**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 | Perform encryption, decryption using the **Ceasercipher** Substitution Techniques. | **4** |
| 1b | Perform encryption, decryption using the **PlayFaircipher** Substitution Techniques. | **6** |
| 1c | Perform encryption, decryption using the  **Hillcipher** Substitution Techniques. | **13** |
| 1d | Perform encryption, decryption using the **Vigenerecipher** Substitution Techniques. | **16** |
| 2a | Perform encryption, decryption using the **RailFence** transposition Techniques. | **18** |
| 2b | Perform encryption, decryption using the **Row & Column Transformation** transposition Technique. | **21** |
| 3 | Apply **DES** algorithm for practical application. | **24** |
| 4 | Apply **AES** algorithm for practical application | **26** |
| 5 | Implement **RSA** algorithm using HTML & JavaScript. | **29** |
| 6 | Implement the **Diffie-Hellman** key exchange algorithm for a given probem. | **33** |
| 7 | Calculate the message digest of a text using **SHA-1** algorithm. | **35** |
| 8 | Implement the **Signature Scheme**-Digital Signature Standard. | **37** |
| -9 | Demonstrate **intrusion detection system**(ids) using any tool eg. 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 :** | **Encryption and Decryption Using Ceaser Cipher** |

**AIM:**

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

**ALGORITHMS:**

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

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

**PROGRAM:**

***CaesarCipher.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="")

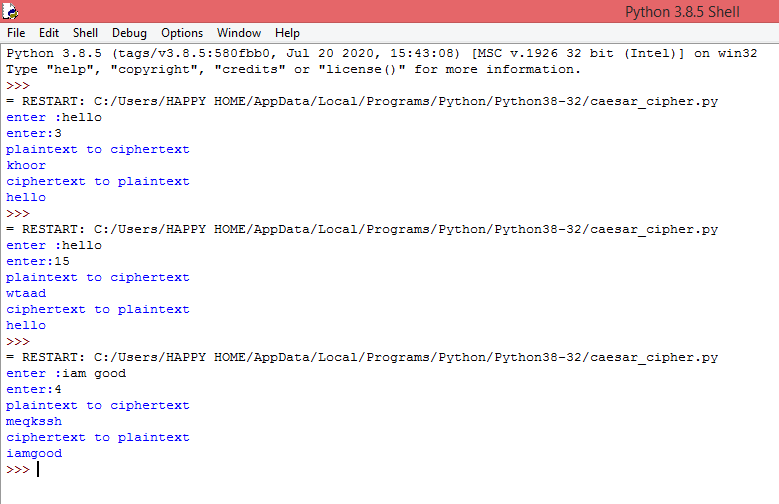
print("\nciphertext to plaintext")

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

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

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

**OUTPUT:**



**RESULT:**

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

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

**AIM:**

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

**ALGORITHM:**

1.To encrypt a message, one would break the message into digrams (groups of 2 letters)

2.For example, "HelloWorld" becomes "HE LL OW OR LD".

3.These digrams will be substituted using the key table.

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

5.The two letters of the digram are considered opposite corners of a rectangle in the key table. To perform the substitution, apply the following 4 rules, in order, to each pair of letters in the plaintext:

**PROGRAM:**

***playfairCipher.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)

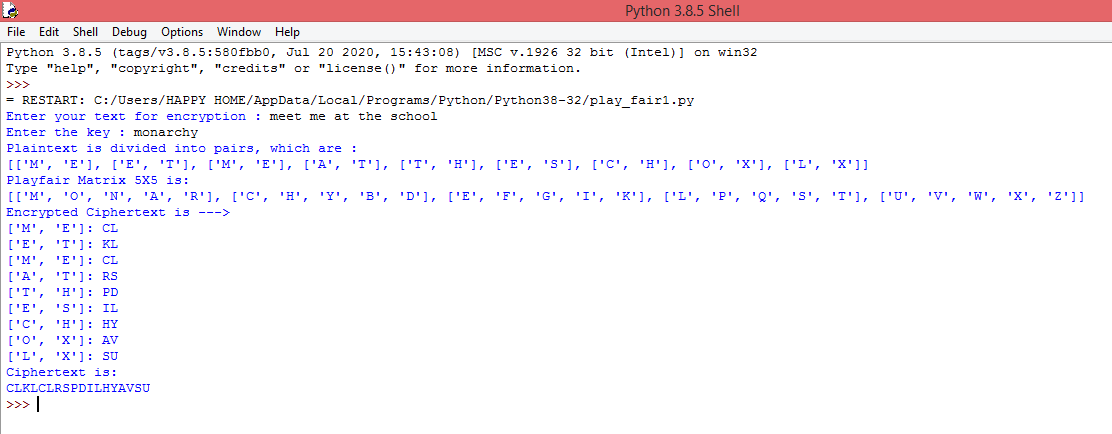
#print combined pair if Ciphertext

print("Ciphertext is:")

for c in cipher\_text\_final :0

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

**OUTPUT:**



**RESULT:**

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

|  |  |
| --- | --- |
| **Ex. No : 1(c)**  **Date :** | **Hill Cipher** |

**AIM:**

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

**ALGORITHM:**

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

**PROGRAM:**

***HillCipher.java***

class hillCipher {

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

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

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

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

public static String key = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

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

String ret = "";

int x, y, z;

int posa = (int) a - 65;

int posb = (int) b - 65;

int posc = (int) c - 65;

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

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

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

a = key.charAt(x % 26);

b = key.charAt(y % 26);

c = key.charAt(z % 26);

ret = "" + a + b + c;

return ret;

}

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

String ret = "";

int x, y, z;

int posa = (int) a - 65;

int posb = (int) b - 65;

int posc = (int) c - 65;

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

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

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

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

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

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

ret = "" + a + b + c;

return ret;

}

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

String msg;

String enc = "";

String dec = "";

int n;

msg = ("SecurityLaboratory");

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

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

msg = msg.toUpperCase();

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

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

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

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

msg += 'X';

}

}

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

char[] pdchars = msg.toCharArray();

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

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

}

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

char[] dechars = enc.toCharArray();

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

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

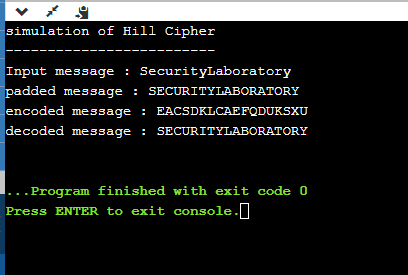
}

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

}

}

**OUTPUT:**



**RESULT**:

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

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

**AIM:**

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

**ALGORITHM:**

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

**PROGRAM:**

***vigenereCipher.java***

public class vigenereCipher {

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

String res = "";

text = text.toUpperCase();

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

char c = text.charAt(i);

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

continue;

}

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

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

}

return res;

}

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

String res = "";

text = text.toUpperCase();

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

char c = text.charAt(i);

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

continue;

}

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

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

}

return res;

}

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

String key = "VIGENERECIPHER";

String msg = "SecurityLaboratory";

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

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

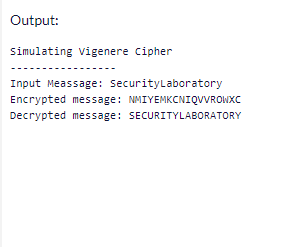
String enc = encode(msg, key);

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

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

}}

**OUTPUT:**



**RESULT:**

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

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

**AIM:**

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

**ALGORITHM**:

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

**PROGRAM:**

***railFenceCipher.java***

class railfenceCipherHelper {

int depth;

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

int r = depth;

int l = msg.length();

int c = l / depth;

int k = 0;

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

String enc = "";

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

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

if (k != l) {

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

} else {

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

}

}

}

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

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

enc += mat[i][j];

}

}

return enc;

}

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

int r = depth;

int l = encmsg.length();

int c = l / depth;

int k = 0;

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

String dec = "";

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

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

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

}

}

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

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

dec += mat[j][i];

}

}

return dec;

}

}

class railFenceCipher {

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

railfenceCipherHelper rf = new railfenceCipherHelper();

String msg, enc, dec;

msg = "Anna University, Chennai";

int depth = 2;

enc = rf.encode(msg, depth);

dec = rf.decode(enc, depth);

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

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

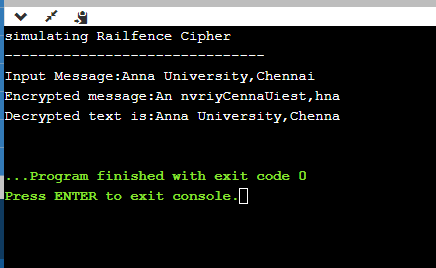
System.out.println("Encrypted Message : " + enc);

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

}

}

**OUTPUT:**



**RESULT:**

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

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

**AIM:**

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

**ALGORITHM:**

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

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

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

**PROGRAM:**

***TransCipher.java***

import java.util.\*;

class TransCipher {

public static void main(String args[]) {

Scanner sc = new Scanner(System.in);

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

String pl = sc.nextLine();

sc.close();

String s = "";

int start = 0;

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

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

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

start = i + 1;

}

}

s = s + pl.substring(start);

System.out.print(s);

System.out.println();

// end of space deletion

int k = s.length();

int l = 0;

int col = 4;

int row = s.length() / col;

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

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

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

if (l < k) {

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

l++;

} else {

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

}

}

}

// arranged in matrix

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

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

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

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

}

}

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

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

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

}

}

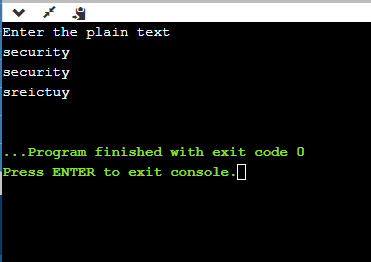
// display

System.out.println();

}

}

**OUTPUT:**



**RESULT:**

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

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

**AIM:**

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

**ALGORITHM:**

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

**PROGRAM:**

***DES.java***

import java.security.InvalidKeyException;

import java.security.NoSuchAlgorithmException;

import javax.crypto.BadPaddingException;

import javax.crypto.Cipher;

import javax.crypto.IllegalBlockSizeException;

import javax.crypto.KeyGenerator;

import javax.crypto.NoSuchPaddingException;

import javax.crypto.SecretKey;

public class DES

{

public static void main(String[] argv) {

try{

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

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

SecretKey myDesKey = keygenerator.generateKey();

Cipher desCipher;

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

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

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

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

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

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

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

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

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

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

}catch(NoSuchAlgorithmException e){

e.printStackTrace();

}catch(NoSuchPaddingException e){

e.printStackTrace();

}catch(InvalidKeyException e){

e.printStackTrace();

}catch(IllegalBlockSizeException e){

e.printStackTrace();

}catch(BadPaddingException e){

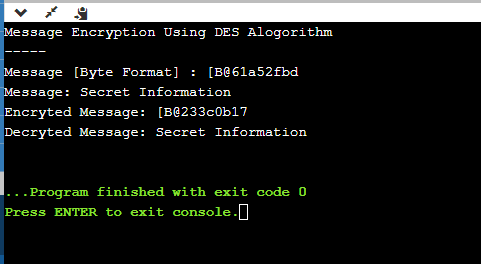
e.printStackTrace();

}

}

}

**OUTPUT:**



**RESULT:**

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

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

**AIM:**

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

**ALGORITHM:**

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

**PROGRAM:**

***AES.java***

import java.io.UnsupportedEncodingException;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

import java.util.Arrays;

import java.util.Base64;

import javax.crypto.Cipher;

import javax.crypto.spec.SecretKeySpec;

public class AES {

private static SecretKeySpec secretKey;

private static byte[] key;

public static void setKey(String myKey) {

MessageDigest sha = null;

try {

key = myKey.getBytes("UTF-8");

sha = MessageDigest.getInstance("SHA-1");

key = sha.digest(key);

key = Arrays.copyOf(key, 16);

secretKey = new SecretKeySpec(key, "AES");

} catch (NoSuchAlgorithmException e) {

e.printStackTrace();

} catch (UnsupportedEncodingException e) {

e.printStackTrace();

}

}

public static String encrypt(String strToEncrypt, String secret) {

try {

setKey(secret);

Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");

cipher.init(Cipher.ENCRYPT\_MODE, secretKey);

return Base64.getEncoder().encodeToString(cipher.doFinal(strToEncrypt.getBytes("UTF-8")));

} catch (Exception e) {

System.out.println("Error while encrypting: " + e.toString());

}

return null;

}

public static String decrypt(String strToDecrypt, String secret) {

try {

setKey(secret);

Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5PADDING");

cipher.init(Cipher.DECRYPT\_MODE, secretKey);

return new String(cipher.doFinal(Base64.getDecoder().decode(strToDecrypt)));

} catch (Exception e) {

System.out.println("Error while decrypting: " + e.toString());

}

return null;

}

public static void main(String[] args) {

final String secretKey = "annaUniversity";

String originalString = "www.annauniv.edu";

String encryptedString = AES.encrypt(originalString, secretKey);

String decryptedString = AES.decrypt(encryptedString, secretKey);

System.out.println("URL Encryption Using AES Algorithm\n------------");

System.out.println("Original URL : " + originalString);

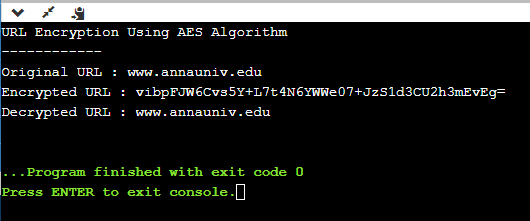
System.out.println("Encrypted URL : " + encryptedString);

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

}

}

**OUTPUT:**

****

**RESULT:**

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

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

**AIM:**

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

**ALGORITHM:**

1. Choose two prime number p and q
2. Compute the value of n and **p**
3. Find the value of ***e*** (public key)
4. Compute the value of ***d*** (private key) using gcd()
5. Do the encryption and decryption
   1. Encryption is given as,

***c = te mod n***

* 1. 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;

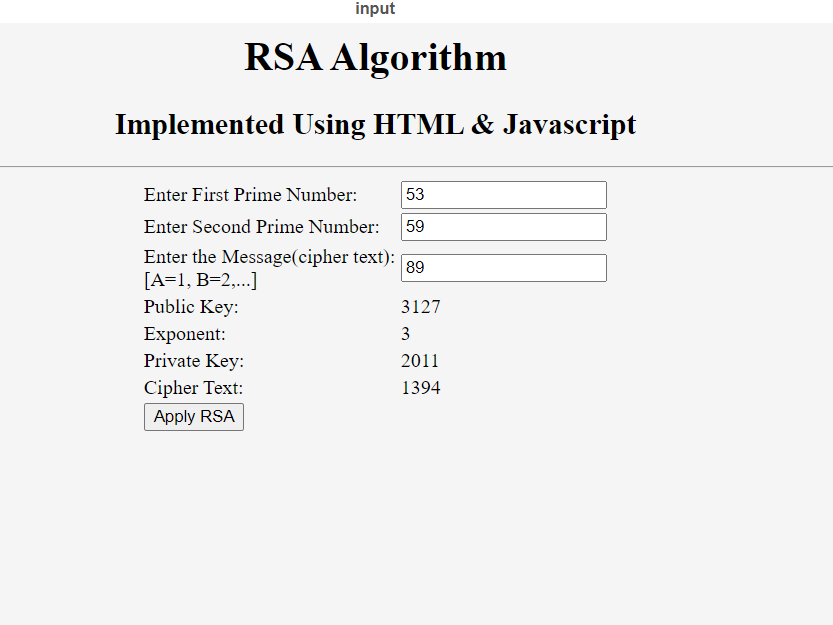
document.getElementById('ciphertext').innerHTML = ct;

}

</script>

</html>

**OUTPUT:**



**RESULT:**

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

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

**AIM:**

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

**ALGORITHM:**

1. [Alice and Bob](https://en.wikipedia.org/wiki/Alice_and_Bob) publicly agree to use a modulus *p* = 23 and base *g* = 5 (which is a primitive root modulo 23).
2. Alice chooses a secret integer ***a*** = 4, then sends Bob *A* = *g****a*** mod *p*
   * *A* = 5**4** mod 23 = 4
3. Bob chooses a secret integer ***b*** = 3, then sends Alice *B* = *g****b*** mod *p*
   * *B* = 5**3** mod 23 = 10
4. Alice computes ***s*** = *B****a*** mod *p*
   * ***s*** = 10**4** mod 23 = 18
5. Bob computes ***s*** = *A****b*** mod *p*
   * ***s*** = 4**3** mod 23 = 18
6. Alice and Bob now share a secret (the number 18).

**PROGRAM:**

***DiffieHellman.java***

class DiffieHellman {

public static void main(String args[]) {

int p = 23; /\* publicly known (prime number) \*/

int g = 5; /\* publicly known (primitive root) \*/

int x = 4; /\* only Alice knows this secret \*/

int y = 3; /\* only Bob knows this secret \*/

double aliceSends = (Math.pow(g, x)) % p;

double bobComputes = (Math.pow(aliceSends, y)) % p;

double bobSends = (Math.pow(g, y)) % p;

double aliceComputes = (Math.pow(bobSends, x)) % p;

double sharedSecret = (Math.pow(g, (x \* y))) % p;

System.out.println("simulation of Diffie-Hellman key exchange algorithm\n---------------------------------------------");

System.out.println("Alice Sends : " + aliceSends);

System.out.println("Bob Computes : " + bobComputes);

System.out.println("Bob Sends : " + bobSends);

System.out.println("Alice Computes : " + aliceComputes);

System.out.println("Shared Secret : " + sharedSecret);

/\* shared secrets should match and equality is transitive \*/

if ((aliceComputes == sharedSecret) && (aliceComputes == bobComputes))

System.out.println("Success: Shared Secrets Matches! " + sharedSecret);

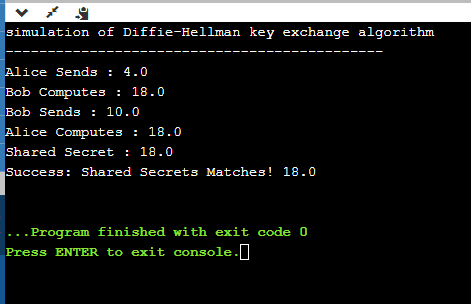
else

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

}

}

**OUTPUT:**

****

**RESULT:**

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

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

**AIM:**

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

**ALGORITHM:**

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

**PROGRAM:**

***sha1.java***

import java.security.\*;

public class sha1 {

public static void main(String[] a) {

try {

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

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

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

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

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

String input = "";

md.update(input.getBytes());

byte[] output = md.digest();

System.out.println();

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

input = "abc";

md.update(input.getBytes());

output = md.digest();

System.out.println();

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

input = "abcdefghijklmnopqrstuvwxyz";

md.update(input.getBytes());

output = md.digest();

System.out.println();

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

System.out.println();

} catch (Exception e) {

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

}

}

private static String bytesToHex(byte[] b) {

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

StringBuffer buf = new StringBuffer();

for (byte aB : b) {

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

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

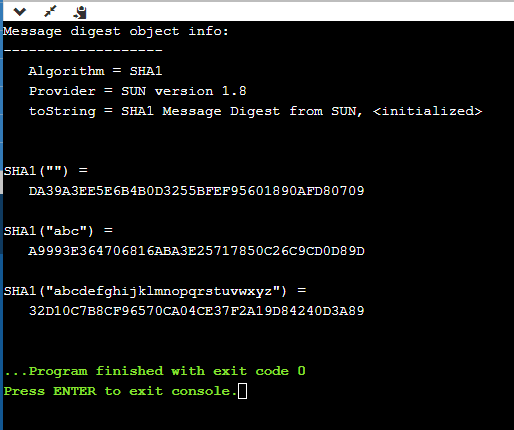
}

return buf.toString();

}

}

**OUTPUT:**



**RESULT:**

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

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

**AIM:**

To implement the SIGNATURE SCHEME - Digital Signature Standard.

**ALGORITHM:**

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

**PROGRAM:**

import java.security.KeyPair;

import java.security.KeyPairGenerator;

import java.security.PrivateKey;

import java.security.Signature;

import java.util.Scanner;

public class CreatingDigitalSignature {

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

Scanner sc = new Scanner(System.in);

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

String msg = sc.nextLine();

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

keyPairGen.initialize(2048);

KeyPair pair = keyPairGen.generateKeyPair();

PrivateKey privKey = pair.getPrivate();

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

sign.initSign(privKey);

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

sign.update(bytes);

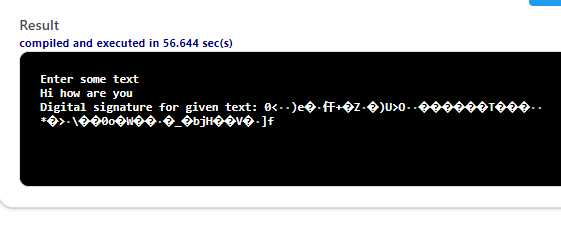
byte[] signature = sign.sign();

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

}

}

**OUTPUT:**



**RESULT:**

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

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

**AIM:**

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

**STEPS ON CONFIGURING AND INTRUSION DETECTION:**

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



Finding an interface

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

9. To run snort in IDS mode, you will need to configure the file “snort.conf” according to your network environment.

10. To specify the network address that you want to protect in snort.conf file, look for the following line.

var HOME\_NET 192.168.1.0/24 (You will normally see any here)

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

Example:

example snort

12. Change the RULE\_PATH variable to the path of rules folder.

var RULE\_PATH c:\snort\rules

path to rules

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

C:\Snort\lib\snort\_dynamiccpreprocessor

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

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

Example:

dynamicengine C:\Snort\lib\snort\_dynamicengine\sf\_engine.dll

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

include c:\snort\etc\classification.config

include c:\snort\etc\reference.config

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

include $RULE\_PATH/icmp.rules

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

include $RULE\_PATH/icmp-info.rules

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

output alert\_fast: snort-alerts.ids

19. Comment (add a #) the whitelist $WHITE\_LIST\_PATH/white\_list.rules and the blacklist

Change the nested\_ip inner , \ to nested\_ip inner #, \

20. Comment out (#) following lines:

#preprocessor normalize\_ip4

#preprocessor normalize\_tcp: ips ecn stream

#preprocessor normalize\_icmp4

#preprocessor normalize\_ip6

#preprocessor normalize\_icmp6

21. Save the “snort.conf” file.

22. To start snort in IDS mode, run the following command:

snort -c c:\snort\etc\snort.conf -l c:\snort\log -i 3

(Note: 3 is used for my interface card)

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

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

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

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

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

Snort monitoring traffic –



**RESULT:**

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

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

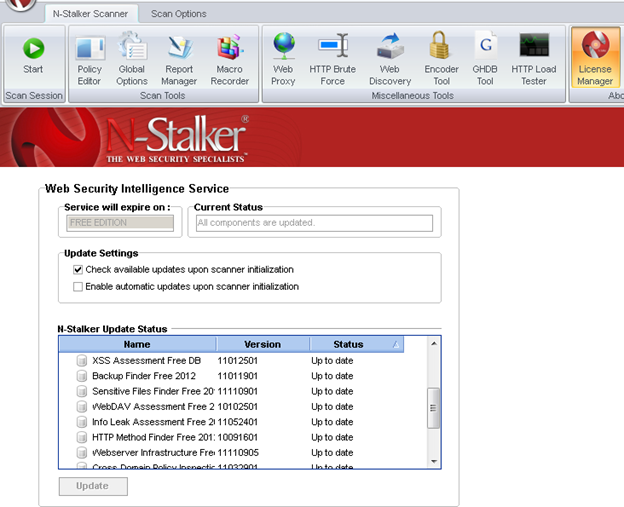
**AIM:**

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

**EXPLORING N-STALKER:**

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

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



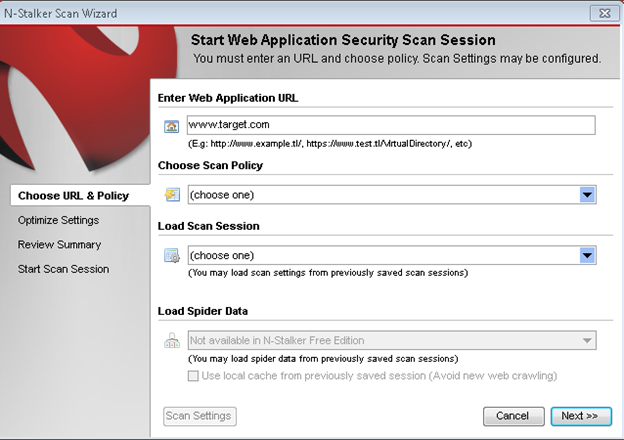
Now goto “Scan Session”, enter the target URL.

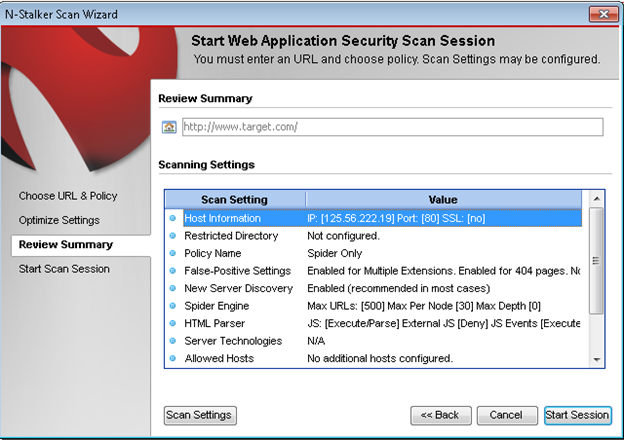
In scan policy, you can select from the four options,

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

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

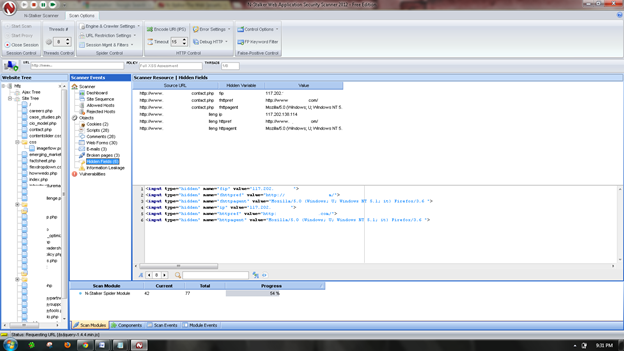
In review option, you can get all the information like host information, technologies used, policy name, etc.



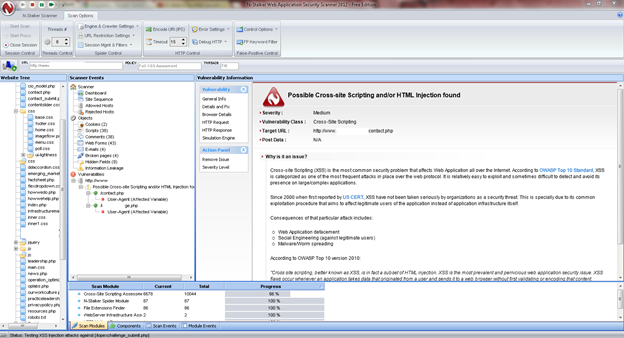


Once done, start the session and start the scan.

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



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



**RESULT:**

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

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

**AIM:**

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

**PROCEDURE:**

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

**TROJAN:**

* In computing, a Trojan horse,or trojan, is any malware which misleads users of its true intent.
* Trojans are generally spread by some form of social engineering, for example where a user is duped into executing an email attachment disguised to appear not suspicious, (e.g., a routine form to be filled in), or by clicking on some fake advertisement on social media or anywhere else.
* Although their payload can be anything, many modern forms act as a backdoor, contacting a controller which can then have unauthorized access to the affected computer.
* Trojans may allow an attacker to access users' personal information such as banking information, passwords, or personal identity.
* ***Example:*** *Ransomware* attacks are often carried out using a *trojan*.

**CODE:**

***Trojan.bat***

@echo off

:x

start mspaint

start notepad

start cmd

start explorer

start control

start calc

goto x

**OUTPUT:**

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

**RESULT:**

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

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

**AIM:**

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

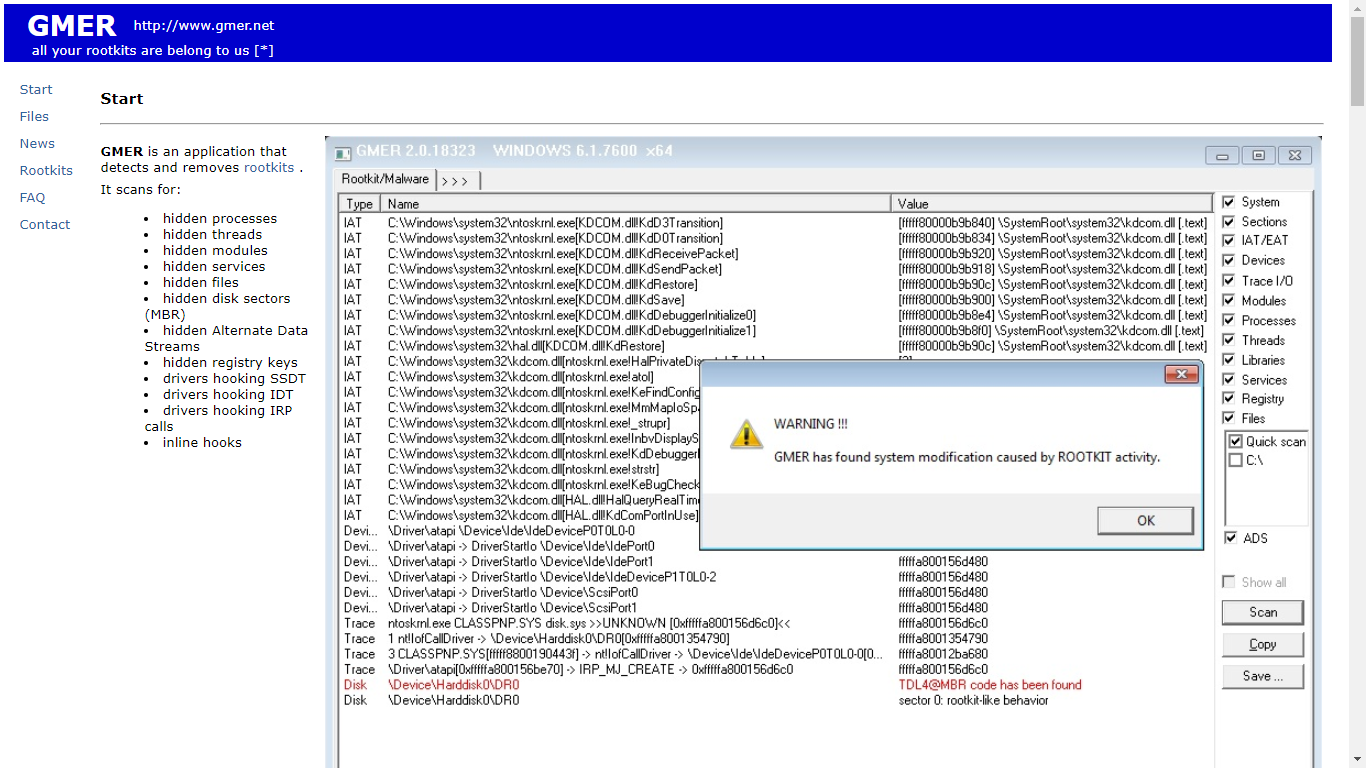
**ROOTKIT HUNTER:**

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

**GMER ROOTKIT TOOL:**

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

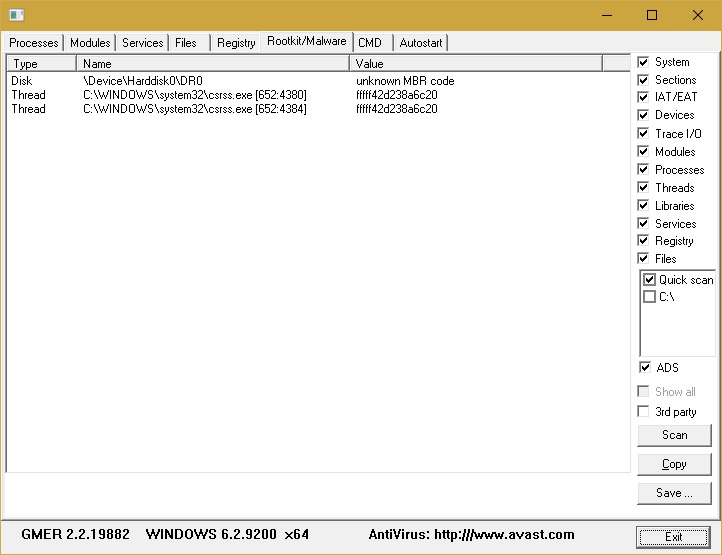
**Step 1**



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

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

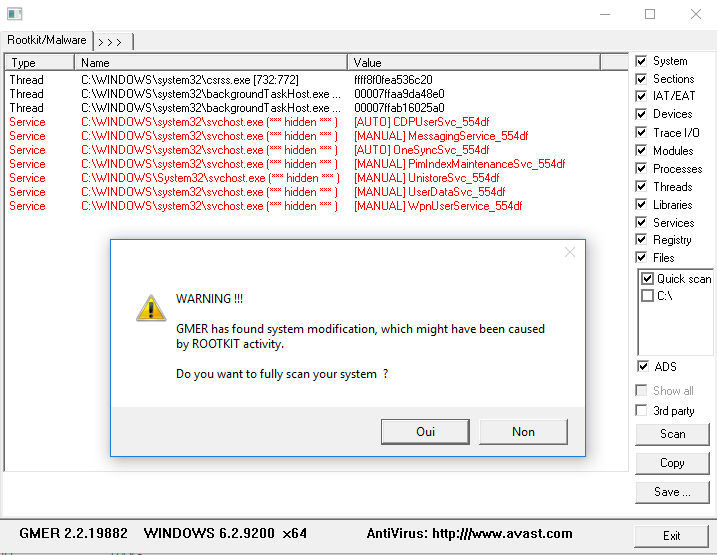
**Step 2**



Double-click the icon for the program.

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

**Step 3**



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

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

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

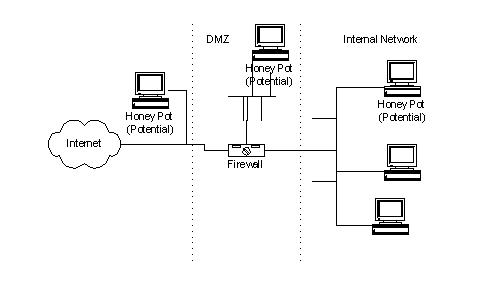
**RESULT:**

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

|  |  |
| --- | --- |
| **Ex. No : 12**  **Date :** | **SETUP A HONEY POT AND MONITOR THE HONEYPOT ON NETWORK** |

**HONEY POT**

Honey Pot Systems are decoy servers or systems setup to gather information regarding an attacker or intruder into your system. It is important to remember that Honey Pots do not replace other traditional Internet security systems; they are an additional level or system. Honey Pots can be setup inside, outside or in the DMZ of a firewall design or even in all of the locations although they are most often deployed inside of a firewall for control purposes. In a sense, they are variants of standard Intruder Detection Systems (IDS) but with more of a focus on information gathering and deception.



A Honey Pot system is setup to be easier prey for intruders than true production systems but with minor system modifications so that their activity can be logged of traced. The general thought is that once an intruder breaks into a system, they will come back for subsequent visits. During these subsequent visits, additional information can be gathered and additional attempts at file, security and system access on the Honey can be monitored and saved. Generally, there are two popular reasons or goals behind setting up a Honey Pot:

1. Learn how intruders probe and attempt to gain access to your systems. The general idea is that since a record of the intruder’sactivitiesiskept, you can gain insight into attack methodologies to better protect your real production systems.

4.Gather forensic information required to aid in the apprehension or prosecution of intruders. This is the sort of information often needed to provide law enforcement

officials with the details needed to prosecute.

The common line of thought in setting up Honey Pot systems is that it is acceptable to use lies or deception when dealing with intruders. What this means to you when setting up a Honey Pot is that certain goal have to be considered.

Those goals are:

1. The Honey Pot system should appear as generic as possible. If you are deploying a Microsoft NT based system, it should appear to the potential intruder that the system has not been modified or they may disconnect before much information is collected.
2. You need to be careful in what traffic you allow the intruder to send back out to the Internet for you donât want to become a launch point for attacks against other entities on the Internet. (One of the reasons for installing a Honey Pot inside of the firewall!)
3. You will want to make your Honey Pot an interesting site by placing "Dummy" information or make it appear as though the intruder has found an "Intranet" server, etc. Expect to spend some time making your Honey Pot appear legitimate so that intruders will spend enough time investigating and perusing the system so that you are able to gather as much forensic information as possible.

**KFSENSORS**

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

KFSensor emulates vulnerable services to help mitigate attacks on your network. These vulnerable services distract and confuse the attacker. The detected attacks on the emulated services are reported in the event log.

The GUI of KFSensor is split between two columns. The column on the left lists the ports and services. The green icons mean that they are being actively monitored. Blue icons means there has been an error. You can click on the individual ports on the left to display attack only on that port. The column on the right is the KFSensor log. Red and Yellow events indicate a serious event.

Deploying and configuring KFSensor depends on what your security goals are and what type of network you are using. KFSensor will detect and mitigate attacks while providing detailed event logs for further defense research.

There are a few places that you can deploy KFSensor. You can deploy KFSensor on a device that is connected directly to the Internet, inside a DMZ, on a workstation behind a firewall on the internal network or in a dedicated zone.

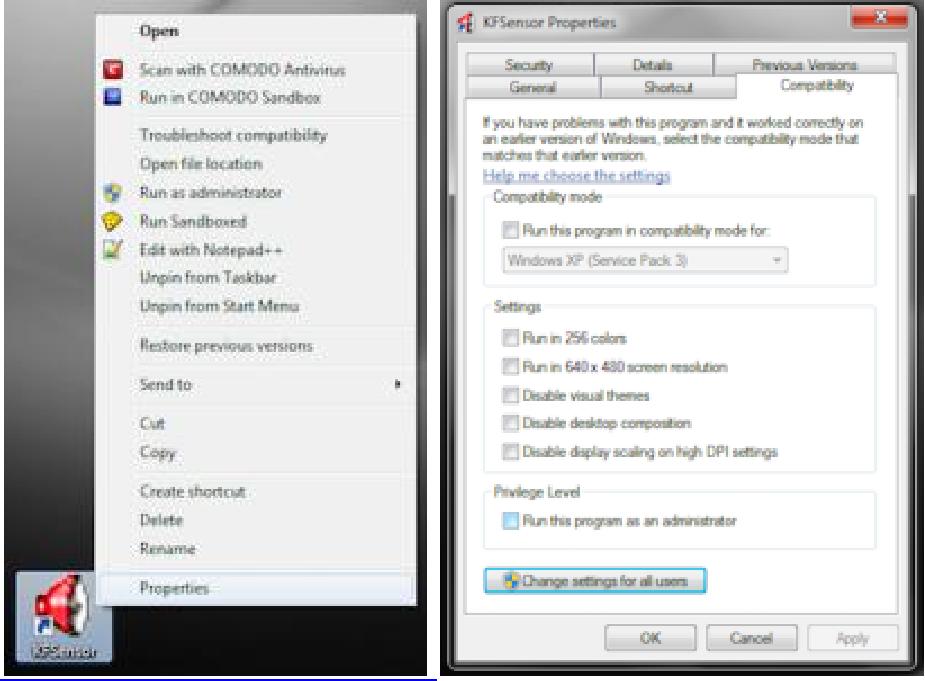
KFSensor can alert you to an events through syslog, audio, system tray, or through email. The system tray alerts are three colors:

|  |  |  |  |
| --- | --- | --- | --- |
| 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 |  |
|  |  |  |  |

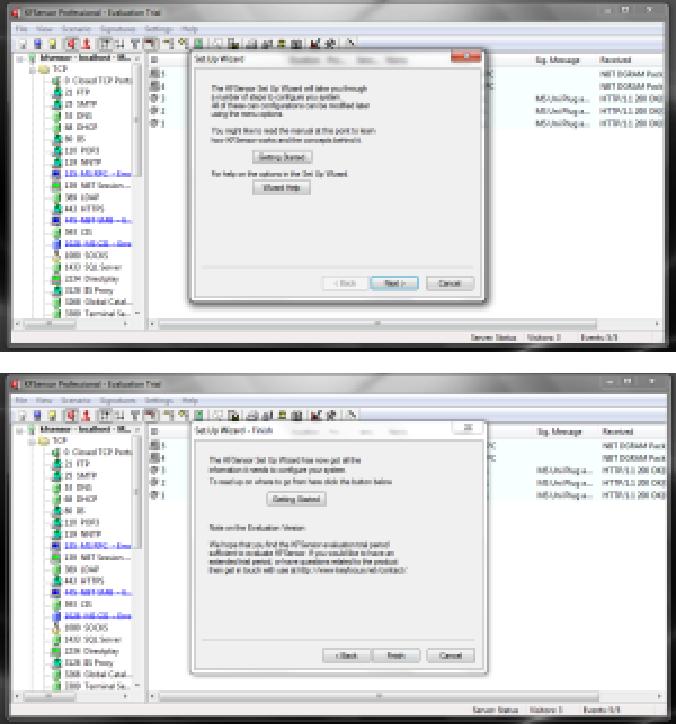


Once you have installed WinPcap and KFSensor set KFSensor to run as Administrator. Right click the icon and select Properties. Under the tab Compatibility check the box, run this

program as an administrator.



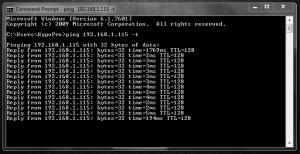
Open KFSensor and proceed with the installation. Follow the Installation Wizard. Once you have finished, restart KFSensor.



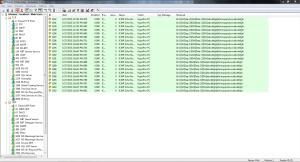
Once installed begin testing your honeypot by launching attacks against the sensors. The first this Im going to do is attempting to login to the machine via FTP with the command: ftp 192.168.1.115. Im going to use the username and password, China:APT



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

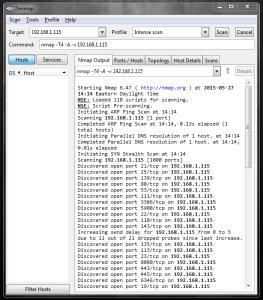


If we look at the event log we can see the flood of ping requests.

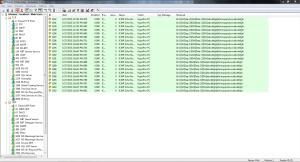


Now lets step up the test with Zenmap. For this example I will scan 192.168.1.115 with nmap -

T4 -A -v 192.168.1.115.



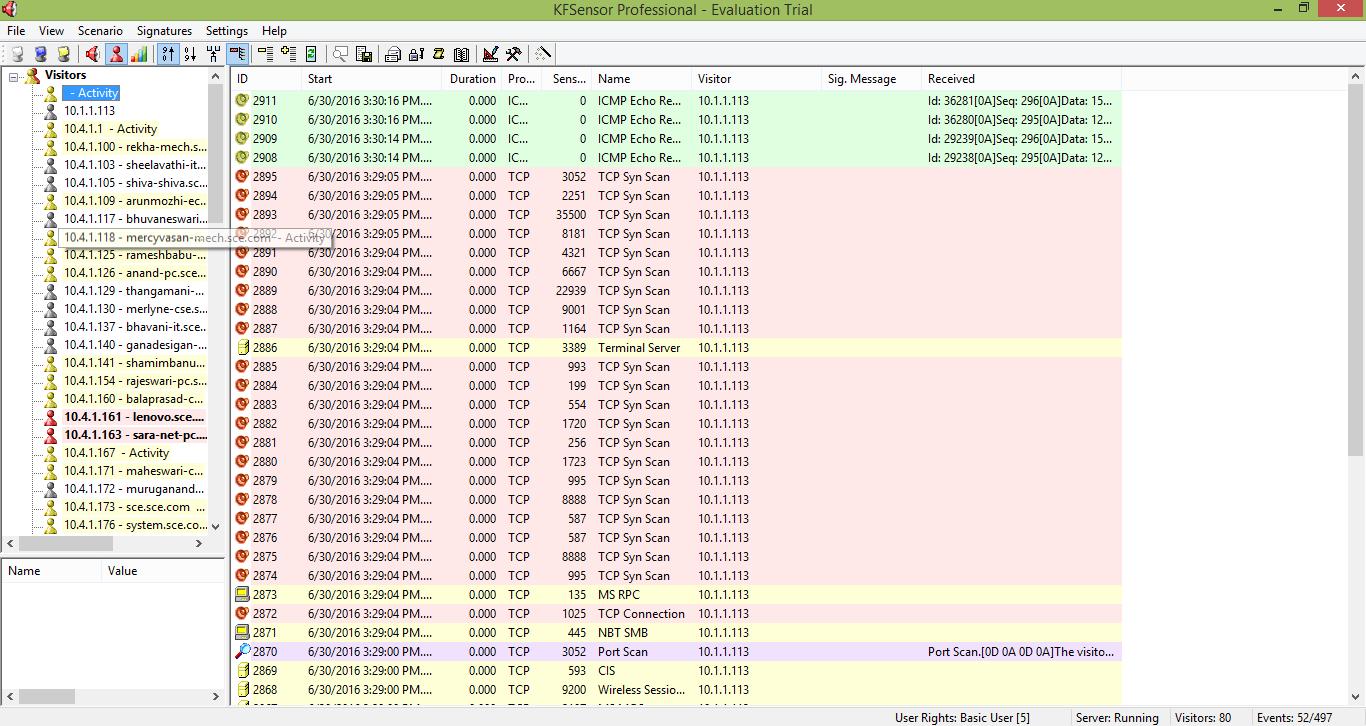
If we look at the event log we can clearly see the results of the Nmap scan.



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. In some cases well-known executables host multiple independent components, and in these cases the sequence of components involved in creating the connection or listening port is displayed. In this case the executable name is in [] at the bottom, on top is the component it called, and so forth until TCP/IP was reached. Note that this option can be time-consuming and will fail unless you have sufficient permissions.

-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.