters' letters into integers, with A = 0, B = 1, and Z = 25 in the scheme.

4. ncryption of a letter x by a shift n can be described mathematically as, ***En(x) = (x + n) mod26***

***5.*** Decryption is performed similarly,

***Dn (x)=(x - n) mod26***

**PROGRAM:**

***CaesarCipher.py***

pt1=input("enter :")

pt=list(filter(str.strip,pt1))

key=int(input("enter:"))

alpha=['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z']

print("plaintext to ciphertext")

s=""

for i in range(0,len(pt)):

eff=(alpha.index(pt[i])+ key)%len(alpha)

s=s+alpha[eff]

print(s,end="")

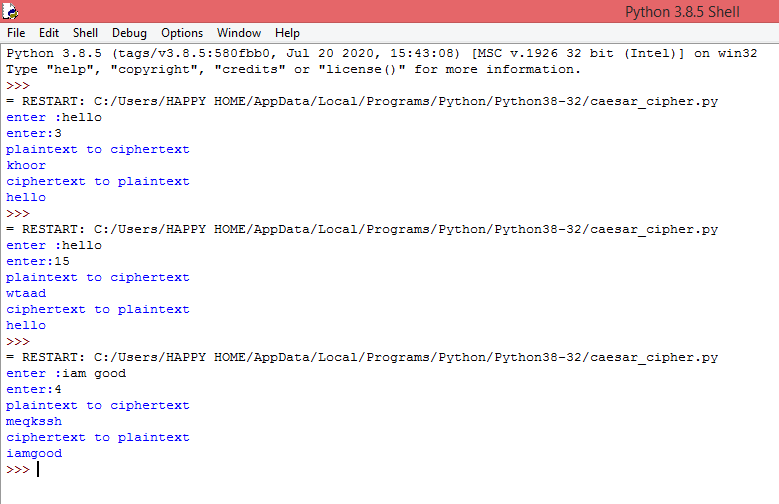
print("\nciphertext to plaintext")

for j in range(0,len(s)):

eff1=(alpha.index(s[j])- key)%len(alpha)

print(alpha[eff1],end="")

**OUTPUT:**



**RESULT:**

As a result, the ceaser cipher cryptographic technique has been built, and the output has been confirmed.

|  |  |
| --- | --- |
| **Ex. No : 1(b)**  **Date :** | **Implementation of Playfair Cipher technique** |

**AIM:**

To develop a python program that uses the play fair Cipher Substitution algorithm to encrypt plain text and decipher cypher text.

**ALGORITHM:**

1. A communication would be broken up into digrams to be encrypted (groups of 2 letters)

2. "HelloWorld," for instance, becomes "HE LL OW OR LD."

3. The key table will be used to replace these digrams.

4.Messages with an odd number of characters typically append an uncommon letter, such as "X," to complete the final digram because encryption requires pairs of letters.

5.In the key table, the two letters of the digram are regarded as the opposite corners of a rectangle. Apply each pair of letters in the plaintext to the following four rules, in that order, to complete the substitution.

**PROGRAM:**

***playfairCipher.py***

PT = input("Enter your text for encryption : ").strip().upper()

plainText = list(PT)

k = input("Enter the key : ").strip().upper()

key = list(k)

def process(plainText):

plaintext = []

for p in plainText :

if p!=" ":

plaintext += p

else:

continue

if p=='J':

ind = plaintext.index(p)

plaintext[ind] = 'I'

for p in plaintext :

x="X"

if len(plaintext)%2!=0:

plaintext += x

return plaintext

plaintext =process(plainText)

pair\_of\_plaintext = []

def Make\_Pair\_PT(plaintext):

for pp in range(0,len(plaintext),2):

pair\_of\_plaintext.append(plaintext[pp:pp+2])

for pair in range(0,len(pair\_of\_plaintext)):

if pair\_of\_plaintext[pair][0] == pair\_of\_plaintext[pair][1]:

pair\_of\_plaintext[pair][1] = 'X'

print(pair\_of\_plaintext)

return pair\_of\_plaintext

print("Plaintext is divided into pairs, which are :")

Make\_Pair\_PT(plaintext)

ALPH = ['A','B','C','D','E','F','G','H','I','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z']

ALPH\_len = len(ALPH)

def Playfair\_Matrix(ALPH,ALPH\_len):

#This is used to keep track of the key alphabet remaining

key\_len = len(key)

#index ind\_ALPHA to keep track of the alphabet in the ALPH list

ind\_ALPHA = 0

#This is used to pop those elements in ALPH list as these alphabet is in the key list (which is already taken in the Playfair\_matrix )

for k in key:

for a in ALPH:

if k==a:

ALPH\_index = ALPH.index(a)

ALPH.pop(ALPH\_index)

#It is a 5X5 matrix, containing the alphabets in some order like ---firstly key is placed in it then remaining alphabet is placed in the same order as they appear (NOTE---If the alphabet which is in key, that alphabet is not placed in the remaining block.)

#(NOTE---Preference of I/J is same and kept in the same block)

playfair\_matrix = [[0,0,0,0,0],[0,0,0,0,0],[0,0,0,0,0],[0,0,0,0,0],[0,0,0,0,0]]

ind=0

#Built playfair\_matrix for different key you entered

for i in range(0,5):

for j in range(0,5):

if key\_len != 0:

#This will place all the alphabets in the key

playfair\_matrix[i][j]=key[ind]

ind += 1

key\_len -=1

else:

#Fill remaining entries in the playfair\_matrix i.e. place remaining alphabet to playfair\_matrix list from ALPH list

if ALPH\_len !=0:

playfair\_matrix[i][j] = ALPH[ind\_ALPHA]

ALPH\_len -= 1

ind\_ALPHA +=1

return playfair\_matrix

#call

playfair\_matrix = Playfair\_Matrix(ALPH,ALPH\_len)

print("Playfair Matrix 5X5 is:")

print(playfair\_matrix)

# Type 1:- Rectangle or square shape (means a block)

#Locate the first alphabet of the pair in playfair\_matrix[] and find the row where tye other alphabet of the pair lies such that we get the rectangle or square shape block

def Type1(loc\_first\_alpha\_1i,loc\_first\_alpha\_1j,loc\_second\_alpha\_2i,loc\_second\_alpha\_2j):

for i in range(0,len(pair\_of\_plaintext)):

cipher\_pair = playfair\_matrix[loc\_first\_alpha\_1i][loc\_second\_alpha\_2j]

cipher\_pair = cipher\_pair + playfair\_matrix[loc\_second\_alpha\_2i][loc\_first\_alpha\_1j]

print(cipher\_pair)

return cipher\_pair

#Type2:--Alphabets in pair are in same row

def Type2(loc\_first\_alpha\_1i,loc\_first\_alpha\_1j,loc\_second\_alpha\_2i,loc\_second\_alpha\_2j):

for i in range(0,len(pair\_of\_plaintext)):

#check first alphabet of the pair is at the extreme end

if loc\_first\_alpha\_1j ==4:

cipher\_pair = playfair\_matrix[loc\_first\_alpha\_1i][0]

cipher\_pair = cipher\_pair + playfair\_matrix[loc\_first\_alpha\_1i][loc\_second\_alpha\_2j+1]

#check second alphabet of the pair is at the extreme end

if loc\_second\_alpha\_2j ==4:

cipher\_pair = playfair\_matrix[loc\_first\_alpha\_1i][loc\_first\_alpha\_1j+1]

cipher\_pair = cipher\_pair + playfair\_matrix[loc\_first\_alpha\_1i][0]

#If first or second alphabet of the pair are not at the extreme end

if (loc\_first\_alpha\_1j!=4) and (loc\_second\_alpha\_2j!=4):

cipher\_pair = playfair\_matrix[loc\_first\_alpha\_1i][loc\_first\_alpha\_1j+1]

cipher\_pair = cipher\_pair + playfair\_matrix[loc\_second\_alpha\_2i][loc\_second\_alpha\_2j+1]

print(cipher\_pair)

return cipher\_pair

#(Type3 :-- Alphabets in pair are in same col)

def Type3(loc\_first\_alpha\_1i,loc\_first\_alpha\_1j,loc\_second\_alpha\_2i,loc\_second\_alpha\_2j):

for i in range(0,len(pair\_of\_plaintext)):

#since both alphabets of the pair are in same col, so either first alphabet or second alphabet lies at the bottom extreme, but not both simultaneousy

#check first alphabet of the pair is at the bottom extreme end

if loc\_first\_alpha\_1i ==4:

cipher\_pair = playfair\_matrix[0][loc\_first\_alpha\_1j]

cipher\_pair = cipher\_pair + playfair\_matrix[loc\_second\_alpha\_2i+1][loc\_second\_alpha\_2j]

if loc\_second\_alpha\_2i ==4:

cipher\_pair = playfair\_matrix[loc\_first\_alpha\_1i+1][loc\_first\_alpha\_1j]

cipher\_pair = cipher\_pair + playfair\_matrix[0][loc\_second\_alpha\_2j]

#If first or second alphabet of the pair are not at the bottom extreme end

if (loc\_first\_alpha\_1i!=4) and (loc\_second\_alpha\_2i!=4):

cipher\_pair = playfair\_matrix[loc\_first\_alpha\_1i+1][loc\_first\_alpha\_1j]

cipher\_pair = cipher\_pair + playfair\_matrix[loc\_second\_alpha\_2i+1][loc\_second\_alpha\_2j]

print(cipher\_pair)

return cipher\_pair

#(Encryption)

cipher\_pair = []

#combine each pair of ciphertext returned into ***cipher\_text\_final[] list***

cipher\_text\_final = []

def Encryption(playfair\_matrix, pair\_of\_plaintext,cipher\_pair):

#search each pair alphabet of plaintext in playfair\_matrix[] and observe the location of the pair alphabet

print("Encrypted Ciphertext is --->")

for pp in pair\_of\_plaintext :

for i in range(0,5):

for j in range(0,5):

#The below two if statement is for use when in a pair (in plaintext) of alphabet is in different row and col

if pp[0]==playfair\_matrix[i][j]:

loc\_first\_alpha\_1i = i

loc\_first\_alpha\_1j = j

if pp[1] == playfair\_matrix[i][j]:

loc\_second\_alpha\_2i = i

loc\_second\_alpha\_2j = j

#if both alphabets of pair are in different row and col

if (loc\_first\_alpha\_1i!=loc\_second\_alpha\_2i) and (loc\_first\_alpha\_1j!= loc\_second\_alpha\_2j):

print("{}:".format(pp),end=" ")

cipher\_Type1 =Type1(loc\_first\_alpha\_1i, loc\_first\_alpha\_1j, loc\_second\_alpha\_2i, loc\_second\_alpha\_2j)

cipher\_text\_final.append(cipher\_Type1)

#pair (in plaintext) of alphabets is in same row

if loc\_first\_alpha\_1i == loc\_second\_alpha\_2i:

print("{}:".format(pp), end =" ")

cipher\_Type2 = Type2(loc\_first\_alpha\_1i, loc\_first\_alpha\_1j, loc\_second\_alpha\_2i, loc\_second\_alpha\_2j)

cipher\_text\_final.append(cipher\_Type2)

#pair (in plaintext) of alphabets is in same col

if loc\_first\_alpha\_1j == loc\_second\_alpha\_2j:

print("{}:".format(pp), end=" ")

cipher\_Type3 = Type3(loc\_first\_alpha\_1i, loc\_first\_alpha\_1j, loc\_second\_alpha\_2i, loc\_second\_alpha\_2j)

cipher\_text\_final.append(cipher\_Type3)

#call

Encryption(playfair\_matrix, pair\_of\_plaintext, cipher\_pair)

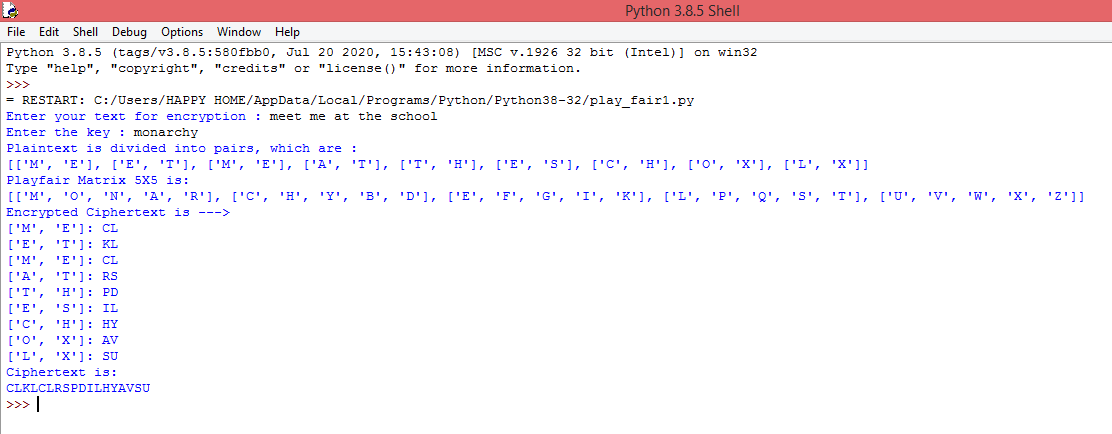
#print combined pair if Ciphertext

print("Ciphertext is:")

for c in cipher\_text\_final :0

print("".join(c), end="")

**OUTPUT:**



**RESULT:**

Thus, the program for playfair cypher encryption and decryption algorithm has been implemented and successfully tested the results.

|  |  |
| --- | --- |
| **Ex. No : 1(c)**  **Date :** | **Implementation of Hill Cipher substitution technique** |

**AIM:**

To write a java program that uses the Hill cypher substitution technique to encrypt and decrypt data.

**ALGORITHM:**

1. In the Hill cipher algorithm, Each letter is represented by a modulo 26 number.

2. Multiply each block of n letters by an invertible n x n matrix modulus 26 to encrypt a message.

3. Multiply each block by the inverse of the matrix used for encryption to decrypt the message.

4. The cipher key is the encryption matrix, which should be chosen at random from the set of invertible n n matrices (modulo 26).

5. The cipher can be adapted to any number of letters in an alphabet.

6. All arithmetic must be performed modulo the number of letters, not modulo 26.

**PROGRAM:**

***HillCipher.java***

class hillCipher {

/\* 3x3 key matrix for 3 characters at once \*/

public static int[][] keymat = new int[][] { { 1, 2, 1 }, { 2, 3, 2 },

{ 2, 2, 1 } }; /\* key inverse matrix \*/

public static int[][] invkeymat = new int[][] { { -1, 0, 1 }, { 2, -1, 0 }, { -2, 2, -1 } };

public static String key = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

private static String encode(char a, char b, char c) {

String ret = "";

int x, y, z;

int posa = (int) a - 65;

int posb = (int) b - 65;

int posc = (int) c - 65;

x = posa \* keymat[0][0] + posb \* keymat[1][0] + posc \* keymat[2][0];

y = posa \* keymat[0][1] + posb \* keymat[1][1] + posc \* keymat[2][1];

z = posa \* keymat[0][2] + posb \* keymat[1][2] + posc \* keymat[2][2];

a = key.charAt(x % 26);

b = key.charAt(y % 26);

c = key.charAt(z % 26);

ret = "" + a + b + c;

return ret;

}

private static String decode(char a, char b, char c) {

String ret = "";

int x, y, z;

int posa = (int) a - 65;

int posb = (int) b - 65;

int posc = (int) c - 65;

x = posa \* invkeymat[0][0] + posb \* invkeymat[1][0] + posc \* invkeymat[2][0];

y = posa \* invkeymat[0][1] + posb \* invkeymat[1][1] + posc \* invkeymat[2][1];

z = posa \* invkeymat[0][2] + posb \* invkeymat[1][2] + posc \* invkeymat[2][2];

a = key.charAt((x % 26 < 0) ? (26 + x % 26) : (x % 26));

b = key.charAt((y % 26 < 0) ? (26 + y % 26) : (y % 26));

c = key.charAt((z % 26 < 0) ? (26 + z % 26) : (z % 26));

ret = "" + a + b + c;

return ret;

}

public static void main(String[] args) throws java.lang.Exception {

String msg;

String enc = "";

String dec = "";

int n;

msg = ("SecurityLaboratory");

System.out.println("simulation of Hill Cipher\n-------------------------");

System.out.println("Input message : " + msg);

msg = msg.toUpperCase();

msg = msg.replaceAll("\\s", "");

/\* remove spaces \*/ n = msg.length() % 3;

/\* append padding text X \*/ if (n != 0) {

for (int i = 1; i <= (3 - n); i++) {

msg += 'X';

}

}

System.out.println("padded message : " + msg);

char[] pdchars = msg.toCharArray();

for (int i = 0; i < msg.length(); i += 3) {

enc += encode(pdchars[i], pdchars[i + 1], pdchars[i + 2]);

}

System.out.println("encoded message : " + enc);

char[] dechars = enc.toCharArray();

for (int i = 0; i < enc.length(); i += 3) {

dec += decode(dechars[i], dechars[i + 1], dechars[i + 2]);

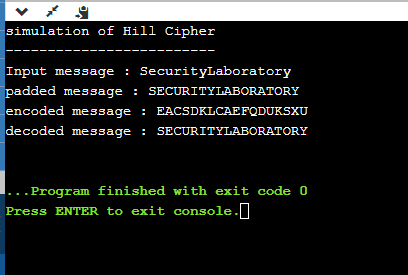
}

System.out.println("decoded message : " + dec);

}

}

**OUTPUT:**



**RESULT**:

Thus, the java program for the hill cipher algorithm has been implemented and successfully tested for given input message.

|  |  |
| --- | --- |
| **Ex. No : 1(d)**  **Date :** | **Implementation of Vigenere Cipher Technique** |

**AIM:**

To write and implement a java program using the vigenere cipher substitution technique.

**ALGORITHM:**

1. The Vigenere cipher encrypts alphabetic text using a sequence of Caesar ciphers depending on a keyword's letters.

2. It is a simple kind of polyalphabetic substitution.

3. To encrypt, a Vigenere square or Vigenere table of alphabets can be employed.

4. It consists of the alphabet spelled out 26 times in separate rows, with each alphabet moved cyclically to the left compared to the preceding alphabet, corresponding to the Caesar stream cipher of 26 characters.

5. The cipher uses a different alphabet from one of the rows at various stages during the encryption process.

6. A repeating keyword determines the alphabet at each stage.

**PROGRAM:**

***vigenereCipher.java***

public class vigenereCipher {

static String encode(String text, final String key) {

String res = "";

text = text.toUpperCase();

for (int i = 0, j = 0; i < text.length(); i++) {

char c = text.charAt(i);

if (c < 'A' || c > 'Z') {

continue;

}

res += (char) ((c + key.charAt(j) - 2 \* 'A') % 26 + 'A');

j = ++j % key.length();

}

return res;

}

static String decode(String text, final String key) {

String res = "";

text = text.toUpperCase();

for (int i = 0, j = 0; i < text.length(); i++) {

char c = text.charAt(i);

if (c < 'A' || c > 'Z') {

continue;

}

res += (char) ((c - key.charAt(j) + 26) % 26 + 'A');

j = ++j % key.length();

}

return res;

}

public static void main(String[] args) throws java.lang.Exception {

String key = "VIGENERECIPHER";

String msg = "SecurityLaboratory";

System.out.println("Simulating Vigenere Cipher\n------------------------");

System.out.println("Input Message : " + msg);

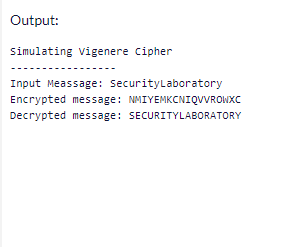
String enc = encode(msg, key);

System.out.println("Encrypted Message : " + enc);

System.out.println("Decrypted Message : " + decode(enc, key));

}}

**OUTPUT:**



**RESULT:**

As a result, the java application for the vigenere cypher encryption and decryption technique has been successfully constructed and tested.

|  |  |
| --- | --- |
| **Ex. No : 2(a)**  **Date :** | **Implementation of Rail Fence Cipher Transposition Technique** |

**AIM:**

To write and implement a java program using rail fence transposition technique for given plain text message.

**ALGORITHM**:

1. In the rail fence cypher, the plaintext is written downwards and diagonally on consecutive "rails" of an imagined fence, then shifted up when we reach the bottom rail.

2. When we reach the top rail, we repeat the process until the complete plaintext is printed out.

3. After that, the message is read in columns.

**PROGRAM:**

***railFenceCipher.java***

class railfenceCipherHelper {

int depth;

String encode(String msg, int depth) throws Exception {

int r = depth;

int l = msg.length();

int c = l / depth;

int k = 0;

char mat[][] = new char[r][c];

String enc = "";

for (int i = 0; i < c; i++) {

for (int j = 0; j < r; j++) {

if (k != l) {

mat[j][i] = msg.charAt(k++);

} else {

mat[j][i] = 'X';

}

}

}

for (int i = 0; i < r; i++) {

for (int j = 0; j < c; j++) {

enc += mat[i][j];

}

}

return enc;

}

String decode(String encmsg, int depth) throws Exception {

int r = depth;

int l = encmsg.length();

int c = l / depth;

int k = 0;

char mat[][] = new char[r][c];

String dec = "";

for (int i = 0; i < r; i++) {

for (int j = 0; j < c; j++) {

mat[i][j] = encmsg.charAt(k++);

}

}

for (int i = 0; i < c; i++) {

for (int j = 0; j < r; j++) {

dec += mat[j][i];

}

}

return dec;

}

}

class railFenceCipher {

public static void main(String[] args) throws java.lang.Exception {

railfenceCipherHelper rf = new railfenceCipherHelper();

String msg, enc, dec;

msg = "Anna University, Chennai";

int depth = 2;

enc = rf.encode(msg, depth);

dec = rf.decode(enc, depth);

System.out.println("Simulating Railfence Cipher\n-------------------------");

System.out.println("Input Message : " + msg);

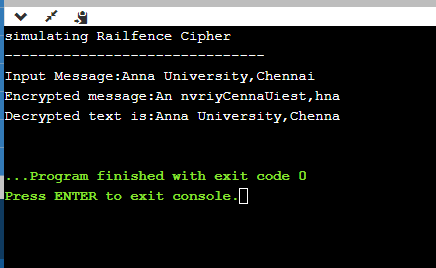
System.out.println("Encrypted Message : " + enc);

System.out.printf("Decrypted Message : " + dec);

}

}

**OUTPUT:**



**RESULT:**

Thus the program written in java to execute Rail Fence Transposition Technique was done and the corresponding output verified successfully.

|  |  |
| --- | --- |
| **Ex. No : 2(b)**  **Date :** | **Implementation of Row and Column Transformation Technique** |

**AIM:**

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

**ALGORITHM:**

1. Take the plain text hello world and apply the simple columnar transposition technique shown below.

|  |  |  |  |
| --- | --- | --- | --- |
| h | e | l | l |
| o | w | o | r |
| l | d |  |  |

2. The plain text characters are arranged horizontally, while the cypher text is arranged vertically as: holewdlo lr.

3. The receiver must now decrypt the cypher text to plain text using the same table.

**PROGRAM:**

***TransCipher.java***

import java.util.\*;

class TransCipher {

public static void main(String args[]) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter the plain text");

String pl = sc.nextLine();

sc.close();

String s = "";

int start = 0;

for (int i = 0; i < pl.length(); i++) {

if (pl.charAt(i) == ' ') {

s = s + pl.substring(start, i);

start = i + 1;

}

}

s = s + pl.substring(start);

System.out.print(s);

System.out.println();

// end of space deletion

int k = s.length();

int l = 0;

int col = 4;

int row = s.length() / col;

char ch[][] = new char[row][col];

for (int i = 0; i < row; i++) {

for (int j = 0; j < col; j++) {

if (l < k) {

ch[i][j] = s.charAt(l);

l++;

} else {

ch[i][j] = '#';

}

}

}

// arranged in matrix

char trans[][] = new char[col][row];

for (int i = 0; i < row; i++) {

for (int j = 0; j < col; j++) {

trans[j][i] = ch[i][j];

}

}

for (int i = 0; i < col; i++) {

for (int j = 0; j < row; j++) {

System.out.print(trans[i][j]);

}

}

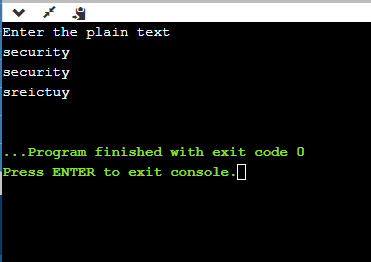
// display

System.out.println();

}

}

**OUTPUT:**



**RESULT:**

Thus, the Java program for Row and Column Transposition Technique has been implemented and successfully tested.

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

**AIM:**

To employ the Data Encryption Standard (DES) Algorithm in a practical application such as User Message Encryption.

**ALGORITHM:** 1. Generate a DES Key.

2. Create a Cipher instance from the Cipher class and enter the information below, separated by a slash (/).

a. Algorithm title b. Mode (optional)

c. Padding design (optional)

3. Convert a String to a Byte[] array.

4. Encrypt Cipher with the Cipher.doFinal() method after putting it in encrypt mode.

5. Put Cipher in decrypt mode and use the Cipher.doFinal() method to decrypt it.

**PROGRAM:**

***DES.java***

import java.security.InvalidKeyException;

import java.security.NoSuchAlgorithmException;

import javax.crypto.BadPaddingException;

import javax.crypto.Cipher;

import javax.crypto.IllegalBlockSizeException;

import javax.crypto.KeyGenerator;

import javax.crypto.NoSuchPaddingException;

import javax.crypto.SecretKey;

public class DES

{

public static void main(String[] argv) {

try{

System.out.println("Message Encryption Using DES Algorithm\n-------");

KeyGenerator keygenerator = KeyGenerator.getInstance("DES");

SecretKey myDesKey = keygenerator.generateKey();

Cipher desCipher;

desCipher = Cipher.getInstance("DES/ECB/PKCS5Padding");

desCipher.init(Cipher.ENCRYPT\_MODE, myDesKey);

byte[] text = "Secret Information ".getBytes();

System.out.println("Message [Byte Format] : " + text);

System.out.println("Message : " + new String(text));

byte[] textEncrypted = desCipher.doFinal(text);

System.out.println("Encrypted Message: " + textEncrypted);

desCipher.init(Cipher.DECRYPT\_MODE, myDesKey);

byte[] textDecrypted = desCipher.doFinal(textEncrypted);

System.out.println("Decrypted Message: " + new String(textDecrypted));

}catch(NoSuchAlgorithmException e){

e.printStackTrace();

}catch(NoSuchPaddingException e){

e.printStackTrace();

}catch(InvalidKeyException e){

e.printStackTrace();

}catch(IllegalBlockSizeException e){

e.printStackTrace();

}catch(BadPaddingException e){

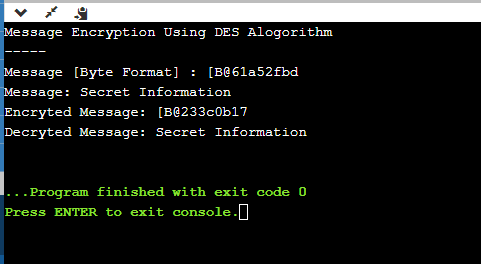
e.printStackTrace();

}

}

}

**OUTPUT:**



**RESULT:**

Thus, the DES Algorithm Java program has been implemented and output was successfully tested.

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

**AIM:**

To employ the Advanced Encryption Standard (AES) Algorithm such as URL Encryption.

**ALGORITHM:**

1. AES is built on a design principle known as substitution-permutation.

2. Unlike DES, AES employs a variant of Rijndael rather than a Feistel network.

3. It has a 128-bit fixed block size and a key size of 128, 192, or 256 bits.

4. AES operates on a 4 x 4 column-major order bytes array known as 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);

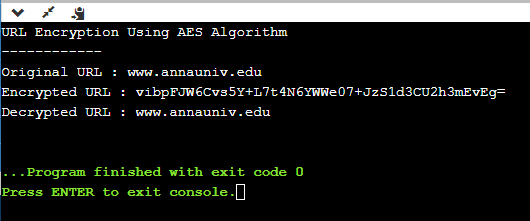
System.out.println("Encrypted URL : " + encryptedString);

System.out.println("Decrypted URL : " + decryptedString);

}

}

**OUTPUT:**

****

**RESULT:**

Thus, the AES Algorithm for URL Encryption has been implemented in Java, and the result has been correctly confirmed.

|  |  |
| --- | --- |
| **Ex. No : 5**  **Date :** | **RSA Algorithm** |

**AIM:**

To use HTML and Javascript to implement the RSA (Rivest-Shamir-Adleman) algorithm.

**ALGORITHM:**

1. Select two prime numbers, p and q.

2. Determine the values of n and p.

3. Determine the value of e (public key)

4. Using gcd, compute the value of d (private key) ()

5. Carry out the encryption and decryption

a. Encryption is given as,

c = te mod n

b. Decryption is given as,

t = cd 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;

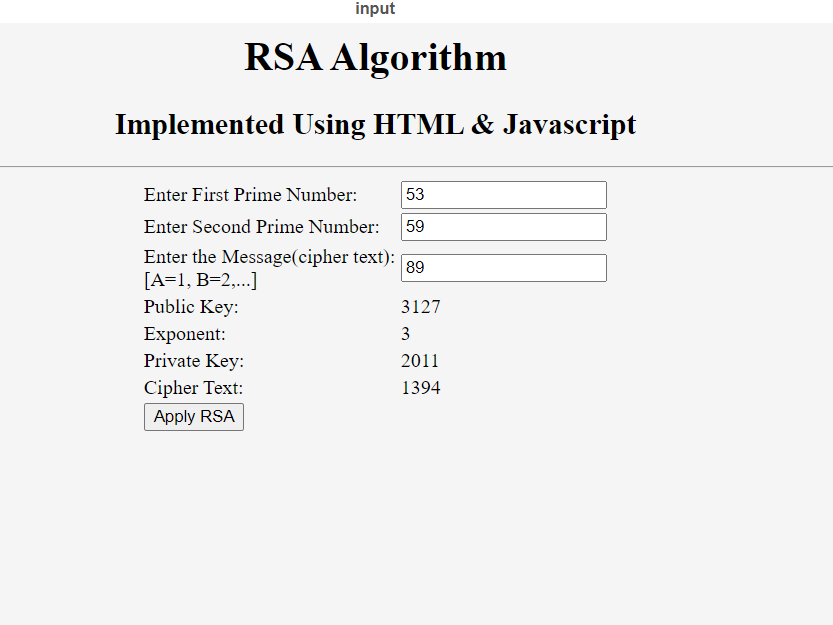
document.getElementById('ciphertext').innerHTML = ct;

}

</script>

</html>

**OUTPUT:**



**RESULT:**

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

|  |  |
| --- | --- |
| **Ex. No : 6**  **Date :** | **Diffie-Hellman key Exchange Algorithm Implementation** |

**AIM:**

The main goal is to apply the Diffie-Hellman Key Exchange algorithm for a specific problem and implement it using any programming language.

**ALGORITHM:**

1. Alice and Bob publicly agree to use p = 23 modulus and g = 5 base (which is a primitive root modulo 23).

2. Alice selects a secret integer a = 4 and sends Bob A = ga mod p.

o A = 54 mod 23 = 4

3. Bob selects a secret integer b = 3 and sends it to Alice as B = gb mod p.

o B = 53 mod 23 = 10

4. Alice calculates s = Ba mod p.

o s = 104 mod 23 = 18

5. Bob calculates s = Ab mod p.

o s = 43 mod 23 = 18

6. Alice and Bob now have a secret in common (the number 18).

**PROGRAM:**

***DiffieHellman.java***

class DiffieHellman {

public static void main(String args[]) {

int p = 23;

int g = 5;

int x = 4;

int y = 3;

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);

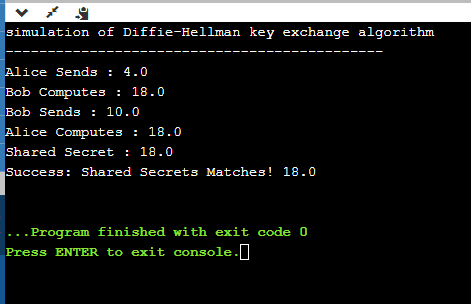
else

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

}

}

**OUTPUT:**

****

**RESULT:**

As a result, the Diffie-Hellman key exchange method was implemented in Java and its output was properly validated.

|  |  |
| --- | --- |
| **Ex. No : 7**  **Date :** | **Implementation of SHA-1 Algorithm** |

**AIM:**

Using the SHA-1 algorithm, compute the message digest of a text.

**ALGOIRTHM:**

1. Append Padding Bits ALGORITHM

2. Append Length - A total of 64 bits are appended to the end.

3. Set up Processing Functions

4. Determine Processing Constants

5. Set up Buffers

6. Message processing in 512-bit blocks (L blocks in total message)

**PROGRAM:**

***sha1.java***

import java.security.\*;

public class sha1 {

public static void main(String[] a) {

try {

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

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

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

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

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

String input = "";

md.update(input.getBytes());

byte[] output = md.digest();

System.out.println();

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

input = "abc";

md.update(input.getBytes());

output = md.digest();

System.out.println();

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

input = "abcdefghijklmnopqrstuvwxyz";

md.update(input.getBytes());

output = md.digest();

System.out.println();

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

System.out.println();

} catch (Exception e) {

System.out.println("Exception:" + e);

}

}

private static String bytesToHex(byte[] b) {

char hexDigit[] = { '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F' };

StringBuffer buf = new StringBuffer();

for (byte aB : b) {

buf.append(hexDigit[(aB >> 4) & 0x0f]);

buf.append(hexDigit[aB & 0x0f]);

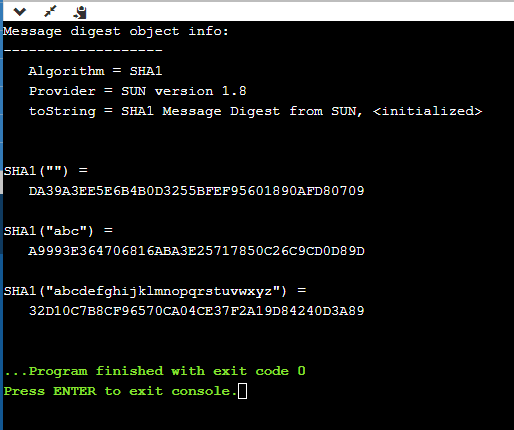
}

return buf.toString();

}

}

**OUTPUT:**



**RESULT:**

Thus SHA-1 implemented and computed values were verified.

|  |  |
| --- | --- |
| **Ex. No : 8**  **Date :** | **Digital Signature Standard** |

**AIM:**

To apply the SIGNATURE SCHEME – Digital Signature Standard into practise.

**ALGORITHM:**

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

**PROGRAM:**

import java.security.KeyPair;

import java.security.KeyPairGenerator;

import java.security.PrivateKey;

import java.security.Signature;

import java.util.Scanner;

public class CreatingDigitalSignature {

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

Scanner sc = new Scanner(System.in);

System.out.println("Enter some text");

String msg = sc.nextLine();

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

keyPairGen.initialize(2048);

KeyPair pair = keyPairGen.generateKeyPair();

PrivateKey privKey = pair.getPrivate();

Signature sign = Signature.getInstance("SHA256withDSA");

sign.initSign(privKey);

byte[] bytes = "msg".getBytes();

sign.update(bytes);

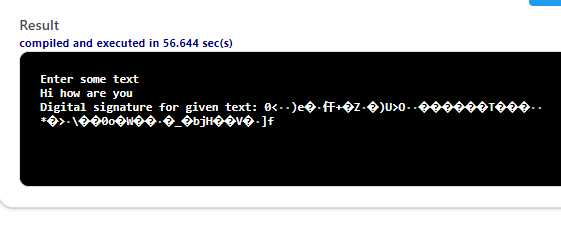
byte[] signature = sign.sign();

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

}

}

**OUTPUT:**



**RESULT:**

As a result, the Digital Signature Standard Signature Scheme has been applied, and the output has been correctly confirmed.

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

**AIM:**

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

**STEPS ON CONFIGURING AND INTRUSION DETECTION:**

**1**. Download Snort from the Snort.org website. (http://www.snort.org/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 every file from the Rules folder to the enlarged folder. The rules should be pasted into the C:\Snort\rules folder.

6. From the etc folder of the unzipped folder, copy the snort.conf file. The appropriate location is the C:Snort\etc  folder. Overwrite existing files. It should be noted that for Snort to function, you must edit your snort.conf file in order to download a new file.

7. Open a command prompt (cmd.exe) Go to the C:\Snort\bin folder. (type cd\snort\bin at the command prompt)

Eighth. To start (run) Snort in sniffer mode, use the following command:

snort -dev -i 3

-i specifies the interface number. You must choose the correct interface number. i have three

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

Check the interface list using the following command:

Snort -W

Snort-W

Find interface

Look at the index number and search Microsoft to find out which interface to use. As you can see in the example above, the other interface is for VMWare. My interface is 3.

9. You must set up the snort.conf file in accordance with your network environment in order to use Snort in IDS mode.

10. Locate the line in the snort.conf file where you may select which network addresses to protect.:

var HOME\_NET 192.168.1.0/24 (usually everything appears here)

11. If your network has DNS\_SERVERS addresses, you can also set them.

Example:

example snort

12. Set the rules folder's path as the value for the RULE PATH variable.

var RULE PATH Path to rules: c:snortrules

13. Replace the name and path on your system with the path of all library files. furthermore, the path of the snort dynamicpreprocessorvariable to be modified.

C:\Snort\lib\snort dynamic preprocessor

This must be done to each and every library file located in the "C:Snortlib" folder. The previous location might be "/usr/local/lib/...". You must substitute your system path for that route. The C:Snortlib directory14. 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. Find the "output log" test in snort.conf and add the following line to add log files to store alerts produced by snort.

Snort-alerts.ids as output from alert fast

19. Comment (add a #) the whitelist and blacklist found at $WHITE LIST PATH/white list.rules.

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 board)

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

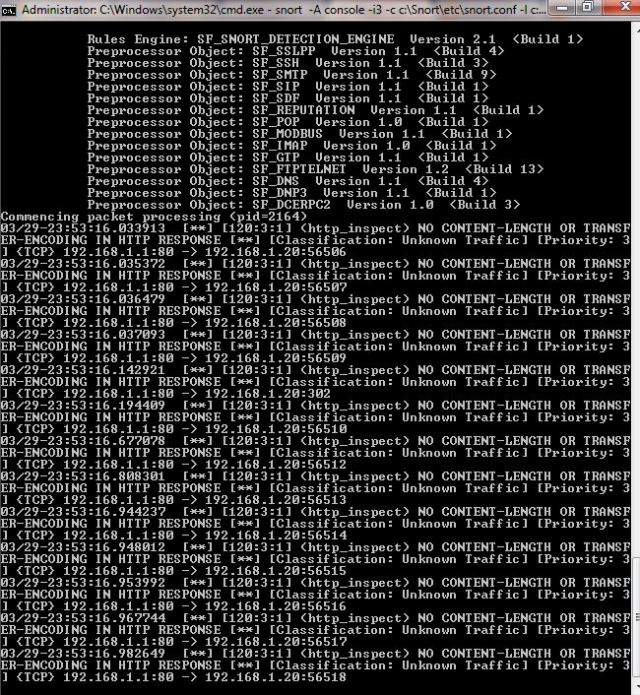
24. To generate a log file in ASCII mode, you can use the following command while running snort in IDS mode.

snort -A console -i3 -c c:\Snort\etc\snort.conf -l c:\Snort\log -K ascii

25. Scan a computer running Snort from another computer using PING or NMap (ZenMap).

26. After or during the scan, you can check the snort-alerts.ids file in the logs folder to make sure it is properly logged. An IP address folder appears.

Snort monitoring traffic –



**RESULT:**

Therefore, an intrusion detection system IDS was demonstrated using the open source Snort intrusion detection tool.

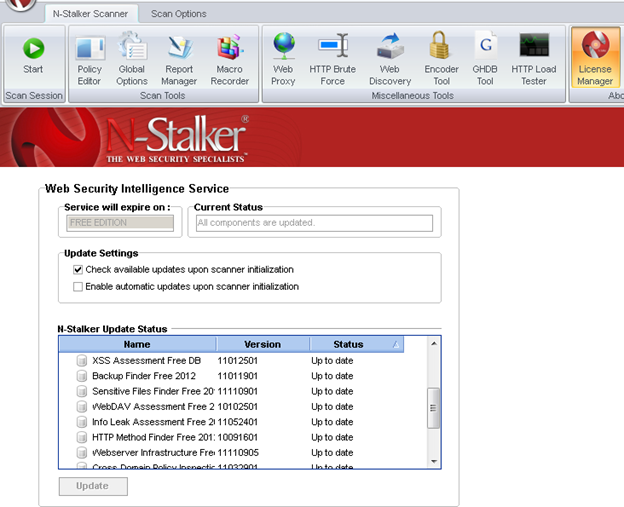
|  |  |
| --- | --- |
| **Ex. No : 10**  **Date :** | **Exploring N-Stalker, a Vulnerability Assessment Tool** |

**AIM:**

To download and install 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.  
• **Includes** **the** **popular** N-Stealth HTTP **security** **scanner** and **signature** **database** **with** 35,000 **web** attack **signatures.**  
• This tool **is** **available** in both free and paid **versions.**  
• Before scanning **targets,** go to **the** **License** **Manager** **tab** **and** perform **an** **update.** **•** **After** **updating,** the **status** **is** **displayed** **as** **current.**  
• **N-Stalker** **must** **be** **downloaded** and **installed** from www.nstalker.com.  
  
1. **Launch** N-Stalker **on** **your** Windows computer. The program is installed **in** Start ➪ Programs ➪ N-Stalker ➪ N-Stalker Free Edition. 2. Enter **the** host address or **address** range to scan.  
3. Click Start Scan.  
**Four.** N-Stalker Report Manager will prompt **you** **when** **the** **scan** **is** **complete.**  
5. **You** **can** **choose** **the** format **of** the resulting report **by** **selecting** Generate HTML.  
6. **Check** **for** **vulnerabilities** **in** the HTML **report.**



Go to Scan Session and enter the target URL.

You can choose from four options in your scan policy.

• Manual tests that crawl websites and wait for manual attacks.

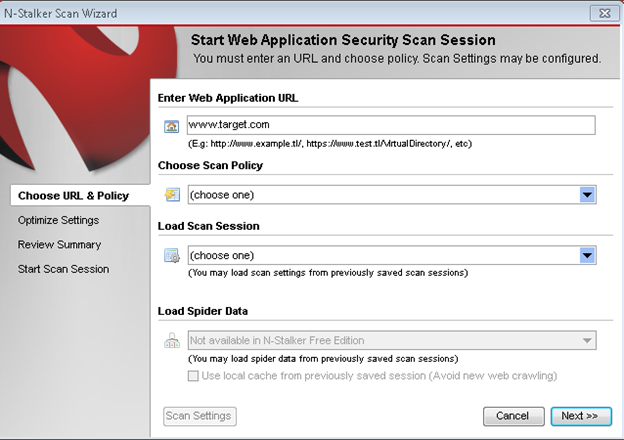
• Full xss evaluation

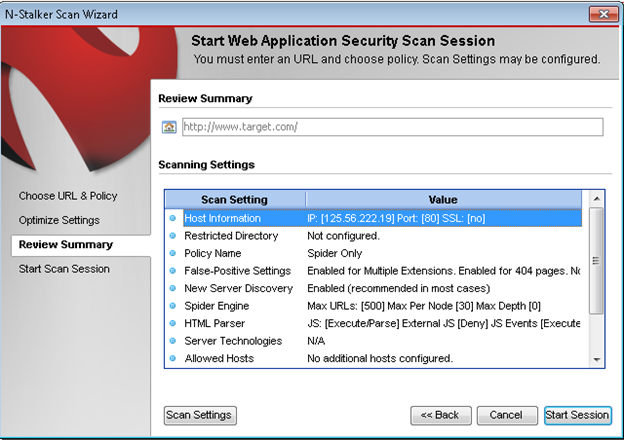
owasp policy

• Analysis of web server infrastructure.

Once you have selected your options, the next step is Optimize Settings. This will crawl the entire website for further analysis.

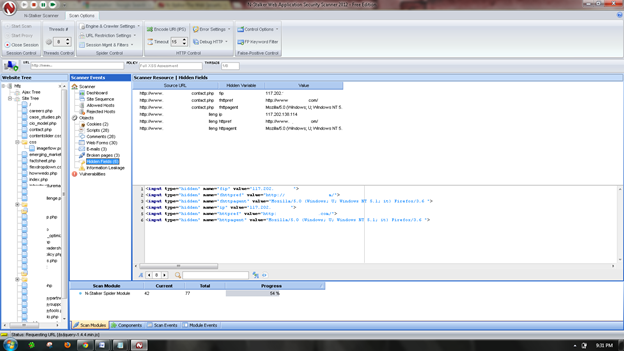
Validate option allows you to get all information like host information, technology used, policy name, etc.



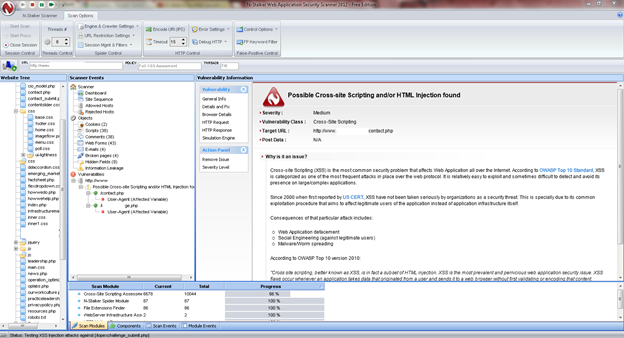


Start the session and the scan after you're finished.

The scanner will browse the entire website and display information about scripts, broken pages, hidden fields, information leaks, and web forms that is useful for further analysis.



When the scan is finished, the NStalker scanner will display information such as the severity level, vulnerability class, the reason why there is a problem, the solution to the problem, and the URL that is vulnerable to the specific vulnerability.



**RESULT:**

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

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

**AIM:**

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

**PROCEDURE:**

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

**TROJAN:**

* A Trojan horse, often known as a trojan, is any software that deceives people about its actual purpose.
* Trojans are often propagated by some type of social engineering, including tricking a user into opening an email attachment that seems innocent (like a normal form to be filled out) or clicking on a phoney ad on social media or elsewhere.
* Many contemporary forms function as a backdoor, contacting a controller who can subsequently get illegal access to the affected machine, despite the fact that their payload can be anything.
* Trojans can provide an attacker access to users' private information, including financial details, passwords, and identities.
* ***Example:*** *Ransomware* attacks are often carried out using a *trojan*.

**CODE:**

***Trojan.bat***

@echo off

:x

start mspaint

start notepad

start cmd

start explorer

start control

start calc

goto x

**OUTPUT:**

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

**RESULT:**

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

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

**AIM:**

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

**ROOTKIT HUNTER:**

• The Unix-based tool rkhunter (Rootkit Hunter) searches for rootkits, backdoors, and potential local exploits.

•It accomplishes this by comparing the SHA-1 hashes of critical files with those of known-good ones in internet databases, looking for rootkit default folders, incorrect permissions, hidden files, suspicious strings in kernel modules, and performing particular tests for Linux and FreeBSD.

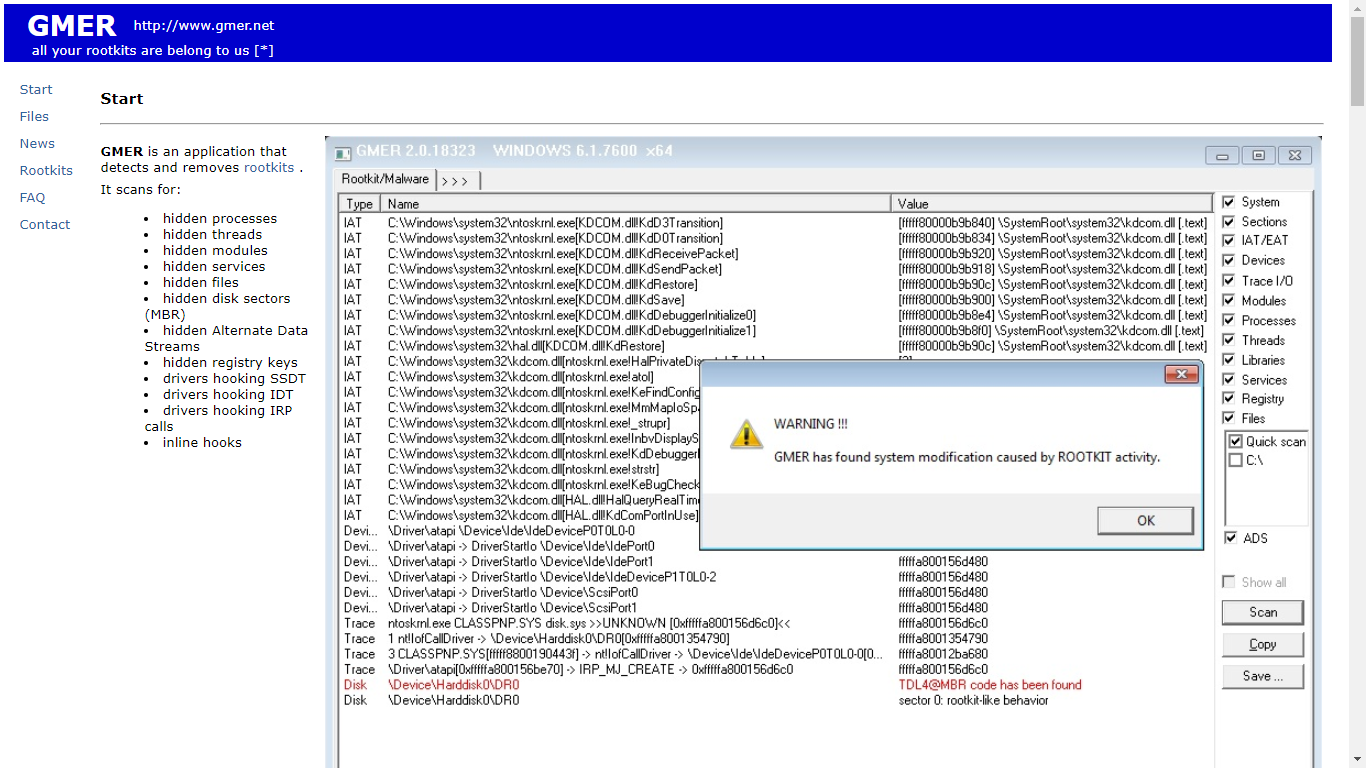
• The availability of rkhunter on well-known operating systems makes it noteworthy (Fedora, Debian, etc.)

• The utility is portable because it was written in Bourne shell. It is compatible with nearly all UNIX-derived platforms.

**GMER ROOTKIT TOOL:**

Przemysaw Gmerek, a Polish researcher, created the rootkit detection and removal software application GMER. It is compatible with Windows NT, 2000, XP, Vista, 7, 8, and 10 and operates on Microsoft Windows. Windows x64 is now fully supported as of version 2.0.18327.

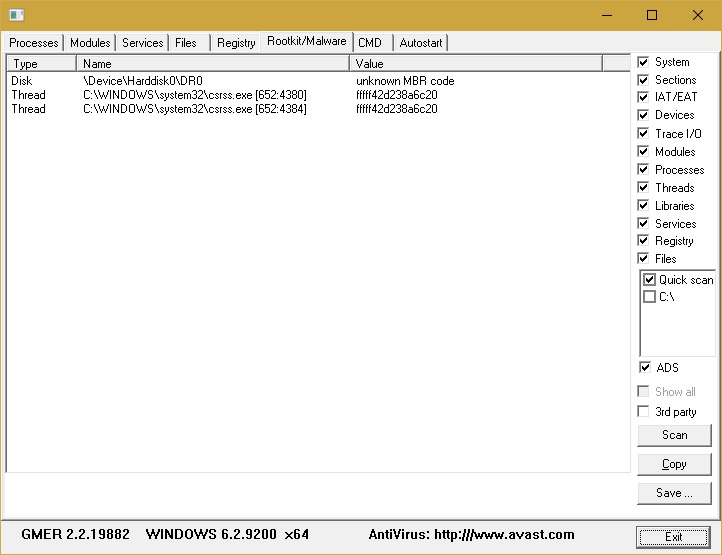
**Step 1**



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

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

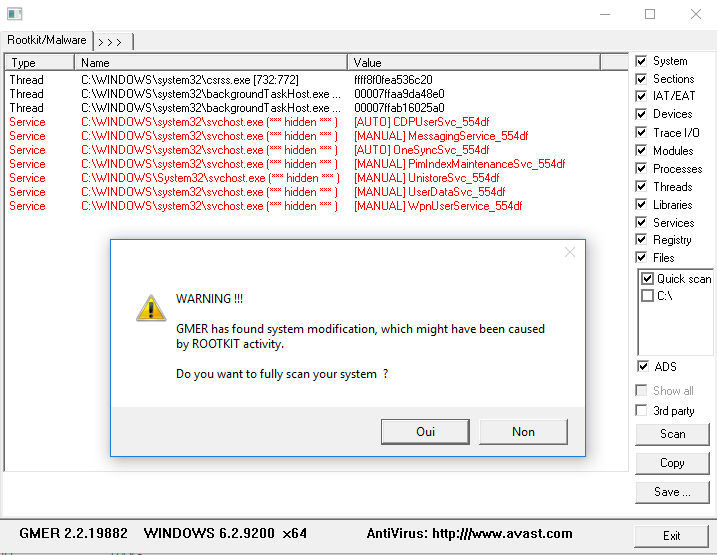
**Step 2**



Double-click the icon for the program.

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

**Step 3**



Select any application or file marked in red when the software has finished scanning your computer. Choose "Delete" from the shortcut menu when using the right click.

It could be protected if the red item is a service. "Disable" may be chosen by right-clicking the service. When that service is found, restart your computer and run the scan again, choosing "Delete" this time.

Close the software and restart your PC after Rootkits have been removed from your machine.

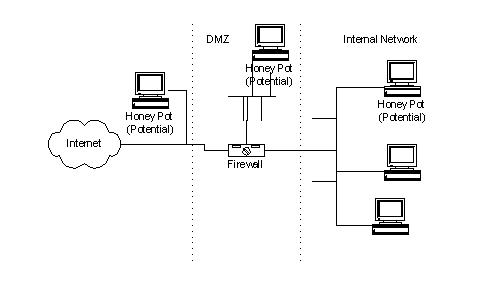
**RESULT:**

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

|  |  |
| --- | --- |
| **Ex. No : 12**  **Date :** | **CONFIGURE A HONEY POT AND MONITOR IT ON INTERNET** |

**HONEY POT**

Decoy servers or systems, such as honey pot systems, are set up to acquire information on an attacker or system intruder. It is crucial to keep in mind that Honey Pots are an additional level or system and do not take the place of other conventional Internet security methods. Although they are most frequently used inside a firewall for control purposes, honey pots can be set up inside, outside, in the DMZ of a firewall architecture, or even in all of these places. They resemble traditional Intruder Detection Systems (IDS) in certain ways, but with a stronger emphasis on information collecting and deception.



Honeypot systems are set up to be easier prey for intruders than real production systems, but the system is modified slightly to allow logging or tracking of intruder activity. It is generally believed that once an intruder breaks into a system, they will come back again and again. During these subsequent visits, additional information may be collected and additional file, security and system access attempts to Honey may be monitored and recorded. In general, there are two common reasons or goals for setting up honeypots.

1. Learn how intruders try to access your system. The general idea is that a record of an intruder's activity is suspicious and can provide insight into attack methods to better protect real production systems.

4. Gather the forensic information needed to apprehend or track down the intruder. This is the type of information law enforcement agencies often require

An official with information necessary for law enforcement.

The general idea when setting up honeypot systems is that it's okay to use lies and deception when dealing with intruders. What this means when setting up a honeypot is that certain goals should be considered. These goals are:

1. The honeypot system should look as generic as possible. If you have a Microsoft NT-based system, it should appear unchanged to a potential intruder. Otherwise, you may be disconnected before much information is collected.

2. You should be careful about the traffic you allow intruders to send back to the Internet because you don't want to be a launching point for attacks on other entities on the Internet. (One reason to install a honeypot behind a firewall!)

3. You should make your honeypot an interesting site by placing "dummy" information or pretending that an intruder has found an "intranet" server. Search your system so that you can gather as much forensic information as possible.

**KFSENSORS**

KFSensor runs on 32-bit or 64-bit versions of Windows XP, Windows 2003 Server, Windows Vista, Windows 2008 Server, Windows 7, Windows 8, or Windows 2012 Server. Other requirements are 2 GB hard disk space, 2 GB RAM, and at least 1.5 GHz processor.

KFSensor emulates vulnerable services to mitigate attacks on your network. These vulnerable services distract and confuse attackers. Attacks detected on emulated services are reported in the event log. The KFSensor GUI is divided into two columns. The left column lists ports and services. A green icon means it is actively monitored. A blue icon means that an error has occurred. You can click on each port on the left to view attacks to that port only. The right column is the KFSensor log. Red and yellow events indicate major events.

Deployment and configuration of KFSensor depends on your security goals and the type of network you are using. KFSensor detects and mitigates attacks and provides detailed event logs for further defense research.

There are several places where KFSensor can be used. KFSensor can be deployed on a device directly connected to the internet, in a DMZ, on a workstation behind a firewall in your internal network, or on a dedicated zone.

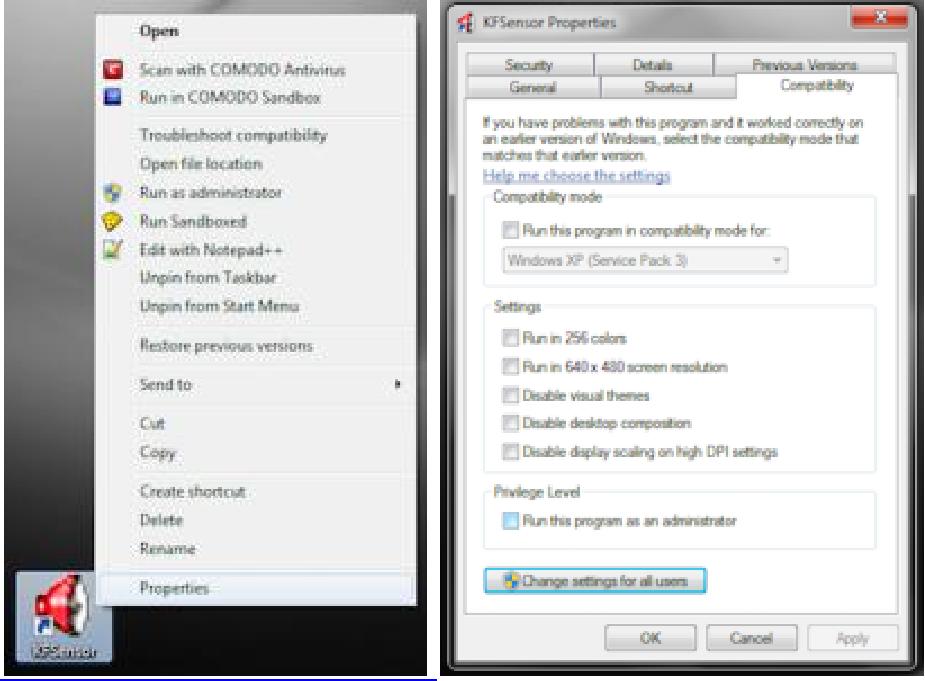
KFSensor can alert events via syslog, audio, system tray, or email. Taskbar alerts are tri-color.

|  |  |  |  |
| --- | --- | --- | --- |
| IconName | | Description |  |
|  |  |  |  |
|  | Banshee | Used for miscellaneous services |  |
|  |  |  |  |
|  | Server | Used for services found on a Windows server, such as Windows Terminal |  |
|  | Server |  |
|  |  |  |
|  |  |  |  |
|  | Workstation | Used for services found on all Windows machines |  |
|  |  |  |  |
|  | World | Used for services that may be exposed to the Internet |  |
|  |  |  |  |
|  | Penguin | Used for services found on Linux systems, but not usually on Windows systems |  |
|  |  |  |  |
|  | Radio activeUsed for non-standard applications such as peer to peer file sharing applications | |  |
|  |  |  |  |
|  | Skull | Used for worms |  |
|  |  |  |  |
|  | Hacker | Used for trojans and root kits |  |
|  |  |  |  |

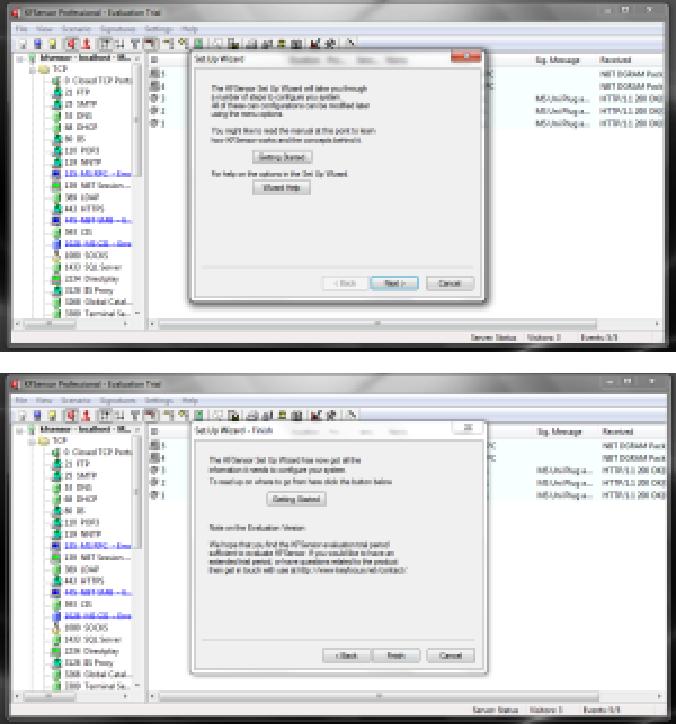


Once you have installed WinPcap and KFSensor set KFSensor to run as Administrator. Right click the icon and select Properties. Under the tab Compatibility check the box, run this

program as an administrator.



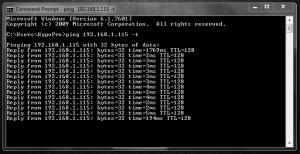
Open KFSensor and proceed with the installation. Follow the Installation Wizard. Once you have finished, restart KFSensor.



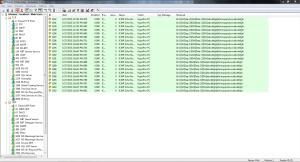
Once installed begin testing your honeypot by launching attacks against the sensors. The first this Im going to do is attempting to login to the machine via FTP with the command: ftp 192.168.1.115. Im going to use the username and password, China:APT



The next thing I’m going to pingdo192is.168.1.115ping-t. the de

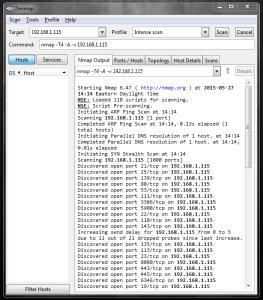


If we look at the event log we can see the flood of ping requests.

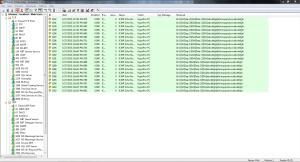


Now lets step up the test with Zenmap. For this example I will scan 192.168.1.115 with nmap -

T4 -A -v 192.168.1.115.



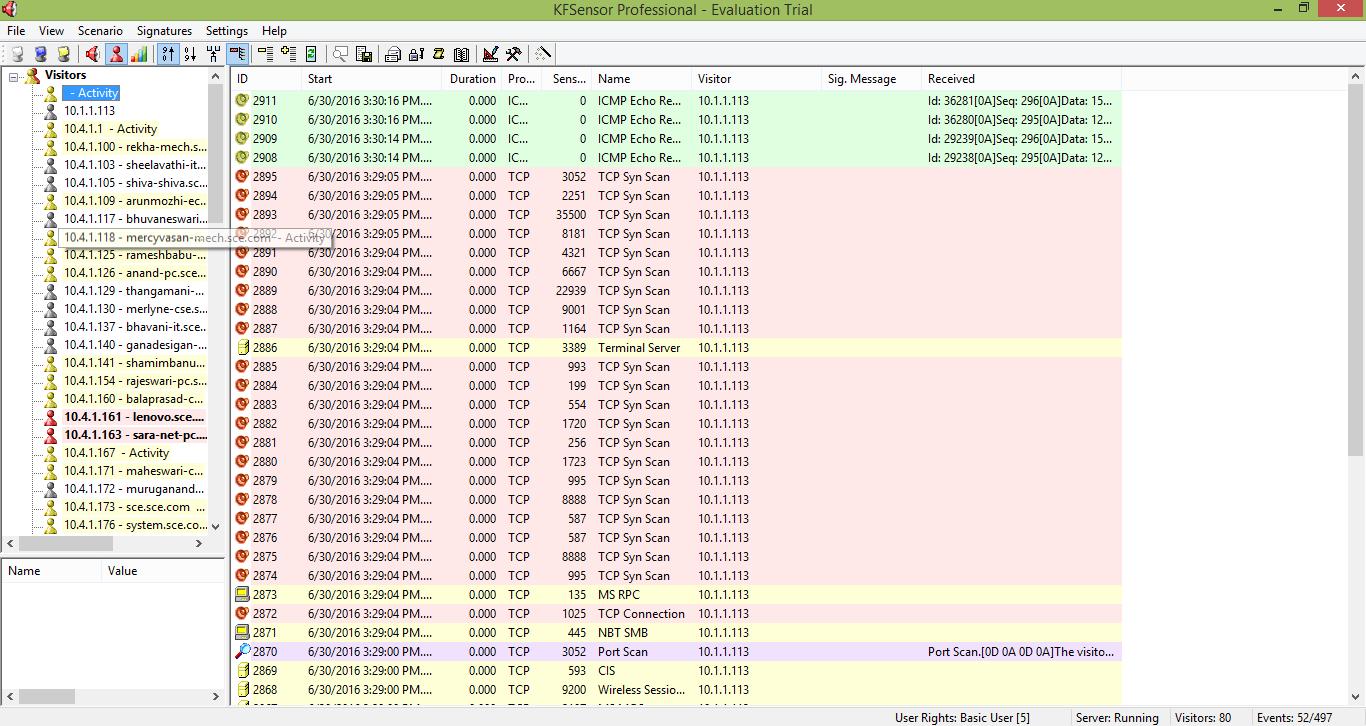
If we look at the event log we can clearly see the results of the Nmap scan.



For the last test we will use LOIC to attack the honeypot with a UPD, TCP and a HTTP flood.



If we look at the event log now we can see that our honeypot is experiencing a DoS attack from the UPD, TCP and HTTP attacks.



Commands to be typed in command prompt:

C:\> netstat -a -b

-a Displays all connections and listening ports.

-b Displays the executable involved in creating each connection or listening port. Well-known executables may host numerous independent components; in these instances, the order in which the components were engaged in setting up the connection or listening port is shown. The executable name in this instance is in [] at the bottom, the component it called is on top, and so on until TCP/IP is reached. If you don't have enough permissions, this option will fail and take a lot of time.

-n Displays addresses and port numbers in numerical form.

-o Displays the owning process ID associated with each connection.

1)C:\> ipconfig 2)C:\> ftp ipaddress

Eg: c:\> ftp 192.168.1.1

3) c:\> ping 192.168.1.2

**Result:**

The KFsensor has been successfully installed and study about the variety of options to monitor the network.