

EX.NO: 1a**Encryption and Decryption Using Ceaser Cipher****DATE :****AIM:**

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

ALGORITHMS:

1. In Ceaser Cipher each letter in the plaintext is replaced by a letter some fixed number 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,
$$\text{En}(x) = (x + n) \bmod 26$$
5. Decryption is performed similarly, $\text{Dn}(x) = (x - n) \bmod 26$

PROGRAM:

```
import java.util.Scanner;

class Main {

    public static String encode(String enc, int key) {
        key = key % 26 + 26;
        StringBuilder encoded = new StringBuilder();
        for (char i : enc.toCharArray()) {
            if (Character.isLetter(i)) {
                if (Character.isUpperCase(i)) {
                    encoded.append((char) ('A' + (i - 'A' + key) % 26));
                } else {
                    encoded.append((char) ('a' + (i - 'a' + key) % 26));
                }
            } else {
                encoded.append(i);
            }
        }
    }
}
```

```

    }}
    return encoded.toString();
}
public static String decode(String enc, int key) {
    return encode(enc, 26 - key);
}
public static void main(String[] args) throws java.lang.Exception {
    Scanner scanner = new Scanner(System.in);
    System.out.println("Simulating Caesar Cipher\n-----");
    System.out.print("Enter the message: ");
    String msg = scanner.nextLine();
    System.out.print("Enter the key: ");
    int key = scanner.nextInt();
    System.out.println("Input : " + msg);
    String encryptedMessage = Main.encode(msg, key);
    System.out.println("Encrypted Message : " + encryptedMessage);
    String decryptedMessage = Main.decode(encryptedMessage, key);
    System.out.println("Decrypted Message : " + decryptedMessage);
    scanner.close();
}}

```

OUTPUT:

Enter the message: Anna University

Enter the key: 3

Input: Anna University

Encrypted Message: Dqqd Xqlyhuvlwb

Decrypted Message: Anna University

RESULT:

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

EX.NO: 1b Encryption and Decryption Play Fair Cipher**DATE :****AIM:**

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

ALGORITHM:

1. To encrypt a message, one would break the message into diagrams(group of 2 letters)
2. For example, "Hello World" becomes "HE LL OW OR LD".
3. These diagrams 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 diagram.
5. The two letters of the diagram 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:

```
import java. util.*;

public class Main {

    public static void main(String[] args) {

        Scanner s = new Scanner(System.in);

        System.out.println("Enter the key :");

        String key = s.next().toUpperCase();

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

        String plain = s.next().toUpperCase().replace("J", "I");

        char[][] mat = new char[5][5];

        Set<Character> usedChars = new HashSet<>();

        int idx = 0;

        for (char c: key.toCharArray()) {

            if (c != 'J' && !usedChars.contains(c)) {

                usedChars.add(c);

                mat[idx / 5][idx % 5] = c;
```

```

        idx++;
    }
}

for (char c = 'A'; c <= 'Z'; c++) {
    if (c != 'J' && !usedChars.contains(c)) {
        usedChars.add(c);
        mat[idx / 5][idx % 5] = c;
        idx++;
    }
}

for (int i = 0; i < 5; i++) {
    for (int j = 0; j < 5; j++) {
        System.out.print(mat[i][j] + " ");
    }
    System.out.println();
}

StringBuilder pt = new StringBuilder();
for (int i = 0; i < plain.length(); i += 2) {
    char first = plain.charAt(i);
    char second = (i + 1 < plain.length()) ? plain.charAt(i + 1) : 'X';
    if (first == second) {
        second = 'X';
        i--;
    }
    pt.append(first).append(second).append(" ");
}

System.out.println("Prepared text: " + pt.toString());

StringBuilder encrypt = new StringBuilder();
for (String pair : pt.toString().split(" ")) {
    char[] letter = pair.toCharArray();

```

```

int[] pos1 = new int[2];
int[] pos2 = new int[2];
for (int i = 0; i < 5; i++) {
    for (int j = 0; j < 5; j++) {
        if (mat[i][j] == letter[0]) {
            pos1[0] = i;
            pos1[1] = j;
        }
        if (mat[i][j] == letter[1]) {
            pos2[0] = i;
            pos2[1] = j;
        }
    }
}

if (pos1[0] == pos2[0]) { // Same row
    letter[0] = mat[pos1[0]][(pos1[1] + 1) % 5];
    letter[1] = mat[pos2[0]][(pos2[1] + 1) % 5];
} else if (pos1[1] == pos2[1]) { // Same column
    letter[0] = mat[(pos1[0] + 1) % 5][pos1[1]];
    letter[1] = mat[(pos2[0] + 1) % 5][pos2[1]];
} else { // Rectangle
    letter[0] = mat[pos1[0]][pos2[1]];
    letter[1] = mat[pos2[0]][pos1[1]];
}

encrypt.append(letter[0]).append(letter[1]).append(" ");
}

System.out.println("Encrypted text: " + encrypt.toString());

StringBuilder decrypt = new StringBuilder();
for (String pair : encrypt.toString().split(" ")) {
    char[] letter = pair.toCharArray();

```

```
int[] pos1 = new int[2];
int[] pos2 = new int[2];
for (int i = 0; i < 5; i++) {
    for (int j = 0; j < 5; j++) {
        if (mat[i][j] == letter[0]) {
            pos1[0] = i;
            pos1[1] = j;
        }
        if (mat[i][j] == letter[1]) {
            pos2[0] = i;
            pos2[1] = j;
        }
    }
}
if (pos1[0] == pos2[0]) { // Same row
    letter[0] = mat[pos1[0]][(pos1[1] + 4) % 5];
    letter[1] = mat[pos2[0]][(pos2[1] + 4) % 5];
} else if (pos1[1] == pos2[1]) { // Same column
    letter[0] = mat[(pos1[0] + 4) % 5][pos1[1]];
    letter[1] = mat[(pos2[0] + 4) % 5][pos2[1]];
} else { // Rectangle
    letter[0] = mat[pos1[0]][pos2[1]];
    letter[1] = mat[pos2[0]][pos1[1]];
}
decrypt.append(letter[0]).append(letter[1]);
}
String decryptedText = decrypt.toString().replaceAll("X$", "");
System.out.println("Decrypted text: " + decryptedText);
}
}
```

OUTPUT:

Enter the key :

monorchy

Enter the plain text :

instruments

M O N R C

H Y A B D

E F G I K

L P Q S T

U V W X Z

Prepared text: IN ST RU ME NT SX

Encrypted text: GR TL MX HL CQ XR

Decrypted text: INSTRUMENTS

RESULT:

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

EX.NO: 1c **Encryption and Decryption Hill Cipher**

DATE :

AIM:

To implement a program to encrypt and decrypt using 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 $n \times n$ 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:

```
import java. util.Scanner;

class hillCipher {

    public static int[][] keymat = new int[][] {

        { 1, 2, 1 },

        { 2, 3, 2 },

        { 2, 2, 1 }

    };

    public static int[][] invkeymat = new int[][] {

        { -1, 0, 1 },

        { 2, -1, 0 },

        { -2, 2, -1 }

    };

};
```



```
public static String key = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

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 {  
    Scanner scanner = new Scanner(System.in);  
    System.out.println("Hill Cipher Simulation");  
    System.out.println("-----");  
    System.out.print("Enter the message to encode: ");  
    String msg = scanner.nextLine();  
    String enc = "";  
    String dec = "";  
    int n;  
    msg = msg.toUpperCase().replaceAll("\\s", "");  
    n = msg.length() % 3;  
    if (n != 0) {  
        for (int i = 1; i <= (3 - n); i++) {  
            msg += 'X'; // Padding with '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);  
    scanner.close();  
}}
```

OUTPUT:

Hill Cipher Simulation

Enter the message to encode: Security Laboratory

Padded message: SECURITYLABORATORY

Encoded message: EACSDKLCAEFQDUKSXU

Decoded message: SECURITYLABORATORY

RESULT:

Thus the program for Hill Cipher encryption and decryption algorithm has been implemented and the output has been verified successfully

EX.NO: 1d Encryption and Decryption Vigenere Cipher**DATE :****AIM:**

To implement a program to encryption and decryption using the Vigenere Cipher substitution technique.

ALGORITHM:

1. The Vigenere cipher is a method of encryption of alphabetic text by using a series of different Caesar Ciphers based on the letter 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 repeating the keyword.

PROGRAM:

```
import java.util.*;

public class VigenereCipher
{
    public static String encrypt(String plaintext, String keyword) {
        StringBuilder cipher = new StringBuilder();
        for (int i = 0, j = 0; i < plaintext.length(); i++) {
            char currentChar = plaintext.charAt(i);
            if (Character.isAlphabetic(currentChar)) {
                int shift = keyword.charAt(j % keyword.length()) - 'A';
                char encryptedChar = (char) ((currentChar + shift - 'A') % 26 + 'A');
                cipher.append(encryptedChar);
                j++;
            }
        }
    }
}
```

```

        } else {
            cipher.append(currentChar);
        }
    }
    return cipher.toString();
}

public static String decrypt(String ciphertext, String keyword) {
    StringBuilder decryptedText = new StringBuilder();
    for (int i = 0, j = 0; i < ciphertext.length(); i++) {
        char currentChar = ciphertext.charAt(i);
        if (Character.isAlphabetic(currentChar)) {
            int shift = keyword.charAt(j % keyword.length()) - 'A';
            char decryptedChar = (char) ((currentChar - shift - 'A' + 26) % 26 + 'A');
            decryptedText.append(decryptedChar);
            j++;
        } else {
            decryptedText.append(currentChar);
        }
    }
    return decryptedText.toString();
}

public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.print("Enter the plaintext: ");
    String pt = sc.nextLine().toUpperCase();
    System.out.print("Enter the keyword: ");
    String keyword = sc.nextLine().toUpperCase();
    String encryptedText = encrypt(pt, keyword);
    String decryptedText = decrypt(encryptedText, keyword);
    System.out.println("-----");
}

```

```
        System.out.println("\nEncrypted Text: " + encryptedText);  
        System.out.println("Decrypted Text: " + decryptedText);  
    }  
}
```

OUTPUT:

Enter the plaintext: meetmetomorrow

Enter the keyword: legend

Encrypted Text: XIKXZHESSEUZA

Decrypted Text: MEETMETOMORROW

RESULT:

Thus the program for Vigenere Cipher encryption and decryption algorithm has been implemented and the output has been verified successfully

EX.NO: 2a Encryption and Decryption Using Rail Fence Transposition Technique.**DATE :****AIM:**

To implement a program for encryption and decryption using the 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 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:

```
import java.util.Scanner;

public class Main {

    public static String encrypt(String message, int rails) {

        char[][] railFence = new char[rails][message.length()];

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

            for (int j = 0; j < message.length(); j++) {

                railFence[i][j] = '.';

            }

        }

        int row = 0;

        boolean down = true;

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

            railFence[row][i] = message.charAt(i);

            if (row == 0) {

                down = true;

            } else if (row == rails - 1) {

                down = false;

            }

        }

    }

}
```

```
        if (down) {
            row++;
        } else {
            row--;
        }
    }}
    StringBuilder encryptedMessage = new StringBuilder();
    for (int i = 0; i < rails; i++) {
        for (int j = 0; j < message.length(); j++) {
            if (railFence[i][j] != '.') {
                encryptedMessage.append(railFence[i][j]);
            }
        }
    }
    return encryptedMessage.toString();
}

public static String decrypt(String encryptedMessage, int rails) {
    char[][] railFence = new char[rails][encryptedMessage.length()];
    for (int i = 0; i < rails; i++) {
        for (int j = 0; j < encryptedMessage.length(); j++) {
            railFence[i][j] = '.';
        }
    }
    int row = 0;
    boolean down = true;
    for (int i = 0; i < encryptedMessage.length(); i++) {
        railFence[row][i] = '*';
        if (row == 0) {
            down = true;
        } else if (row == rails - 1) {
            down = false;
        }
    }
    if (down) {
        row++;
    }
```



```
        } else {  
            row--;  
        }  
    }  
    int index = 0;  
    for (int i = 0; i < rails; i++) {  
        for (int j = 0; j < encryptedMessage.length(); j++) {  
            if (railFence[i][j] == '*' && index < encryptedMessage.length()) {  
                railFence[i][j] = encryptedMessage.charAt(index++);  
            }  
        }  
    }  
    StringBuilder decryptedMessage = new StringBuilder();  
    row = 0;  
    down = true;  
    for (int i = 0; i < encryptedMessage.length(); i++) {  
        decryptedMessage.append(railFence[row][i]);  
        if (row == 0) {  
            down = true;  
        } else if (row == rails - 1) {  
            down = false;  
        }  
        if (down) {  
            row++;  
        } else {  
            row--;  
        }  
    }  
    return decryptedMessage.toString();  
}  
  
public static void main(String[] args) {  
    Scanner scanner = new Scanner(System.in);  
    System.out.print("Enter the message to encrypt: ");
```

```
String message = scanner.nextLine();  
int rails = 2;  
String encryptedMessage = encrypt(message, rails);  
System.out.println("Encrypted message: " + encryptedMessage);  
String decryptedMessage = decrypt(encryptedMessage, rails);  
System.out.println("Decrypted message: " + decryptedMessage);  
}  
}
```

OUTPUT:

Enter the message to encrypt: Anna University, Chennai

Encrypted message: An nvriy hnanaUiest,Ceni

Decrypted message: Anna University, Chennai

RESULT:

Thus the program for the Rail fence transposition technique has been implemented and the output has been verified successfully

EX.NO: 2a Encryption, Decryption Using Row and column Transposition technique**DATE :****AIM:**

To implement a program for encryption and decryption using the Row and column Transposition technique

ALGORITHM:

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

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

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

PROGRAM:

```
import java.util.Scanner;

public class Main {

    static void printMatrix(char[][] matrix) {
        for (char[] row : matrix) {
            for (char val : row) {
                System.out.print(val + " ");
            }
            System.out.println();
        }
    }

    static char[][] transposeMatrix(char[][] matrix) {
        int rows = matrix.length;
        int cols = matrix[0].length;
        char[][] transposed = new char[cols][rows];
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
```

```
        transposed[j][i] = matrix[i][j];
    }
}
return transposed;
}

static String encrypt(String plaintext, int numRows, int numCols) {
    char[][] matrix = new char[numRows][numCols];
    int index = 0;
    for (int i = 0; i < numRows; i++) {
        for (int j = 0; j < numCols; j++) {
            if (index < plaintext.length()) {
                matrix[i][j] = plaintext.charAt(index++);
            } else {
                matrix[i][j] = ' '; // Padding character
            }
        }
    }
    char[][] transposedMatrix = transposeMatrix(matrix);
    StringBuilder cipherText = new StringBuilder();
    for (int i = 0; i < transposedMatrix.length; i++) {
        for (int j = 0; j < transposedMatrix[i].length; j++) {
            cipherText.append(transposedMatrix[i][j]);
        }
    }
    return cipherText.toString();
}

static String decrypt(String ciphertext, int numRows, int numCols) {
    char[][] matrix = new char[numCols][numRows];
    int index = 0;
    for (int i = 0; i < numCols; i++) {
        for (int j = 0; j < numRows; j++) {
            if (index < ciphertext.length()) {
```

```

        matrix[i][j] = ciphertext.charAt(index++);
    } else {
        matrix[i][j] = 'X'; // Padding character
    }
}

char[][] originalMatrix = transposeMatrix(matrix);
StringBuilder plainText = new StringBuilder();
for (int i = 0; i < originalMatrix.length; i++) {
    for (int j = 0; j < originalMatrix[i].length; j++) {
        plainText.append(originalMatrix[i][j]);
    }
}

int endIndex = plainText.length();
while (endIndex > 0 && plainText.charAt(endIndex - 1) == 'X') {
    endIndex--;
}

return plainText.substring(0, endIndex);
}

public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter the number of rows: ");
    int numRows = scanner.nextInt();
    System.out.print("Enter the number of columns: ");
    int numCols = scanner.nextInt();
    scanner.nextLine();
    System.out.print("Enter the plaintext (uppercase letters only): ");
    String plaintext = scanner.nextLine().toUpperCase();
    if (!plaintext.matches("[A-Z]*")) {
        System.out.println("Invalid plaintext. Only uppercase letters are allowed.");
        return;
    }
}

```

```
}  
String cipherText = encrypt(plaintext, numRows, numCols);  
System.out.println("Encrypted text: " + cipherText);  
String decryptedText = decrypt(cipherText, numRows, numCols);  
System.out.println("Decrypted text: " + decryptedText);  
}  
}
```

OUTPUT:

Enter the number of rows: 3

Enter the number of columns: 4

Enter the plaintext (uppercase letters only): securitylab

Encrypted text: SRLEIACTBUY

Decrypted text: SECURITYLAB

RESULT:

Thus the program for the Row and Column transformation technique has been implemented and the output has been verified successfully

EX.NO: 3**Data Encryption Standard (DES) Algorithm****DATE :****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 the cipher class, specify the following information, and separate by a slash(/).
 - a. Algorithm name
 - b. Mode(optional)
 - c. Padding scheme(optional)
3. Convert String into Byte[] array format.
4. Make a cipher in encrypt mode, and encrypt it using Cipher.doFinal() method.
5. Make Cipher in decrypt mode, and decrypt it with Cipher.doFinal() method.

PROGRAM:

```
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;
import java.util.Scanner;

public class Main
{
    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);
System.out.print("Enter message:");
String s1=new Scanner(System.in).nextLine();
byte[] text = s1.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:

Message Encryption Using DES Algorithm

Enter message:Secret Information

Message [Byte Format] : [B@57ffcd7

Message : Secret Information

Encrypted Message: [B@553f17c

Decrypted Message: Secret Information

RESULT:

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

EX.NO: 4**Advanced Encryption Standard (AES) Algorithm****DATE :****AIM:**

To use the 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 a 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 4x4 column-major order array of bytes, termed the state.

PROGRAM:

```
import java.io.UnsupportedEncodingException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Arrays;
import java.util.Base64;
import java.util.Scanner;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;

public class Main {
    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); // Use only the first 128 bit
```

```
        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) {
```

```
Scanner scanner = new Scanner(System.in);
System.out.println("URL Encryption Using AES Algorithm\n-----");
System.out.print("Enter the original URL: ");
String originalString = scanner.nextLine();
System.out.print("Enter the secret key: ");
String secretKey = scanner.nextLine();
String encryptedString = Main.encrypt(originalString, secretKey);
String decryptedString = Main.decrypt(encryptedString, secretKey);
System.out.println("Original URL : " + originalString);
System.out.println("Encrypted URL : " + encryptedString);
System.out.println("Decrypted URL : " + decryptedString);
scanner.close();
}
}
```

OUTPUT:

URL Encryption Using AES Algorithm

Enter the original URL: www.annauniv.edu

Enter the secret key: annaUniversity

Original URL: www.annauniv.edu

Encrypted URL: vibpFJW6Cvs5Y+L7t4N6YWWe07+JzS1d3CU2h3mEvEg=

Decrypted URL: www.annauniv.edu

RESULT:

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

EX.NO: 5**Implement RSA Algorithm Using HTML and JavaScript****DATE:****AIM:**

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

ALGORITHM:

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

PROGRAM:

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>RSA Encryption</title>

</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" id="p" required></td>

      </tr>
```

```
<tr>

    <td>Enter Second Prime Number:</td>

    <td><input type="number" id="q" required></td>

</tr>

<tr>

    <td>Enter the Message (cipher text):<br>[A=1, B=2,...]</td>

    <td><input type="number" id="msg" required></td>

</tr>

<tr>

    <td>Public Key (n):</td>

    <td><p id="publickey"></p></td>

</tr>

<tr>

    <td>Exponent (e):</td>

    <td><p id="exponent"></p></td>

</tr>

<tr>

    <td>Private Key (d):</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>
```

```
<script type="text/javascript">

function RSA() {

    var gcd, p, q, no, n, t, e, i, x, d, ct;

    gcd = function (a, b) { return (!b) ? a : gcd(b, a % b); };

    p = parseInt(document.getElementById('p').value);
    q = parseInt(document.getElementById('q').value);
    no = parseInt(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;
        }
    }

    ct = Math.pow(no, e) % n; // Encrypting

    document.getElementById('publickey').innerHTML = n;
    document.getElementById('exponent').innerHTML = e;
    document.getElementById('privatekey').innerHTML = d;
    document.getElementById('ciphertext').innerHTML = ct;

}

</script>

</body>

</html>
```

OUTPUT:**RSA Algorithm****Implemented Using HTML & JavaScript**

Enter First Prime Number:

Enter Second Prime Number:

Enter the Message (cipher text):

[A=1, B=2,...]

Public Key (n):

Exponent (e):

Private Key (d):

Cipher Text:

RSA Algorithm**Implemented Using HTML & JavaScript**

Enter First Prime Number:

Enter Second Prime Number:

Enter the Message (cipher text):

[A=1, B=2,...]

Public Key (n): 3127

Exponent (e): 3

Private Key (d): 2011

Cipher Text: 1394

RESULT:

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

EX.NO: 6**Diffie-Hellman key exchange algorithm****DATE:****AIM:**

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

ALGORITHM:

1. Alice and Bob publicly agree to use a modulus $p=23$ and base $g=5$ (which is primitive root modulo 23).
2. Alice chooses a secret integer $a=4$, then sends Bob $A=g^a \bmod p$
 - a. $A=5^4 \bmod 23=4$
3. Bob chooses a secret integer $b=3$, then sends Alice $B=g^b \bmod p$
 - a. $B=5^3 \bmod 23=10$
4. Alice computes $s=B^a \bmod p$
 - a. $S=10^4 \bmod 23=18$
5. Bob computes $s=A^b \bmod p$
 - a. $S=4^3 \bmod 23=18$
6. Alice and Bob now share a secret (the number 18).

PROGRAM:

```
import java.util.Scanner;

class Main {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter a prime number (p): ");

        int p = scanner.nextInt();

        System.out.print("Enter a primitive root (g): ");

        int g = scanner.nextInt();

        System.out.print("Enter Alice's secret (x): ");

        int x = scanner.nextInt();

        System.out.print("Enter Bob's secret (y): ");

        int y = scanner.nextInt();

        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("\nSimulation of Diffie-Hellman Key Exchange Algorithm");
System.out.println("-----");
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);
if ((aliceComputes == sharedSecret) && (aliceComputes == bobComputes)) {
    System.out.println("Success: Shared Secrets Match! " + sharedSecret);
} else {
    System.out.println("Error: Shared Secrets do not Match");
}
scanner.close();
}
}
```

OUTPUT:

Enter a prime number (p): 23

Enter a primitive root (g): 5

Enter Alice's secret (x): 4

Enter Bob's secret (y): 3

Simulation of Diffie-Hellman Key Exchange Algorithm

Alice Sends : 4.0

Bob Computes : 18.0

Bob Sends : 10.0

Alice Computes : 18.0

Shared Secret : 18.0

Success: Shared Secrets Match! 18.0

RESULT:

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

EX.NO: 7**SHA – 1 Algorithm****DATE:****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 block in total message).

PROGRAM:

```
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:

Message digest object info:

Algorithm=SHA1

Provider=SUN version 11

ToString=SHA1 Message Digest from SUN, <initialized>

SHA1("")=DA39A3EE5E6B4B0D3255BFEF95601890AFD80709

SHA1("abc")=A9993E364706816ABA3E25717850C26C9CD0D89D

SHA1("abcdefghijklmnopqrstuvwxyz")=32D10C7B8CF96570CA04CE37F2A19D84240D3
A89

RESULT:

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

EX.NO: 8**Digital Signature Standard****DATE:****AIM:**

To implement the SIGNATURE SCHEME – Digital Signature Standard.

ALGORITHM:

1. Create a KeyPairGenerator object.
2. Initailaize the KeyPairGenerator object.
3. Generate the KeyPairGenerator.
4. Get the private key from the pair.
5. Create the signature object.
6. Intialize 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 Main {
    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:

Enter some text:

Hi how are you

Digital signature for given text: 0=!s~LylP*m-
 g|@YtI\eezBgj2M

RESULT:

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

Ex.No : 9

Demonstration of Intrusion Detection System(IDS)

Date:

AIM:

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

STEPS ON CONFIGURING AND INTRUSION DETECTION:

1. Download Snort from the Snort.org website. (<http://www.snort.org/snort-downloads>)
2. Download Rules(<https://www.snort.org/snort-rules>). You must register to get the rules. (You should download these often)
3. Double click on the .exe to install snort. This will install snort in the “C:\Snort” folder. It is important to have WinPcap (<https://www.winpcap.org/install/>) installed
4. Extract the Rules file. You will need WinRAR for the .gz file.
5. Copy all files from the “rules” folder of the extracted folder. Now paste the rules into “C:\Snort\rules” folder.
6. Copy “snort.conf” file from the “etc” folder of the extracted folder. You must paste it into “C:\Snort\etc” folder. Overwrite any existing file. Remember if you modify your snort.conf file and download a new file, you must modify it for Snort to work.
7. Open a command prompt (cmd.exe) and navigate to folder “C:\Snort\bin” folder. (at the Prompt, type cd\snort\bin)
8. To start (execute) snort in sniffer mode use following command: snort -dev
-i 3
-i indicates the interface number. You must pick the correct interface number. In my case, it is 3.
-dev is used to run snort to capture packets on your network.
To check the interface list, use following command:
snort -W

```

Administrator: C:\Windows\system32\cmd.exe
Total Memory Allocated: 0
Snort exiting
C:\Snort\bin>snort -W

--> Snort! <--
Version 2.9.6.0-WIN32 GRE (Build 47)
By Martin Roesch & The Snort Team: http://www.snort.org/snort/snort-team
Copyright (C) 2014 Cisco and/or its affiliates. All rights reserved.
Copyright (C) 1998-2013 Sourcefire, Inc., et al.
Using PCRE version: 8.10 2010-06-25
Using ZLIB version: 1.2.3

Index  Physical Address      IP Address      Device Name      Description
-----
1      00:00:00:00:00:00      0000:0000:fe80:0000:0000:0000:78d2:6299 \Device\
NPF_{45DAC1EF-7002-4C33-B712-AE311620E7A} VMware Virtual Ethernet Adapter
2      00:00:00:00:00:00      0000:0000:fe80:0000:0000:0000:bca1:2f66 \Device\
NPF_{C355D233-3D77-484F-A344-65626159980E} VMware Virtual Ethernet Adapter
3      00:00:00:00:00:00      0000:0000:fe80:0000:0000:0000:ada3:46c9 \Device\
NPF_{3264BC0F-4BF2-49C5-B5D9-A12EFE40F17C} Microsoft

C:\Snort\bin>

```

Finding an interface

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

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

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

var HOME_NET 192.168.1.0/24 (You will normally see any here)

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

Example:

example snort

12. Change the RULE_PATH variable to the path of rules folder. var

RULE_PATH c:\snort\rules

Path to rules

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

C:\Snort\lib\snort_dynamicccpreprocessor

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.

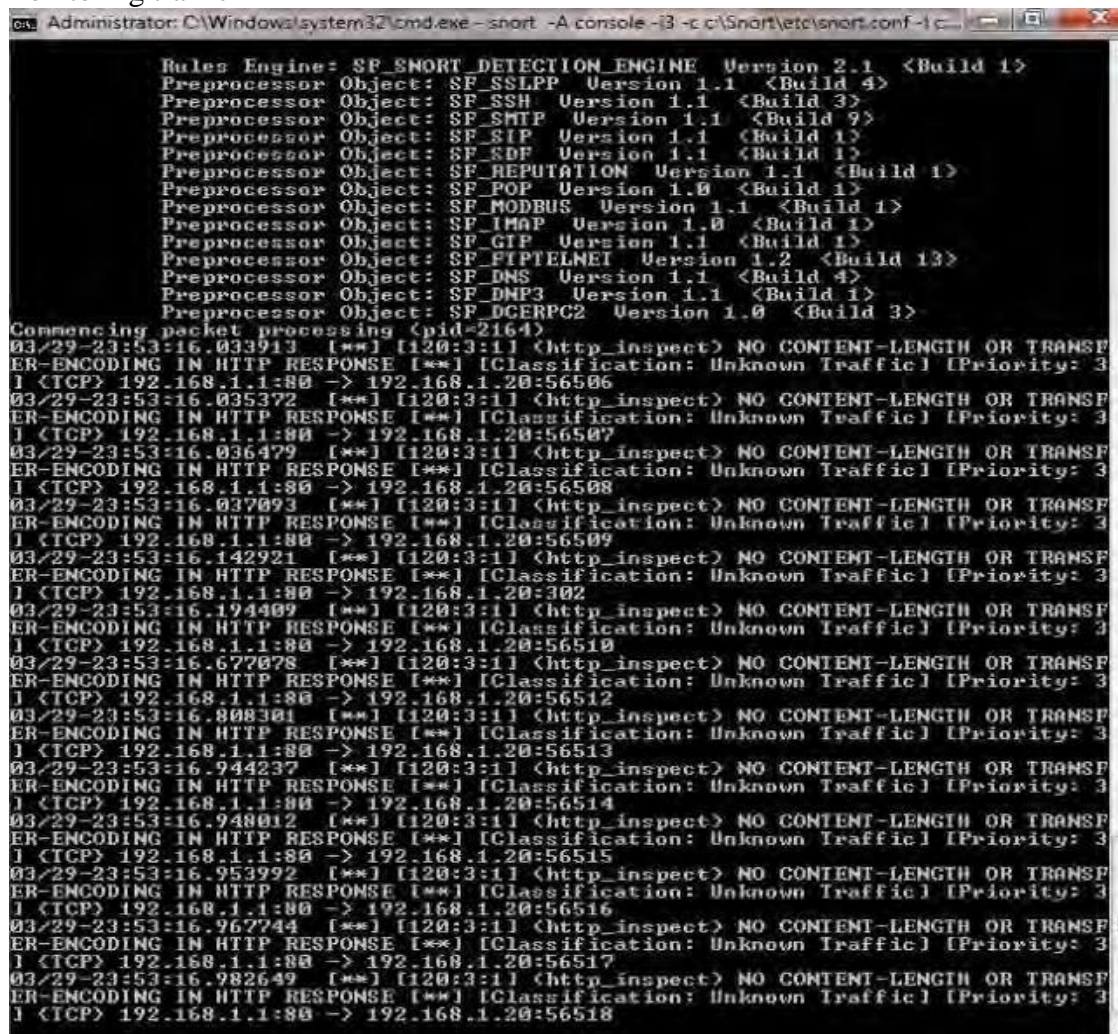
To generate Log files in ASCII mode, you can use following command while running snort in IDS mode:

```
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

Exploring N-Stalker, a Vulnerability Assessment Tool

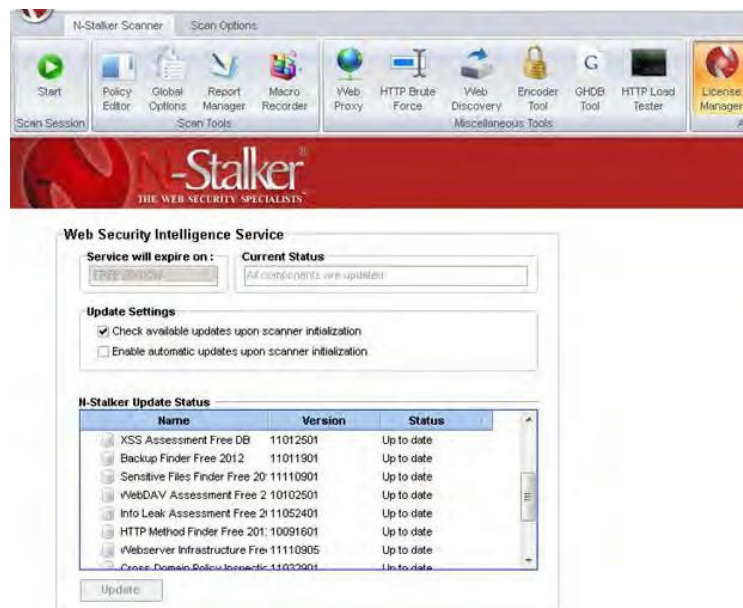
Date:

AIM:

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

EXPLORING N-STALKER:

- N-Stalker Web Application Security Scanner is a Web security assessment tool.
 - It incorporates a well-known N-Stealth HTTP Security Scanner and 35,000 Web attack signature database.
 - This tool also comes in both free and paid versions.
 - Before scanning the target, go to the “License Manager” tab, perform the update.
 - Once updated, you will note the status as up to date.
 - You need to download and install N-Stalker from www.nstalker.com.
1. Start N-Stalker from a Windows computer. The program is installed under Start ⇨ Programs ⇨ N-Stalker ⇨ N-Stalker Free Edition.
 2. Enter a host address or a range of addresses to scan.
 3. Click Start Scan.
 4. After the scan completes, the N-Stalker Report Manager will prompt
 5. you can 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, the next step is “Optimize settings” which will crawl the whole website for further analysis.

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

N-Stalker Scan Wizard

Start Web Application Security Scan Session

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

Enter Web Application URL

(E.g. http://www.example.it/, https://www.test.it/ActualDirectory/, etc.)

Choose Scan Policy

Load Scan Session

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

Load Spider Data

(You may load spider data from previously saved scan sessions)

☐ Use local cache from previously saved session (Avoid reweb crawling)

N-Stalker Scan Wizard

Start Web Application Security Scan Session

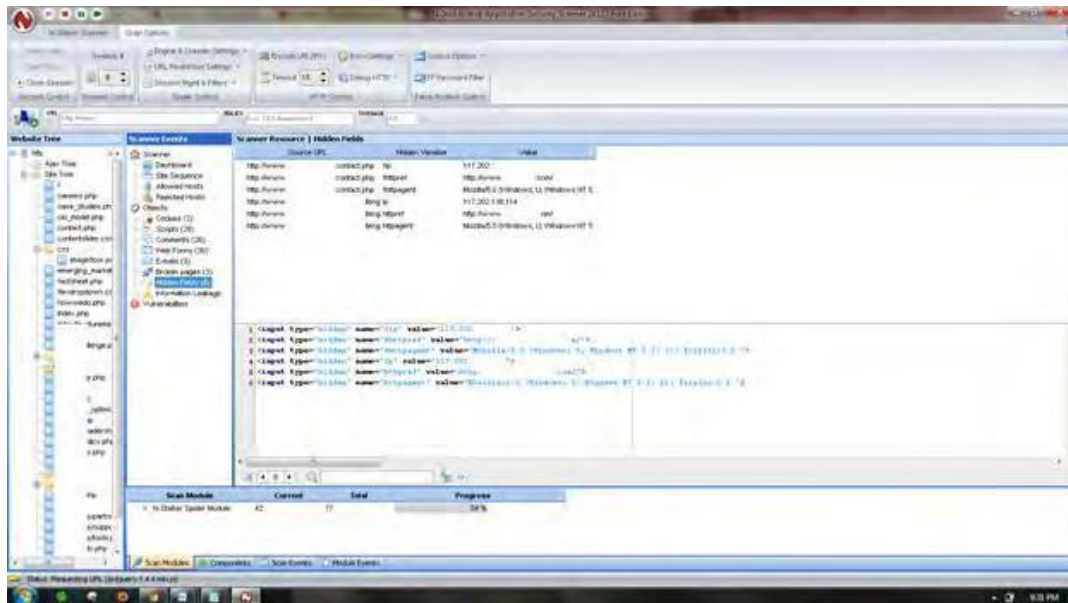
You must enter an URL and choose policy. Scan Settings may be configured.

Review Summary

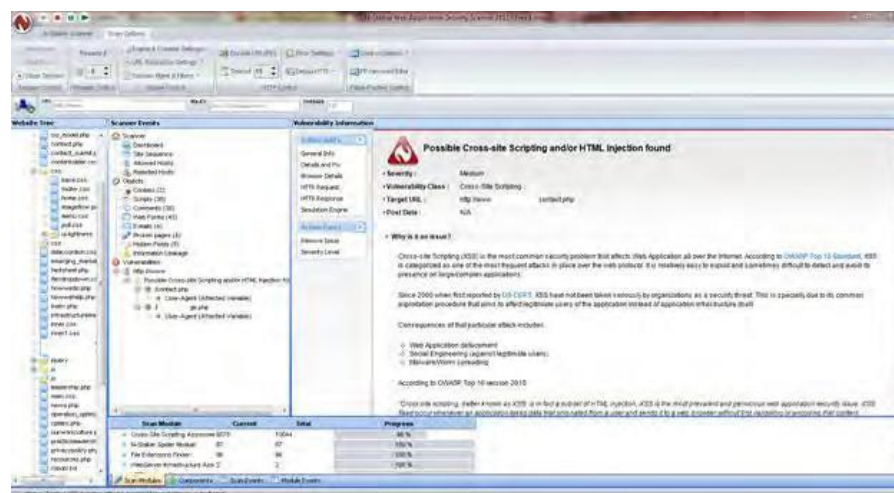
Scanning Settings

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

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?



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

Ex. No : 11a

Defeating Malware - Building Trojans

Date :

AIM:

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

PROCEDURE:

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

TROJAN:

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

CODE:

Trojan.bat

@echo off

:x

start mspaint

start notepad

start cmd start

explorer start

control start

calc goto x

OUTPUT

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

RESULT:

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

Ex.No:11b

Defeating Malware - Rootkit hunter

Date :

AIM:

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

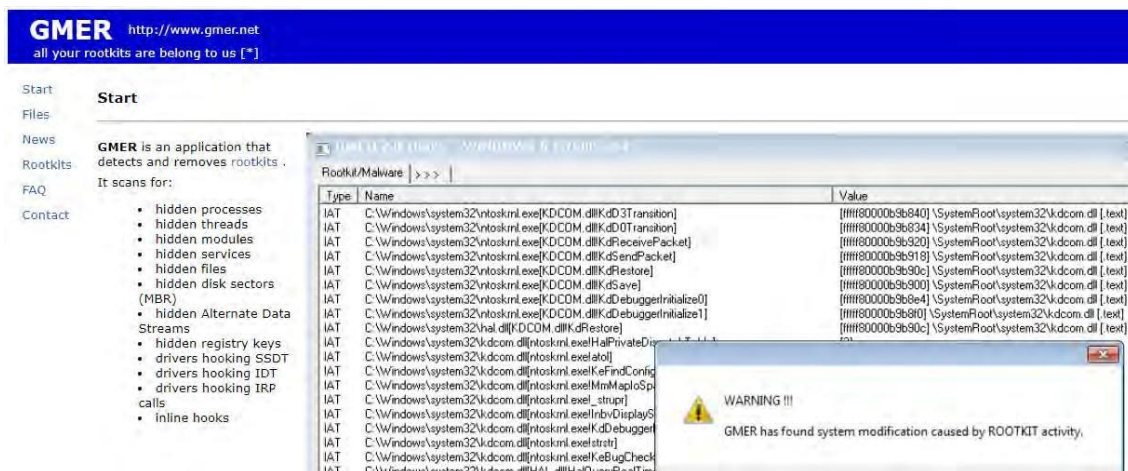
ROOTKIT HUNTER:

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

GMER ROOTKIT TOOL:

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

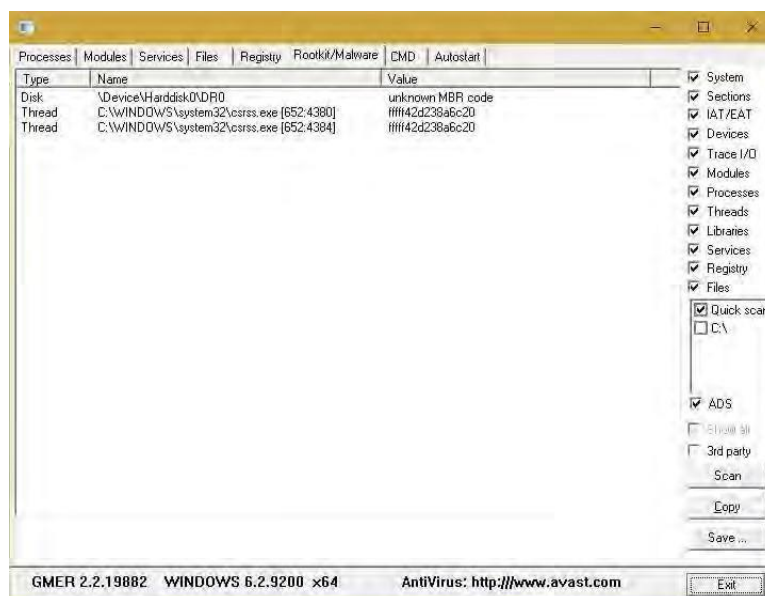
Step 1



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

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

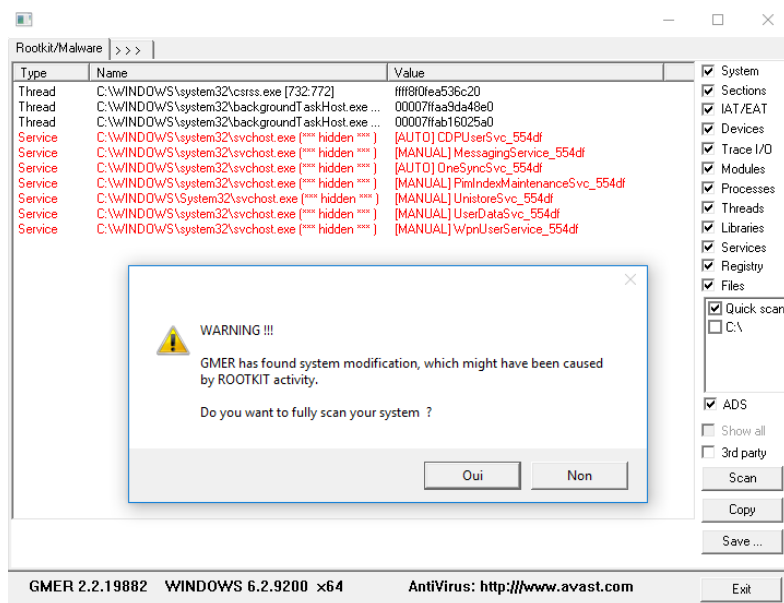
Step 2



Double-click the icon for the program.

Click the "Scan" button in the lower-right corner 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.