SRM VALLIAMMAI ENGINEERING COLLEGE

(An Autonomous Institution)

SRM Nagar, Kattankulathur – 603 203

DEPARTMENT OF CYBER SECURITY

Lab Manual

SECURITY LABORATORY

(REGULATION 2023)

VI SEMESTER-THIRD YEAR

CY3664 – CRYPTOGRAPHY AND NETWORK SECURITY LABORATORY

Regulation – 2019

Academic Year: 2025 – 2026 (EVEN)

Prepared by

Ms. K. R. Nandhashree, Assistant Professor

PROGRAMME OUTCOMES

1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using the first principles of mathematics, natural sciences, and engineering sciences.

3. Design/development of solutions: Design solutions for the complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

4. Conduct investigations of complex problems: Use research-based knowledge and research methods including the design of experiments, analysis and interpretation of data, and the synthesis of the information to provide valid conclusions.

5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO1: Exhibit proficiency in planning, implementing and evaluating team oriented-software programming solutions to specific business problems and society needs.

PSO2: Demonstrate professional skills in applying programming skills, competency and decision making capability through hands-on experiences.

PSO3: Apply logical thinking in analyzing complex real world problems, and use professional and ethical behaviors to provide proper solutions to those problems.

PSO4: Demonstrate the ability to work effectively as part of a team in applying technology to Business and personal situations.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

PEO1: To mould students to exhibit top performance in higher education and research and to become a State-of –the-art technocrat.

PEO2: To impart the necessary background in Computer Science and Engineering by providing solid foundation in Mathematical, Science and Engineering fundamentals.

PEO3: To equip the students with the breadth of Computer Science and Engineering innovate novel solutions for the benefit of common man.

CO – PO and PSO MAPPING:

Course Outcomes	**Programme Outcomes (PO)**												**PSO**			
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**1**	**2**	**3**	**4**
CO 1	3		3		3						2			3	3	
CO 2	3		3		3						2		3		3	2
CO 3	3		3		3						2		3	3		
CO 4	3		3		3						2		3	3		
CO 5	3		3		3	2				2	2		3	3		

1904008 **SECURITY LABORTORY** **L T P C**
0 0 4 2

OBJECTIVES:
The student should be made to:

- To learn different cipher techniques.
- To implement the algorithms DES.
- To implement the RSA Algorithm.
- To implement the MD5, Digital Signature Algorithms.
- To use network security tools and vulnerability assessment tools.

LIST OF EXPERIMENTS

1. Perform encryption, decryption using the following substitution techniques
   (i) Ceaser cipher (ii) playfair cipher (iii) Hill Cipher (iv) Vigenere cipher
2. Perform encryption and decryption using following transposition techniques
   (i) Rail fence (ii) row & Column Transformation
3. Apply DES algorithm for practical applications.
4. Apply AES algorithm for practical applications.
5. Implement RSA Algorithm using HTML and JavaScript
6. Implement the Diffie-Hellman Key Exchange algorithm for a given problem.
7. Calculate the message digest of a text using the SHA-1 algorithm.
8. Implement the SIGNATURE SCHEME – Digital Signature Standard.
9. Demonstrate intrusion detection system (ids) using any tool eg. Snort or any other s/w.
10. Automated Attack and Penetration Tools Exploring N-Stalker, a Vulnerability Assessment Tool
11. Defeating Malware
    i) Building Trojans ii) Rootkit Hunter

TOTAL: 60 PERIODS

LIST OF EQUIPMENT FOR A BATCH OF 30 STUDENTS:

SOFTWARE: C / C++ / Java or equivalent compiler GnuPG, Snort, N–Stalker or Equivalent

HARDWARE: Standalone desktops –30 Nos. (or) Server supporting 30 terminals or more.

INDEX

Ex. No. : 1(i) **CAESAR CIPHER**

AIM:

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

ALGORITHM DESCRIPTION:

- It is a type of substitution cipher in which each letter in the plain text is replaced by a letter some fixed number of positions down the alphabet. For example, with a left shift of 3, D would be replaced by A, E would become B, and so on.
- The method is named after Julius Caesar, who used it in his private correspondence.
- The transformation can be represented by aligning two alphabets; the cipher alphabet is the plain alphabet rotated left or right by some number of positions.
- 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.
- Encryption of a letter x by a shift n can be described mathematically as, En(x) = (x + n) mod26
- Decryption is performed similarly,

  Dn(x)=(x - n) mod26

PROGRAM:

```
import java.util.*;
class caesarCipher
{
public static String encode(String enc, int offset)
{
offset = offset % 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' + offset) % 26 ));
}
else
{
encoded.append((char) ('a' + (i - 'a' + offset) % 26 ));
}
}
 else
{
encoded.append(i);
}
}
return encoded.toString();
}
public static String decode(String enc, int offset)
{
return encode(enc, 26-offset);
}
public static void main (String[] args) throws java.lang.Exception
{
String msg = "Hello welcome to Security Laboratory";
System.out.println("simulation of Caesar Cipher");
System.out.println("input message   : " + msg);
System.out.printf( "encoded message : ");
System.out.println(caesarCipher.encode(msg, 12));
System.out.printf( "decoded message : ");
System.out.println(caesarCipher.decode(caesarCipher.encode(msg, 12), 12));
```

```
}

}
```

OUTPUT

stdin:

```
Standard input is empty
```

stdout:

```
simulation of Caesar Cipher
input message       : Hello welcome to Security Laboratory
encoded message     : Tqxxaiqxoayq fa EqogdufkXmnadmfadk
decoded message     : Hello welcome to Security Laboratory
```

RESULT:

Thus the program for encrypting a plain text and decrypting a cipher text using Caesar Cipher (shift cipher) substitution technique was executed and verified successfully.

Ex. No. : 1(ii) **PLAY FAIR CIPHER**

AIM:

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

ALGORITHM DESCRIPTION:

- The Playfair cipher uses a 5 by 5 table containing a key word or phrase.
- To generate the key table, first fill the spaces in the table with the letters of the keyword, then fill the remaining spaces with the rest of the letters of the alphabet in order (usually omitting "Q" to reduce the alphabet to fit; other versions put both "I" and "J" in the same space).
- The key can be written in the top rows of the table, from left to right, or in some other pattern, such as a spiral beginning in the upper-left-hand corner and ending in the centre.
- The keyword together with the conventions for filling in the 5 by 5 table constitutes the cipher key. To encrypt a message, one would break the message into diagrams (groups of 2 letters) such that, for example, "HelloWorld" becomes "HE LL OW OR LD", and map them out on the key table. Then apply the following 4 rules, to each pair of letters in the plaintext:
- If both letters are the same (or only one letter is left), add an "X" after the first letter. Encrypt the new pair and continue. Some variants of Playfair use "Q" instead of "X", but any letter, itself uncommon as a repeated pair, will do.
- If the letters appear on the same row of your table, replace them with the letters to their immediate right respectively (wrapping around to the left side of the row if a letter in the original pair was on the right side of the row).
- If the letters appear on the same column of your table, replace them with the letters immediately below respectively (wrapping around to the top side of the column if a letter in the original pair was on the bottom side of the column).
- If the letters are not on the same row or column, replace them with the letters on the same row respectively but at the other pair of corners of the rectangle defined by the original pair. The order is important – the first letter of the encrypted pair is the one that lies on the same row as the first letter of the plaintext pair.
- To decrypt, use the INVERSE (opposite) of the last 3 rules, and the 1st as-is (dropping any extra "X"s, or "Q"s that do not make sense in the final message when finished).

PROGRAM :

```
import java.awt.Point;

 import java.util.*;

class playfairCipher

{

private static char[][] charTable;

private static Point[] positions;

private static String prepareText(String s, booleanchgJtoI)

{
```

```
s = s.toUpperCase().replaceAll("[^A-Z]", "");
return chgJtoI ? s.replace("J", "I") : s.replace("Q", "");
}
private static void createTbl(String key, booleanchgJtoI)
{
charTable = new char[5][5];
positions = new Point[26];
String s = prepareText(key + "ABCDEFGHIJKLMNOPQRSTUVWXYZ", chgJtoI);
int len = s.length();
for (int i = 0, k = 0; i<len; i++)
{
char c = s.charAt(i);
if (positions[c - 'A'] == null)
{
charTable[k / 5][k % 5] = c;
positions[c - 'A'] = new Point(k % 5, k / 5);
k++;
}
}
}
private static String codec(StringBuilder txt, int dir)
{
int len = txt.length(); for (int i = 0; i<len; i += 2)
{
char a = txt.charAt(i); char b = txt.charAt(i + 1);
 int row1 = positions[a - 'A'].y;
 int row2 = positions[b - 'A'].y;
 int col1 = positions[a - 'A'].x;
 int col2 = positions[b - 'A'].x;
```

```
if (row1 == row2)
{
col1 = (col1 + dir) % 5; col2 = (col2 + dir) % 5;
}
 else if (col1 == col2)
{
row1 = (row1 + dir) % 5; row2 = (row2 + dir) % 5;
}
else
{
Inttmp = col1; col1 = col2;
col2 = tmp;
}
txt.setCharAt(i, charTable[row1][col1]);
txt.setCharAt(i + 1, charTable[row2][col2]);
}
return txt.toString();
}
private static String encode(String s)
{
StringBuilder sb = new StringBuilder(s);
for (int i = 0; i<sb.length(); i += 2)
{
if (i == sb.length() - 1)
{
sb.append(sb.length() % 2 == 1 ? 'X' : "");
}
else if (sb.charAt(i) == sb.charAt(i + 1))
{ sb.insert(i + 1, 'X');
}
}
 return codec(sb, 1);
}
private static String decode(String s)
{
return codec(new StringBuilder(s), 4);
}
public static void main (String[] args) throws java.lang.Exception
{
String key = "mysecretkey";
String txt = "CRYPTOLABS";
 /* make sure string length is even */
/* change J to I */
booleanchgJtoI = true; createTbl(key, chgJtoI);
String enc = encode(prepareText(txt, chgJtoI));
System.out.println("simulation of Playfair Cipher");
System.out.println("input message   : " + txt);
System.out.println("encoded message : " + enc);
System.out.println("decoded message : " + decode(enc));
}
}
```

OUTPUT :

stdin:

```
Standard input is empty
```

stdout:

```
simulation of Playfair Cipher
input message       : CRYPTOLABS
encoded message     : MBENKNPRKC
decoded message     : CRYPTOLABS
```

RESULT:

Thus the program to encrypt a plain text and decrypt a cipher text using play fair Cipher substitution technique. was executed and verified successfully.

Ex. No. : 1(iii) **HILL CIPHER**

AIM:

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

ALGORITHM DESCRIPTION:

- The Hill cipher is a substitution cipher invented by Lester S. Hill in 1929.
- Each letter is represented by a number modulo 26. To encrypt a message, each block of n letters is multiplied by an invertible n × n matrix, again modulus26.
- To decrypt the message, each block is multiplied by the inverse of the matrix used for encryption. The matrix used for encryption is the cipher key, and it should be chosen randomly from the set of invertible n × n matrices (modulo26).
- The cipher can, be adapted to an alphabet with any number of letters.
- All arithmetic just needs to be done modulo the number of letters instead of modulo26.

PROGRAM :

```
import java.util.*; 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");
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

stdin:

```
Standard input is empty
```

stdout:

```
simulation of Hill Cipher
input message          :Security Laboratory
padded message         : SECURITYLABORATORY
encoded message         : EACSDKLCAEFQDUKSXU
decoded message        : SECURITYLABORATORY
```

RESULT:

Thus the program to encrypt and decrypt using the Hill cipher substitution technique was executed and verified successfully

Ex. No. : 1(iv) **VIGENERE CIPHER**

AIM:

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

ALGORITHM DESCRIPTION:

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

PROGRAM:

```
import java.util.*;
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("simulation of Vigenere Cipher");
System.out.println("input message   : " + msg);
String enc = encode(msg, key);
System.out.println("encoded message : " + enc);
System.out.println("decoded message : " + decode(enc, key));
}
}
```

OUTPUT

stdin:

Standard input is empty

stdout:

```
simulation of Vigenere Cipher
 input message           : SecurityLaboratory
encoded message       : NMIYEMKCNIQVVROWXC
decoded message       : SECURITYLABORATORY
```

RESULT:

Thus the program program for encryption and decryption using vigenere cipher substitution technique was executed and verified successfully.

Ex. No. : 2 (i) **RAIL FENCE CIPHER**

AIM:

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

ALGORITHM DESCRIPTION:

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

PROGRAM :

```
import java.util.*;
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 = "hellorailfencecipher";
int depth = 2;
enc = rf.encode(msg, depth);
dec = rf.decode(enc, depth);
System.out.println("simulation of Railfence Cipher");
System.out.println("input message   : " + msg);
System.out.println("encoded message : " + enc);
System.out.printf( "decoded message : " + dec);
}
}
```

OUTPUT

stdin:

```
Standard input is empty
```

stdout:

```
Simulation of Railfence Cipher
input message       : hello railfence cipher
encoded message     : hloal eccpeelri fneihr
decoded message     : hello railfence cipher
```

RESULT:

Thus the program for encryption and decryption using rail fence transposition technique was executed and verified successfully.

Ex. No. : 2 (ii) RAIL FENCE – ROW & COLUMN TRANSFORMATION

AIM:

To implement Rail fence - row & column transformation in Java to encrypt and decrypt a given plain text.

ALGORITHM DESCRIPTION:

- The railfence cipher is a very simple, easy to crack cipher. It is a transposition cipher that follows a simple rule for mixing up the characters in the plaintext to form the cipher text. The railfence cipher offers essentially no communication security, and it will be shown that it can be easily broken even by hand.
- Although weak on its own, it can be combined with other ciphers, such as a substitution cipher, the combination of which is more difficult to break than either cipher on it's own.
- Many websites claim that the rail-fence cipher is a simpler "write down the columns, read along the rows" cipher. This is equivalent to using an un-keyed columnar transposition cipher.

PROGRAM:

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

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

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

OUTPUT:

```
Enter plain text: railfencecipher

Enter depth for Encryption:

3

Encrypted text is: rlnchafcieieepr

Decrypted text is: Railfencecipher
```

RESULT:

Thus the implementation Rail fence - row & column transformation in Java to encrypt and decrypt a given plain text was executed and verified successfully.

Ex. No. : 3 APPLY DES ALGORITHM FOR PRACTICAL APPLICATIONS.

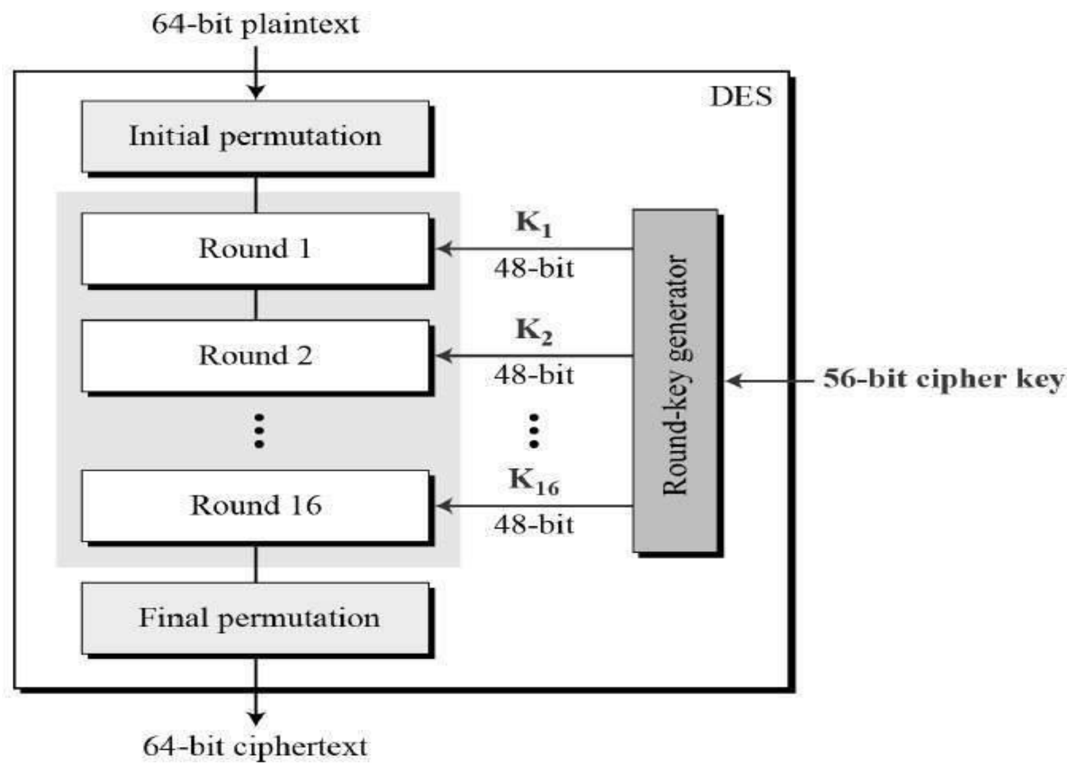
AIM:

To develop a program to apply DES algorithm for practical applications.

ALGORITHM DESCRIPTION:

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

General Structure of DES is depicted in the following illustration



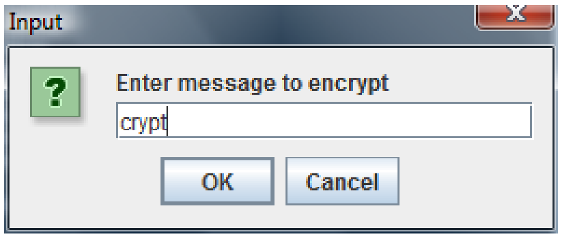
PROGRAM:

DES :-

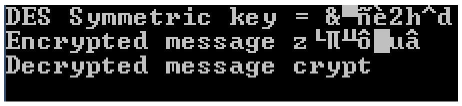
```
import javax.swing.*;
import java.security.SecureRandom;
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
 import java.util.Random ;
clss DES
{
byte[] skey = new byte[1000];
String skeyString;
static byte[] raw;
String inputMessage,encryptedData,decryptedMessage;
public DES()
 {
Try
 {
generateSymmetricKey();
inputMessage=JOptionPane.showInputDialog(null,"Enter message to encrypt");
byte[] ibyte = inputMessage.getBytes();
 byte[] ebyte=encrypt(raw, ibyte);
String encryptedData = new String(ebyte);
System.out.println("Encrypted message "+encryptedData);
JOptionPane.showMessageDialog(null,"Encrypted Data "+"\n"+encryptedData);
byte[] dbyte= decrypt(raw,ebyte);
String decryptedMessage = new String(dbyte);
System.out.println("Decrypted message "+decryptedMessage);
JOptionPane.showMessageDialog(null,"Decrypted Data "+"\n"+decryptedMessage);
}
catch(Exception e)
{
System.out.println(e);
}
}
void generateSymmetricKey()
{
 try
{
Random r = new Random(); intnum = r.nextInt(10000);
String knum = String.valueOf(num);
 byte[] knumb = knum.getBytes();
skey=getRawKey(knumb);
skeyString = new String(skey);
System.out.println("DES Symmetric key = "+skeyString);
 }
catch(Exception e)
{
System.out.println(e);
}
}
private static byte[] getRawKey(byte[] seed) throws Exception
{
KeyGeneratorkgen = KeyGenerator.getInstance("DES");
SecureRandomsr = SecureRandom.getInstance("SHA1PRNG");
sr.setSeed(seed);
kgen.init(56, sr);
```

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

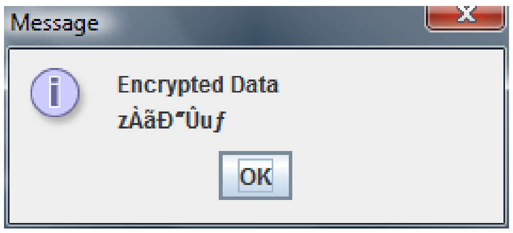
OUTPUT:



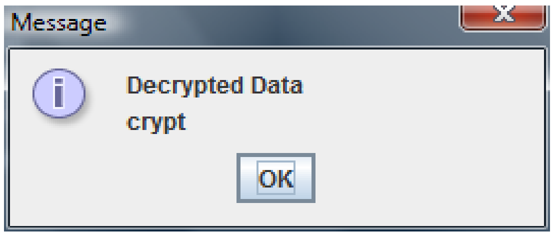
```
Enter message to encrypt
crypt
```



```
DES Symmetric key = &▀ñè2h^d
Encrypted message z╚╥╝ô█uâ
Decrypted message crypt
```



```
Encrypted Data
zÀãÐ˝Ûuƒ
```



```
Decrypted Data
crypt
```

RESULT:

Thus the program for apply DES algorithm for practical applications was executed and verified successfully.

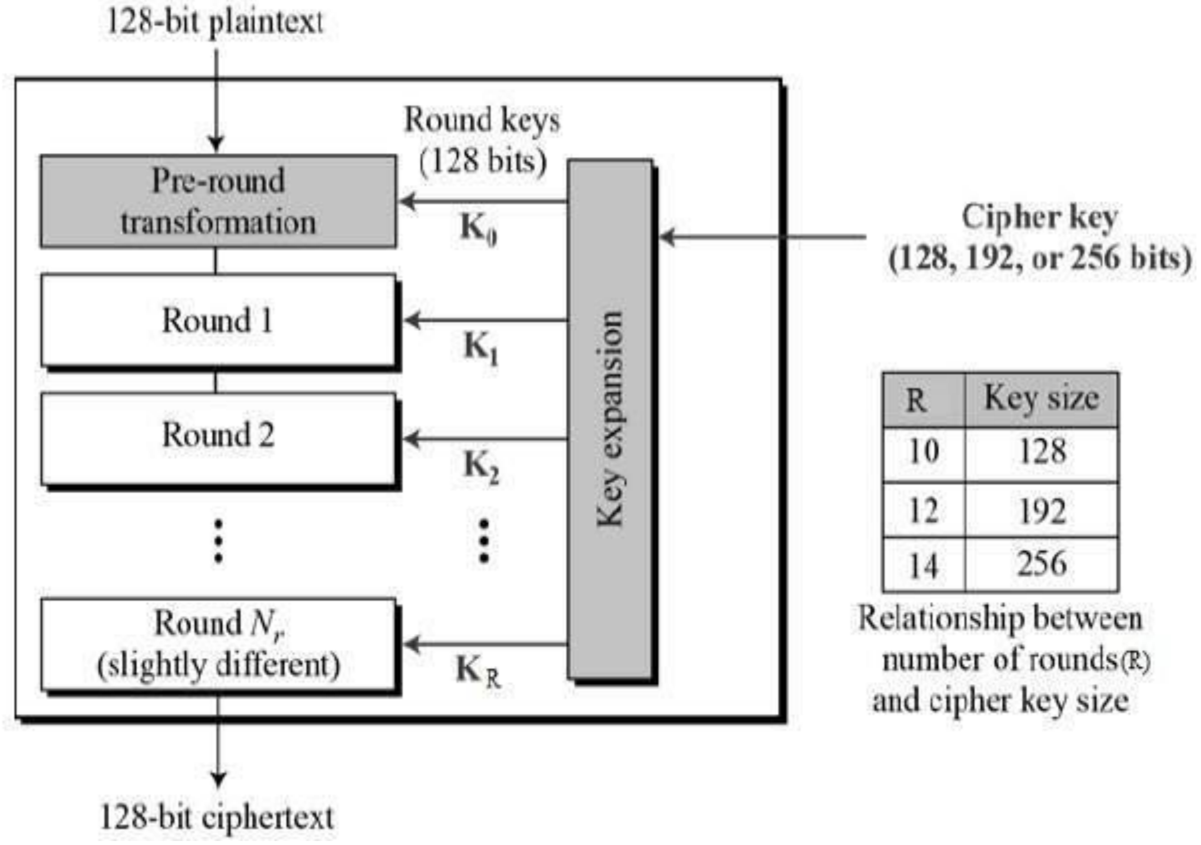
Ex. No. : 4 APPLY AES ALGORITHM FOR PRACTICAL APPLICATIONS.

AIM:

To develop a program to apply AES algorithm for practical applications

ALGORITHM DESCRIPTION:

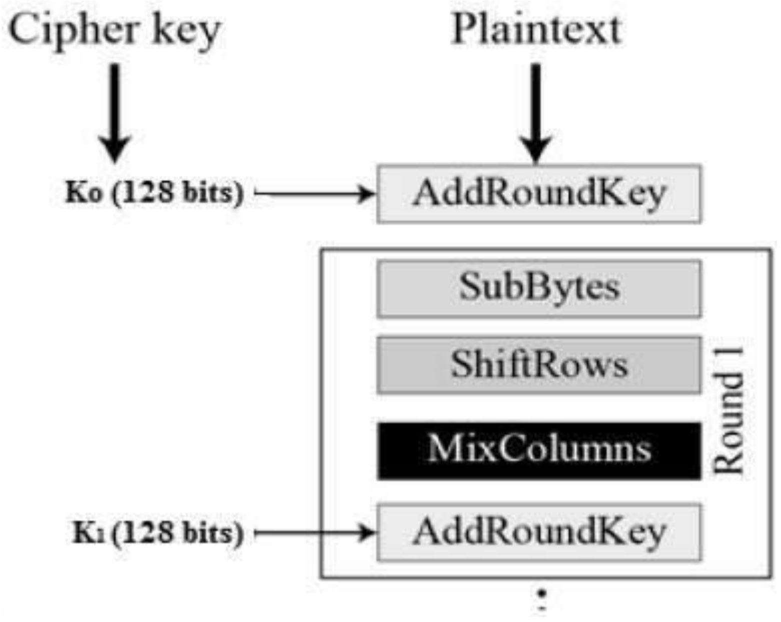
- AES is an iterative rather than Feistel cipher. It is based on 'substitution–permutation network'. It comprises of a series of linked operations, some of which involve replacing inputs by specific outputs (substitutions) and others involve shuffling bits around (permutations).
- Interestingly, AES performs all its computations on bytes rather than bits. Hence, AES treats the 128 bits of a plaintext block as 16 bytes. These 16 bytes are arranged in four columns and four rows for processing as a matrix −
- Unlike DES, the number of rounds in AES is variable and depends on the length of the key. AES uses 10 rounds for 128-bit keys, 12 rounds for 192-bit keys and 14 rounds for 256-bit keys. Each of these rounds uses a different 128-bit round key, which is calculated from the original AES key.
- The schematic of AES structure is given in the following illustration −


128-bit plaintext
Round keys (128 bits)
Pre-round transformation
K0
Cipher key (128, 192, or 256 bits)
Round 1
K1
Key expansion
Round 2
K2
Round Nr (slightly different)
KR
128-bit ciphertext
Relationship between number of rounds(R) and cipher key size

R	Key size
10	128
12	192
14	256

ENCRYPTION PROCESS

- Here, we restrict to description of a typical round of AES encryption. Each round comprise of four sub-processes. The first round process is depicted below −



BYTE SUBSTITUTION (SUBBYTES)

- The 16 input bytes are substituted by looking up a fixed table (S-box) given in design. The result is in a matrix of four rows and four columns.

SHIFTROWS

- Each of the four rows of the matrix is shifted to the left. Any entries that 'fall off' are re-inserted on the right side of row. Shift is carried out as follows −
    - First row is not shifted.
    - Second row is shifted one (byte) position to the left.
    - Third row is shifted two positions to the left.
    - Fourth row is shifted three positions to the left.
- The result is a new matrix consisting of the same 16 bytes but shifted with respect to each other.

MIXCOLUMNS

- Each column of four bytes is now transformed using a special mathematical function. This function takes as input the four bytes of one column and outputs four completely new bytes, which replace the original column. The result is another new matrix consisting of 16 new bytes. It should be noted that this step is not performed in the last round.

ADD ROUND KEY

- The 16 bytes of the matrix are now considered as 128 bits and are XORed to the 128 bits of the round key. If this is the last round then the output is the ciphertext. Otherwise, the resulting 128 bits are interpreted as 16 bytes and we begin another similar round.

DECRYPTION PROCESS

- The process of decryption of an AES ciphertext is similar to the encryption process in the reverse order. Each round consists of the four processes conducted in the reverse order −
    - Add round key
    - Mix columns
    - Shift rows
    - Byte substitution
- Since sub-processes in each round are in reverse manner, unlike for a Feistel Cipher, the encryption and decryption algorithms needs to be separately implemented, although they are very closely related.

PROGRAM

```
package com.includehelp.stringsample;
import java.util.Base64;
import java.util.Scanner;
import javax.crypto.Cipher;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;
/**
* Program to Encrypt/Decrypt String Using AES 128 bit Encryption Algorithm
*/
public class EncryptDecryptString {
private static final String encryptionKey           = "ABCDEFGHIJKLMNOP";
private static final String characterEncoding       = "UTF-8";
private static final String cipherTransformation    = "AES/CBC/PKCS5PADDING";
private static final String aesEncryptionAlgorithem = "AES";
/**
* Method for Encrypt Plain String Data
* @param plainText
* @return encryptedText
*/
public static String encrypt(String plainText) {
String encryptedText = "";
try {
Cipher cipher   = Cipher.getInstance(cipherTransformation);
byte[] key      = encryptionKey.getBytes(characterEncoding);
SecretKeySpec secretKey = new SecretKeySpec(key, aesEncryptionAlgorithem);
IvParameterSpec ivparameterspec = new IvParameterSpec(key);
cipher.init(Cipher.ENCRYPT_MODE, secretKey, ivparameterspec);
byte[] cipherText = cipher.doFinal(plainText.getBytes("UTF8"));
Base64.Encoder encoder = Base64.getEncoder();
encryptedText = encoder.encodeToString(cipherText);
} catch (Exception E) {
System.err.println("Encrypt Exception : "+E.getMessage());
}
return encryptedText;
}

/**
* Method For Get encryptedText and Decrypted provided String
* @param encryptedText
* @return decryptedText
*/
public static String decrypt(String encryptedText) {
```

```
String decryptedText = "";
try {
Cipher cipher = Cipher.getInstance(cipherTransformation);
byte[] key = encryptionKey.getBytes(characterEncoding);
SecretKeySpec secretKey = new SecretKeySpec(key, aesEncryptionAlgorithem);
IvParameterSpec ivparameterspec = new IvParameterSpec(key);
cipher.init(Cipher.DECRYPT_MODE, secretKey, ivparameterspec);
Base64.Decoder decoder = Base64.getDecoder();
byte[] cipherText = decoder.decode(encryptedText.getBytes("UTF8"));
decryptedText = new String(cipher.doFinal(cipherText), "UTF-8");
} catch (Exception E) {
System.err.println("decrypt Exception : "+E.getMessage());
}
return decryptedText;
}
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
System.out.println("Enter String : ");
String plainString = sc.nextLine();
String encyptStr   = encrypt(plainString);
String decryptStr  = decrypt(encyptStr);
System.out.println("Plain   String  : "+plainString);
System.out.println("Encrypt String  : "+encyptStr);
System.out.println("Decrypt String  : "+decryptStr);

}
}
```

OUTPUT:

stdin

```
Enter String : Hello World
```

stdout

```
Plain   String  : Hello World
Encrypt String  : IMfL/ifkuvkZwG/v2bn6Bw==
Decrypt String  : Hello World
```

<u>RESULT:</u>

Thus the program for apply AES algorithm for practical applications was executed and verified successfully.

Ex. No. : 5 IMPLEMENT RSA ALGORITHM USING HTML AND JAVASCRIPT

AIM:

To Implement a program RSA Algorithm using Html And Javascript.

ALGORITHM DESCRIPTION:

GENERATION OF RSA KEY PAIR

- Each person or a party who desires to participate in communication using encryption needs to generate a pair of keys, namely public key and private key. The process followed in the generation of keys is described below − Generate the RSA modulus (n) Select two large primes, p and q.
- Calculate n=p*q. For strong unbreakable encryption, let n be a large number, typically a minimum of 512 bits. Find Derived Number (e) Number e must be greater than 1 and less than (p − 1)(q − 1).
- There must be no common factor for e and (p − 1)(q − 1) except for 1. In other words two numbers e and (p – 1)(q – 1) are coprime.

FORM THE PUBLIC KEY

- The pair of numbers (n, e) form the RSA public key and is made public. Interestingly, though n is part of the public key, difficulty in factorizing a large prime number ensures that attacker cannot find in finite time the two primes (p & q) used to obtain n. This is strength of RSA.

GENERATE THE PRIVATE KEY

- Private Key d is calculated from p, q, and e. For given n and e, there is unique number d.
- Number d is the inverse of e modulo (p - 1)(q – 1). This means that d is the number less than (p - 1)(q - 1) such that when multiplied by e, it is equal to 1 modulo (p - 1)(q - 1).
- This relationship is written mathematically as follows ed = 1 mod (p − 1)(q − 1)
- The Extended Euclidean Algorithm takes p, q, and e as input and gives d as output.

PROGRAM:

```
<!doctype html>

<html>

  <head>

    <title>JavaScript RSA Encryption</title>

    <script src="http://code.jquery.com/jquery-1.8.3.min.js"></script>

    <script src="bin/jsencrypt.min.js"></script>

    <script type="text/javascript">



      // Call this code when the page is done loading.

      $(function() {



        // Run a quick encryption/decryption when they click.
        $('#testme').click(function() {

          // Encrypt with the public key...
          var encrypt = new JSEncrypt();
          encrypt.setPublicKey($('#pubkey').val());
          var encrypted = encrypt.encrypt($('#input').val());

          // Decrypt with the private key...
          var decrypt = new JSEncrypt();
          decrypt.setPrivateKey($('#privkey').val());
          var uncrypted = decrypt.decrypt(encrypted);

          // Now a simple check to see if the round-trip worked.
          if (uncrypted == $('#input').val()) {
            alert('It works!!!');
          }
          else {
            alert('Something went wrong....');
          }
        });
      });
    </script>
  </head>
  <body>
    <label for="privkey">Private Key</label><br/>
    <textarea id="privkey" rows="15" cols="65">


-----BEGIN RSA PRIVATE KEY-----
```

```
MIICXQIBAAKBgQDlOJu6TyygqxfWT7eLtGDwajtNFOb9I5XRb6khyfD1Yt3YiCgQ
WMNW649887VGJiGr/L5i2osbl8C9+WJTeucF+S76xFxdU6jE0NQ+Z+zEdhUTooNR
aY5nZiu5PgDB0ED/ZKBUSLKL7eibMxZtMlUDHjm4gwQco1KRMDSmXSMkDwIDAQAB
AoGAfY9LpnuWK5Bs50UVep5c93SJdUi82u7yMx4iHFMc/Z2hfenfYEzu+57fI4fv
xTQ//5DbzRR/XKb8ulNv6+CHyPF31xk7YOBfkGI8qjLoq06V+FyBfDSwL8KbLyeH
m7KUZnLNQbk8yGLzB3iYKkRHlmUanQGaNMIJziWOkN+N9dECQQD0ONYRNZeuM8zd
8XJTSdcIX4a3gy3GGCJxOzv16XHxD03GW6UNLmfPwenKu+cdrQeaqEixrCejXdAF
z/7+BSMpAkEA8EaSOeP5Xr3ZrbiKzi6TGMwHMvC7HdJxaBJbVRfApFrE0/mPwmP5
rN7QwjrMY+0+AbXcm8mRQyQ1+IGEembsdwJBAN6az8Rv7QnD/YBvi52POIlRSSIM
V7SwWvSK4WSMnGb1ZBbhgdg57DXaspcwHsFV7hByQ5BvMtIduHcT14ECfcECQATe
aTgjFnqE/lQ22Rk0eGaYO80cc643BXVGafNfd9fcvwBMnk0iGX0XRsOozVt5Azil
psLBYuApa66NcVHJpCECQQDTjI2AQhFc1yRnCU/YgDnSpJVm1nASoRUnU8Jfm3Oz
uku7JUXcVpt08DFSceCEX9unCuMcT72rAQlLpdZir876
-----END RSA PRIVATE KEY-----</textarea><br/>
    <label for="pubkey">Public Key</label><br/>
    <textarea id="pubkey" rows="15" cols="65">
-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDlOJu6TyygqxfWT7eLtGDwajtN
FOb9I5XRb6khyfD1Yt3YiCgQWMNW649887VGJiGr/L5i2osbl8C9+WJTeucF+S76
xFxdU6jE0NQ+Z+zEdhUTooNRaY5nZiu5PgDB0ED/ZKBUSLKL7eibMxZtMlUDHjm4
gwQco1KRMDSmXSMkDwIDAQAB
-----END PUBLIC KEY-----
</textarea><br/>
    <label for="input">Text to encrypt:</label><br/>
    <textarea id="input" name="input" type="text" rows=4 cols=70>This is a test!</textarea><br/>
    <input id="testme" type="button" value="Test Me!!!" /><br/>
  </body>
</html>
```

OUTPUT:

PRIVATE KEY

```
-----BEGIN RSA PRIVATE KEY-----
MIICXQIBAAKBgQDlOJu6TyygqxfWT7eLtGDwajtNFOb9I5XRb6khyfD1Yt3YiCgQWMNW
649887VGJiGr/L5i2osbl8C9+WJTeucF+S76xFxdU6jE0NQ+Z+zEdhUTooNRaY5nZiu5PgDB0
ED/ZKBUSLKL7eibMxZtMlUDHjm4gwQco1KRMDSmXSMkDwIDAQABAoGAf9LpnuWK
5Bs50UVep5c93SJdUi82u7yMx4iHFMc/Z2hfenfYEzu+57fI4fvxTQ//5DbzRR/XKb8ulNv6+Hy
PF31xk7YOBfkGI8qjLoq06V+FyBfDSwL8KbLyeHm7KUZnLNQbk8yGLzB3iYKkRHlmUan
QGaNMIJziWOkN+N9dECQQD0ONYRNZeuM8zd8XJTSdcIX4a3gy3GGCJxOzv16XHxD03
GW6UNLmfPwenKu+cdrQeaqEixrCejXdAFz/7+BSMpAkEA8EaSOeP5Xr3ZrbiKzi6TGMwH
MvC7HdJxaBJbVRfApFrE0/mPwmP5rN7QwjrMY+0+AbXcm8mRQyQ1+IGEembsdwJBAN6
az8Rv7QnD/YBvi52POIlRSSIMV7SwWvSK4WSMnGb1ZBbhgdg57DXaspcwHsFV7hBy5Bv
MtIduHcT14ECfcECQATeaTgjFnqE/lQ22Rk0eGaYO80cc643BXVGafNfd9fcvwBMnk0iGX0
XRsOozVt5AzilpsLBYuApa66NcVHJpCECQQDTjI2AQhFc1yRnCU/YgDnSpJVm1nASoRUn
U8Jfm3Ozuku7JUXcVpt08DFSceCEX9unCuMcT72rAQlLpdZir876

-----END RSA PRIVATE KEY-----
```

PUBLIC KEY

```
-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDlOJu6TyygqxfWT7eLtGDwajtNF
Ob9I5XRb6khyfD1Yt3YiCgQWMNW649887VGJiGr/L5i2osbl8C9+WJTeucF+S76xFxdU6jE0
NQ+Z+zEdhUTooNRaY5nZiu5PgDB0ED/ZKBUSLKL7eibMxZtMlUDHjm4gwQco1KRMDS
mXSMkDwIDAQAB

-----END PUBLIC KEY-----
```

RESULT:

Thus the program for RSA Algorithm using Html And Javascript was executed and verified successfully.

Ex. No. : 6 IMPLEMENT THE DIFFIE-HELLMAN KEY EXCHANGE ALGORITHM FOR A GIVEN PROBLEM.

AIM:

To develop a program to implement diffie-hellman key exchange algorithm for encryption and decryption.

ALGORITHM DESCRIPTION:

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

GLOBAL PUBLIC ELEMENTS:

Let q be a prime number and □ where □< q and □ is a primitive root of q.

1. User A Key Generation:
   Select private X_A where $X_A < q$
   Calculate public Y_A where Y_A = $□^{XA}$ mod q
2. User B Key Generation:
   Select private X_B where $X_B < q$
   Calculate public Y_B where Y_B = $□^{XB}$ mod q
3. Calculation of Secret Key by User A
   $K = (Y_B)^{XA}$ mod q

4.Calculation of Secret Key by User B:

PROGRAM:

```
package diffiehellmanbigint;

import java.util.*;

import java.math.BigInteger;

public class Main {

final static BigInteger one = new BigInteger("1");

public static void main(String args[]) {

Scanner stdin = new Scanner(System.in);

BigInteger p;

// Get a start spot to pick a prime from the user.

System.out.println("Enter the approximate value of p you want.");
```

```
String ans = stdin.next();
p = getNextPrime(ans);
System.out.println("Your prime is "+p+".");
// Get the base for exponentiation from the user.
System.out.println("Now, enter a number in between 2 and p-1.");
BigInteger g = new BigInteger(stdin.next());
// Get A's secret number.
System.out.println("Person A: enter your secret number now.");
BigInteger a = new BigInteger(stdin.next());
// Make A's calculation.
BigInteger resulta = g.modPow(a,p);
// This is the value that will get sent from A to B.
// This value does NOT compromise the value of a easily.
System.out.println("Person A sends to person B "+resulta+".");
// Get B's secret number.
System.out.println("Person B: enter your secret number now.");
BigInteger b = new BigInteger(stdin.next());
// Make B's calculation.
BigInteger resultb = g.modPow(b,p);
// This is the value that will get sent from B to A.
// This value does NOT compromise the value of b easily.
System.out.println("Person B sends to person A "+resultb+".");
// Once A and B receive their values, they make their new calculations.
// This involved getting their new numbers and raising them to the
// same power as before, their secret number.
BigInteger KeyACalculates = resultb.modPow(a,p);
BigInteger KeyBCalculates = resulta.modPow(b,p);
// Print out the Key A calculates.
System.out.println("A takes "+resultb+" raises it to the power "+a+" mod "+p);
System.out.println("The Key A calculates is "+KeyACalculates+".");
// Print out the Key B calculates.
System.out.println("B takes "+resulta+" raises it to the power "+b+" mod "+p);
System.out.println("The Key B calculates is "+KeyBCalculates+".");
}
public static BigInteger getNextPrime(String ans)
{
BigInteger test = new BigInteger(ans);
while (!test.isProbablePrime(99))
test = test.add(one);
return test;
}
}
```

OUTPUT:

```
Enter the approximate value of p you want.
7
Your prime is 7.
Now, enter a number in between 2 and p-1.
5
Person A: enter your secret number now.
4
Person A sends to person B 2.
Person B: enter your secret number now.
5
Person B sends to person A 3.
A takes 3 raises it to the power 4 mod 7
The Key A calculates is 4.
B takes 2 raises it to the power 5 mod 7
The Key B calculates is 4.
```

RESULT:

Thus the program for Diffie-Hellman key exchange algorithm was executed and verified successfully.

Ex. No. : 7 CALCULATE THE MESSAGE DIGEST OF A TEXT USING THE SHA-1 ALGORITHM

AIM:

To develop a program to calculate the message digest of a text using the sha-1 algorithm.

ALGORITHM DESCRIPTION:

SHA1 ALGORITHM CONSISTS OF 6 TASKS:

TASK 1. Appending Padding Bits. The original message is "padded" (extended) so that its length (in bits) is congruent to 448, modulo 512.

The padding rules are:

- The original message is always padded with one bit "1" first.
- Then zero or more bits "0" are padded to bring the length of the message up to 64 bits fewer than a multiple of 512.

TASK 2. Appending Length. 64 bits are appended to the end of the padded message to indicate the length of the original message in bytes. The rules of appending length are:

- The length of the original message in bytes is converted to its binary format of 64 bits. If overflow happens, only the low-order 64 bits are used.
- Break the 64-bit length into 2 words (32 bits each).
- The low-order word is appended first and followed by the high-order word.

TASK 3. Preparing Processing Functions. SHA1 requires 80 processing functions defined as:

```
f(t;B,C,D) = (B AND C) OR ((NOT B) AND D)            ( 0 <= t <= 19)
f(t;B,C,D) = B XOR C XOR D                           (20 <= t <= 39)
f(t;B,C,D) = (B AND C) OR (B AND D) OR (C AND D)     (40 <= t <= 59)
f(t;B,C,D) = B XOR C XOR D                           (60 <= t <= 79)
```

TASK 4. Preparing Processing Constants. SHA1 requires 80 processing constant words defined as:

```
K(t) = 0x5A827999     ( 0 <= t <= 19)
K(t) = 0x6ED9EBA1     (20 <= t <= 39)
K(t) = 0x8F1BBCDC     (40 <= t <= 59)
K(t) = 0xCA62C1D6     (60 <= t <= 79)
```

TASK 5. Initializing Buffers. SHA1 algorithm requires 5 word buffers with the following initial values:

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

TASK 6. Processing Message in 512-bit Blocks. This is the main task of SHA1 algorithm, which loops through the padded and appended message in blocks of 512 bits each. For each input block, a number of operations are performed. This task can be described in the following pseudo code slightly modified from the RFC 3174's method 1:

Input and predefined functions:

```
  M[1, 2, ..., N]: Blocks of the padded and appended message
  f(0;B,C,D), f(1,B,C,D), ..., f(79,B,C,D): Defined as above
  K(0), K(1), ..., K(79): Defined as above
  H0, H1, H2, H3, H4, H5: Word buffers with initial values
  For loop on k = 1 to N
 (W(0),W(1),...,W(15)) = M[k] /* Divide M[k] into 16 words */
  For t = 16 to 79 do:
 W(t) = (W(t-3) XOR W(t-8) XOR W(t-14) XOR W(t-16)) <<<
A = H0, B = H1, C = H2, D = H3, E = H4
```

PROGRAM:

```
import java.security.*;
class Main

{

public static void main(String[] a)

{

Try

 {

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

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

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

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

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

String input = "";
md.update(input.getBytes());
byte[] output = md.digest();

System.out.println();

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

System.out.println("   "+bytesToHex(output));

input = "abc";

md.update(input.getBytes());

output = md.digest();

System.out.println();

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

System.out.println("   "+bytesToHex(output));

input = "abcdefghijklmnopqrstuvwxyz";

md.update(input.getBytes());

output = md.digest();

System.out.println();

System.out.println("SHA1(\""+input+"\") =");
System.out.println("   "+bytesToHex(output));
} catch (Exception e)
 {
System.out.println("Exception: "+e);
}
}
public static String bytesToHex(byte[] b)
{
char hexDigit[] = {'0', '1', '2', '3', '4', '5', '6', '7',
 '8', '9', 'A', 'B', 'C', 'D', 'E', 'F'};
StringBuffer buf = new StringBuffer();
for (int j=0; j<b.length; j++)
{
buf.append(hexDigit[(b[j] >> 4) & 0x0f]);
buf.append(hexDigit[b[j] & 0x0f]);
}
return buf.toString();
}
}
```

OUTPUT:

```
Message digest object info:
 Algorithm = SHA1
 Provider = SUN version 1.6
 toString = SHA1 Message Digest from SUN, <initialized>
SHA1("") = DA39A3EE5E6B4B0D3255BFEF95601890AFD80709
SHA1("abc") = A9993E364706816ABA3E25717850C26C9CD0D89D
SHA1("abcdefghijklmnopqrstuvwxyz") = 32D10C7B8CF96570CA04CE37F2A19D84240D3A89.
```

RESULT:

Thus the program for calculate the message digest of a text using the SHA-1algorithm was executed and verified successful.

Ex. No. : 8 IMPLEMENT THE SIGNATURE SCHEME DIGITAL SIGNATURE STANDARD

AIM:

To develop a program to implement the signature scheme – digital signature.

ALGORITHM DESCRIPTION:

GENERATING A DIGITAL SIGNATURE

The GenSig program you are about to create will use the JDK Security API to generate keys and a digital signature for data using the private key and to export the public key and the signature to files. The application gets the data file name from the command line.

The following steps create the GenSig sample program.

1. Prepare Initial Program Structure
   Create a text file named GenSig.java. Type in the initial program structure (import statements, class name, main method, and so on).
2. Generate Public and Private Keys
   Generate a key pair (public key and private key). The private key is needed for signing the data. The public key will be used by the VerSig program for verifying the signature.
3. Sign the Data
   Get a Signature object and initialize it for signing. Supply it with the data to be signed, and generate the signature.
4. Save the Signature and the Public Key in Files
   Save the signature bytes in one file and the public key bytes in another.
5. Compile and Run the Program.

PROGRAM:

```
package javaapplication;

import java.io.BufferedInputStream;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

import java.security.InvalidKeyException;

import java.security.KeyFactory;

import java.security.KeyPair;

import java.security.KeyPairGenerator;

import java.security.NoSuchAlgorithmException;

import java.security.NoSuchProviderException;

import java.security.PrivateKey;

import java.security.PublicKey;

import java.security.SecureRandom;
import java.security.Signature;
import java.security.SignatureException;
import java.security.spec.InvalidKeySpecException;
import java.security.spec.X509EncodedKeySpec;
public class DigitalSignatureHandler {
private final String ENCRYPTION_ALGORITHM = "DSA";
private final String PROVIDER = "SUN";
private final String SIGNATURE_ALGORITHM = "SHA1withDSA";
private final int KEY_SIZE = 1024;
private final String DOC_TO_BE_SIGNED = "info.txt";
private final String SIGNATURE_FILE = "info.signature.sha1.dsa";
private final String PUBLIC_KEY_FILE = "info.public.key";
private PrivateKey privateKey;
private PublicKey publicKey;
private void generateKeys()
{
try
{
KeyPairGenerator keyGen = KeyPairGenerator.getInstance(ENCRYPTION_ALGORITHM,
PROVIDER);
keyGen.initialize(KEY_SIZE, SecureRandom.getInstance("SHA1PRNG", PROVIDER));
KeyPair keyPair = keyGen.generateKeyPair();
privateKey = keyPair.getPrivate();
publicKey = keyPair.getPublic();
savePublicKey();
}
catch (NoSuchAlgorithmException | NoSuchProviderException e) {
System.err.print("Either the algorithm or the provider is wrong!");
e.printStackTrace();
}
}
private void savePublicKey()
 {
FileOutputStream keyfos = null;
Try
 {
keyfos = new FileOutputStream(PUBLIC_KEY_FILE);
keyfos.write(publicKey.getEncoded());
}
 catch (IOException e)
 {
System.err.println("File not found/could not write - " + e.getMessage());
e.printStackTrace();
}
finally
{
if (keyfos != null)
{
try {
keyfos.close();
} catch (IOException e)
{
System.err.println("File not found/could not be closed - " + e.getMessage());
e.printStackTrace();
}
}
}
}
```

```
private void readPublicKey()
 {
FileInputStream keyfis = null;
try
{
keyfis = new FileInputStream(PUBLIC_KEY_FILE);
byte[] encKey = new byte[keyfis.available()];
keyfis.read(encKey);
X509EncodedKeySpec pubKeySpec = new X509EncodedKeySpec(encKey);
KeyFactory keyFactory = KeyFactory.getInstance(ENCRYPTION_ALGORITHM, PROVIDER);
publicKey = keyFactory.generatePublic(pubKeySpec);
}
catch (IOException e)
{
System.err.println("File not found/could not read - " + e.getMessage());
e.printStackTrace();
}
catch (NoSuchAlgorithmException | NoSuchProviderException e)
 {
System.err.print("Either the algorithm or the provider is wrong!");
e.printStackTrace();
}
catch (InvalidKeySpecException e) {
System.err.println("Invlaid Key Spec - " + e.getMessage());
e.printStackTrace();
}
 finally
{
if (keyfis != null)
```

```
{
Try
 {
keyfis.close();
}
catch (IOException e)
{
System.err.println("File not found/could not be closed - " + e.getMessage());
e.printStackTrace();
}
}
}
}
private void updateSignature(Signature signature)
{
BufferedInputStream bufin = null;
Try
 {
FileInputStream fis = new FileInputStream(new File(DOC_TO_BE_SIGNED));
bufin = new BufferedInputStream(fis);
byte[] buffer = new byte[1024];
int len;
while ((len = bufin.read(buffer)) >= 0)
 {
signature.update(buffer, 0, len);
}
}
catch (SignatureException e)
 {
```

```
System.err.println("Signature Error - " + e.getMessage());
e.printStackTrace();
}
catch (IOException e)
{
System.err.println("Document not found/could not be read - " + e.getMessage());
e.printStackTrace();
}
finally
 {
if (bufin != null)
{
Try
 {
bufin.close();
}
catch (IOException e)
{
System.err.println("Document not found/could not be closed - " + e.getMessage());
e.printStackTrace();
}
}
}
}
private void saveSignature(byte[] signature) {
FileOutputStream sigfos = null;
try
{
sigfos = new FileOutputStream(SIGNATURE_FILE);
```

```
sigfos.write(signature);
}
 catch (IOException e)
{
System.err.println("File not found/could not write - " + e.getMessage());
e.printStackTrace();
}
finally
{
if (sigfos != null)
 {
Try
 {
sigfos.close();
}
 catch (IOException e)
{
System.err.println("File not found/could not be closed - " + e.getMessage());
e.printStackTrace();
}
}
}
}
private byte[] readSignature() {
FileInputStream sigfis = null;
try
{
sigfis = new FileInputStream(SIGNATURE_FILE);
byte[] signature = new byte[sigfis.available()];
```

```
sigfis.read(signature);
return signature;
}
 catch (IOException e)
{
System.err.println("Signature File not found/could not read - " + e.getMessage());
e.printStackTrace();
}
 finally
{
if (sigfis != null)
 {
try
{
sigfis.close();
}
catch (IOException e)
{
System.err.println("Signature File not found/could not be closed - " + e.getMessage());
e.printStackTrace();
}
}
}
return null;
}
public void signDocument()
{
generateKeys();
BufferedInputStream bufin = null;
```

```
try
{
Signature signature = Signature.getInstance(SIGNATURE_ALGORITHM, PROVIDER);
signature.initSign(privateKey);
updateSignature(signature);
byte[] sign = signature.sign();
if (sign != null)
{
saveSignature(sign);
}
}
catch (NoSuchAlgorithmException | NoSuchProviderException e)
{
System.err.println("Either the algorithm or the provider is wrong!");
e.printStackTrace();
}
catch (SignatureException e)
{
System.err.println("Signature Error - " + e.getMessage());
e.printStackTrace();
}
catch (InvalidKeyException e)
{
System.err.println("Invlaid Key - " + e.getMessage());
e.printStackTrace();
}
 Finally
 {
if (bufin != null)
```

```
 {
Try
 {
bufin.close();
}
 catch (IOException e)
{
System.err.println("Document not found/could not be closed - " + e.getMessage());
e.printStackTrace();
}
}
}
}
public void verifyDocument()
{
readPublicKey();
byte[] sign = readSignature();
if (sign != null)
{
Signature signature;
try
{
signature = Signature.getInstance(SIGNATURE_ALGORITHM, PROVIDER);
signature.initVerify(publicKey);
updateSignature(signature);
boolean result = signature.verify(sign);
if (result)
{
System.out.println("Signature Verification is Successful!");
}
else
{
System.out.println("Signature Verification is Failed");
}
}
catch (NoSuchAlgorithmException | NoSuchProviderException e)
{
System.err.println("Either the algorithm or the provider is wrong!");
e.printStackTrace();
}
catch (SignatureException e)
{
System.err.println("Signature Error - " + e.getMessage());
e.printStackTrace();
}
catch (InvalidKeyException e)
 {
System.err.println("Invlaid Key - " + e.getMessage());
e.printStackTrace();
}
}
}
public static void main(String[] args)
 {
DigitalSignatureHandler signatureHandler = new DigitalSignatureHandler();
//signatureHandler.signDocument();
signatureHandler.verifyDocument();
}
```

```
}
```

OUTPUT:

stdin:

```
Standard input is empty
```

stdout:

```
simulation of Digital Signature Algorithm global public key components are:
p is: 10601
q is: 53 g is: 3848 secret information are:
x (private) is: 48 k (secret)  is: 25 y
(public)  is: 5885 h (rndhash) is: 8794
generating digital signature: r is : 4 s is : 16
verifying digital signature (checkpoints):
w  is : 10
u1 is : 13
u2 is : 40
v  is : 4
success: digital signature is verified! 4
```

RESULT:

Thus the program to implement the signature scheme – digital signature was executed and verified successfully.

Ex. No. : 9 DEMONSTRATE INTRUSION DETECTION SYSTEM (IDS) USING ANY TOOL EXAMPLE. SNORT OR ANY OTHER SOFTWARE.

AIM:

To Demonstrate Intrusion Detection System (Ids) Using Snort Tool.

DESCRIPTION:

SNORT INSTALLATION STEPS

Getting and Installing Necessary Tools

Installaing Packages
Snort: <Snort_2_9_8_2_installer.exe>
WinPcap: <WinPcap_4_1_3.exe>
Snort rules: <snortrules-snapshot-2982.tar.gz>

Once Completed

1.Change the Snort program directory:
<c:\cd \Snort\bin
2.Check the installed version for Snort:
<c:\Snort\bin> snort -V
3.Check to see what network adapters are on your system
<c:\Snort\bin> snort -W>
Configure Snort with snort.conf
<snort.conf> is located in <c:\Snort\ect>

CONTAINS NINE STEPS:

1. Set the network variables

a. Change <HOME_NET> to your home network IP address range <10.6.2.1/24>
b. Change <EXTERNAL_NET> to <!$HOME_NET>
This expression means the external network will be defined as - any IP not part of home network
c. Check for <HTTP_PORTS>
d. Change var <RULE_PATH> - actual path of rule files. i.e<c:\Snort\rules>
e. Change var <PREPROC_RULE_PATH> - actual path of preprocessor rule files i.e<c:\Snort\preproc_rules>
f. Comment <#><SO_RULE_PATH> - as windows Snort doesn't use shared object rules
g. Configure trusted <white.list> and untrusted <black.list> IP address - reputation preprocessor

2. Configure the decoder

a. No changes in this part
b. Set the default directory for Snort logs i.e<c:\Snort\logs>

3. Configure the base detection engine

a. No changes in this part

4. Configure dynamic loaded libraries

a. Change the dynamic loaded library path references
i.e. <dynamicpreprocessdirec c:\Snort\lib\snort_dynamicpreprocessor>
i.e. <dynamicenginedirec c:\Snort\lib\snort_dynamicengine\sf_engine.dll>
b. Comment out <dynamicdetection directory> declaration

5. Configure preprocessors

a. Many Preprocessors are used by Snort - Check Snort manual before setting them.
b. Comment on <inline packet normalization preprocessor>

This preprocessor is used when Snort is in-line IPS mode>

c. For general purpose Snort usage - check these preprocessors are active frag3 stream5 http_inpectftp_telnet smtp dnsssl

sensitive_data

6. Confgiure output plugins

a. Be default Snort uses only one output plugings - <default:unified2>

b. Want to use Syslog output pluging - activate it by uncommenting.

1. Uncomment and edit the syslog output line

<output alert_syslog: host=127.0.0.1:514, LOG_AUTH LOG_ALERT> Note: If you are going to use syslog - install <Syslog Server> c. Uncomment metadata reference lines

<include classification.config and include reference.config>

7. Customise your rule set

a. Initial test, reduce the number of rules loaded at start-up, uncomment <local.rules>

b. First time users, comment most of include statements.

8. Customise preprocessor and decoder rule set

a. Uncomment the first two lines in Step 8

<include $PREPROC_RULE_PATH\preprocessor.rules>

<include $PREPROC_RULE_PATH\decoder.rules>

b. If you enables the sensitive_data preprocessor <step 5> uncomment

<include $PREPROC_RULE_PATH\sensitive-data.rules>

c. Make sure rules you declare - available in <c:\Snort\preproc_rules>

9. Customiseshart object rule set

a. Comment on lines

b. Uncomment <include threshold.conf>

Generating Alerts

This is for validation of Snort

1. Open <local.rules> in a text editor
2. Start typing this:

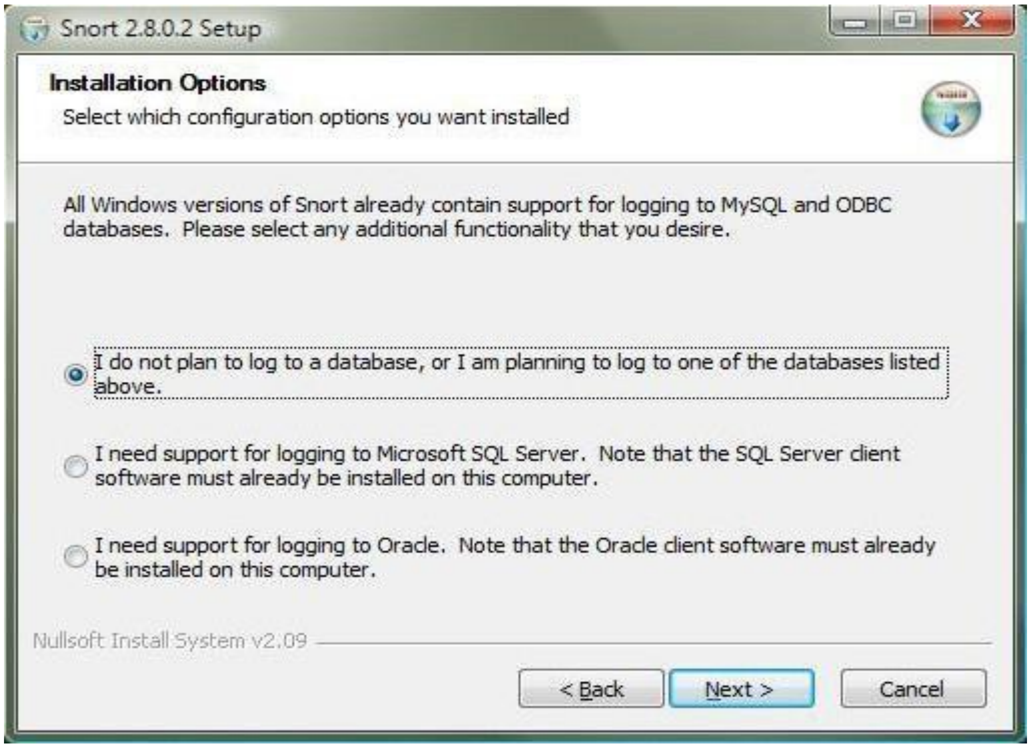
```
<alert icmp any any -> any any (msg:"ICMP Testing Rule"; sid:1000001; rev:1;)
<alert tcp any any -> any 80 (msg:"TCP Testing Rule"; sid:1000002; rev:1;)
<alert udp any any -> any any (msg:"UDP Testing Rule"; sid:1000003; rev:1;)
```

3. Save as <local.rules>
4. Open <CMD> and run it as <ADMINISTRATOR>
5. Start Snort <c:\Snort\bin> snort -i 2 -c c:\Snort\etc\snort.conf -A console
6. Open <CMD> no need to be an ADMINISTRATOR
7. Send a <PING> command to your local gateway: <c:\> ping 10.6.0.1>
8. Open a web browser and browse to any web page

You can see the alerts Snort produces and shows it in First terminal.

INSTALLATION OF SNORT IN WINDOWS:

- Download SNORT from snort.org
- Install snort with or without database support.


Snort 2.8.0.2 Setup

Installation Options

Select which configuration options you want installed

All Windows versions of Snort already contain support for logging to MySQL and ODBC databases. Please select any additional functionality that you desire.

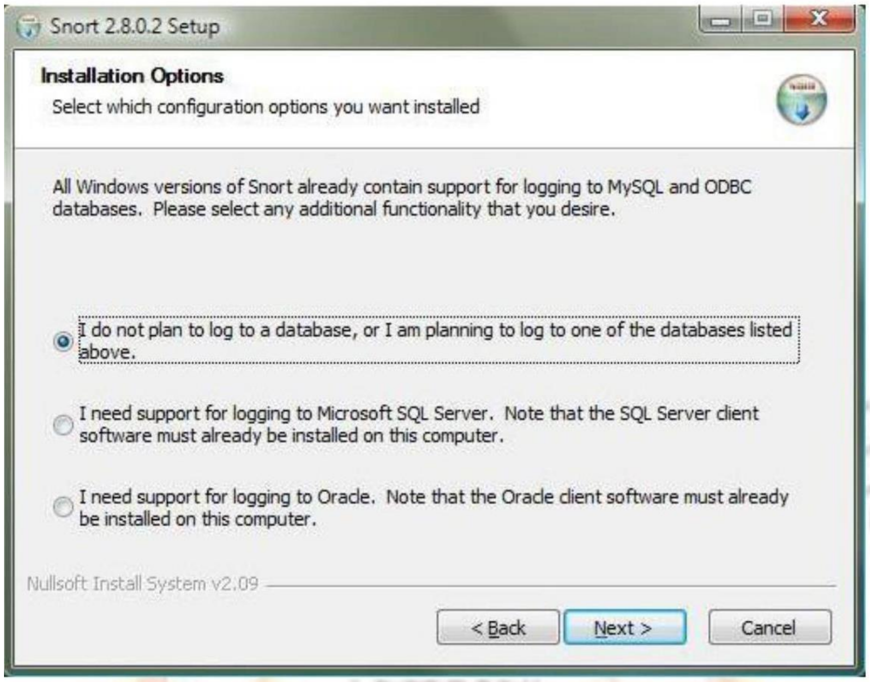
(●) I do not plan to log to a database, or I am planning to log to one of the databases listed above.

( ) I need support for logging to Microsoft SQL Server. Note that the SQL Server client software must already be installed on this computer.

( ) I need support for logging to Oracle. Note that the Oracle client software must already be installed on this computer.

Nullsoft Install System v2.09

- Select all the components and Click Next.
  Install and Close.
- Skip the WinPcap driver installation
- Add the path variable in windows environment variable by selecting new classpath. Create a path variable and point it at snort.exe variable name path and variable value c:\snort\bin.


Snort 2.8.0.2 Setup

Installation Options

Select which configuration options you want installed

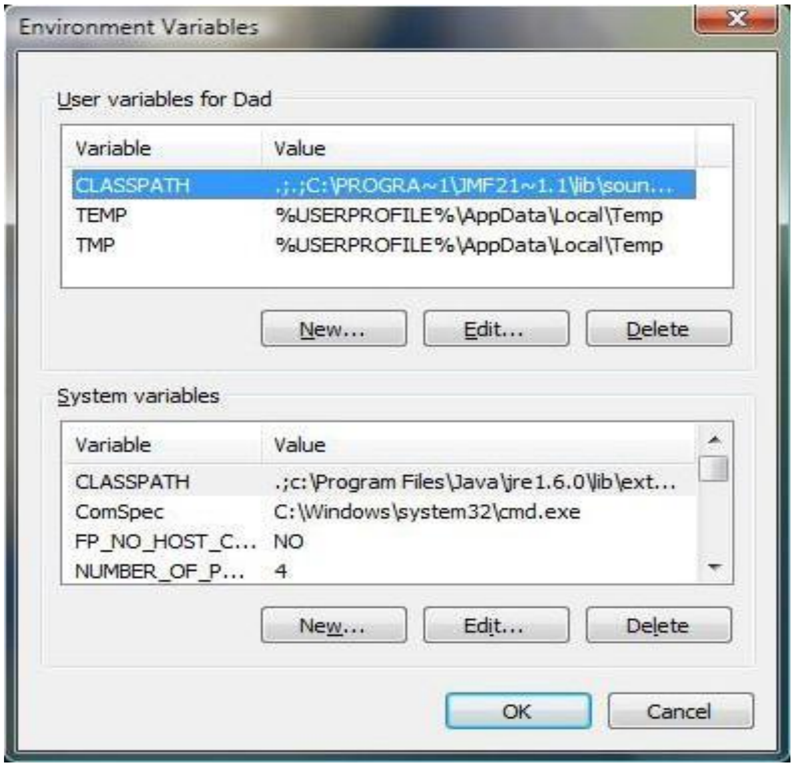
All Windows versions of Snort already contain support for logging to MySQL and ODBC databases. Please select any additional functionality that you desire.

(●) I do not plan to log to a database, or I am planning to log to one of the databases listed above.

( ) I need support for logging to Microsoft SQL Server. Note that the SQL Server client software must already be installed on this computer.

( ) I need support for logging to Oracle. Note that the Oracle client software must already be installed on this computer.

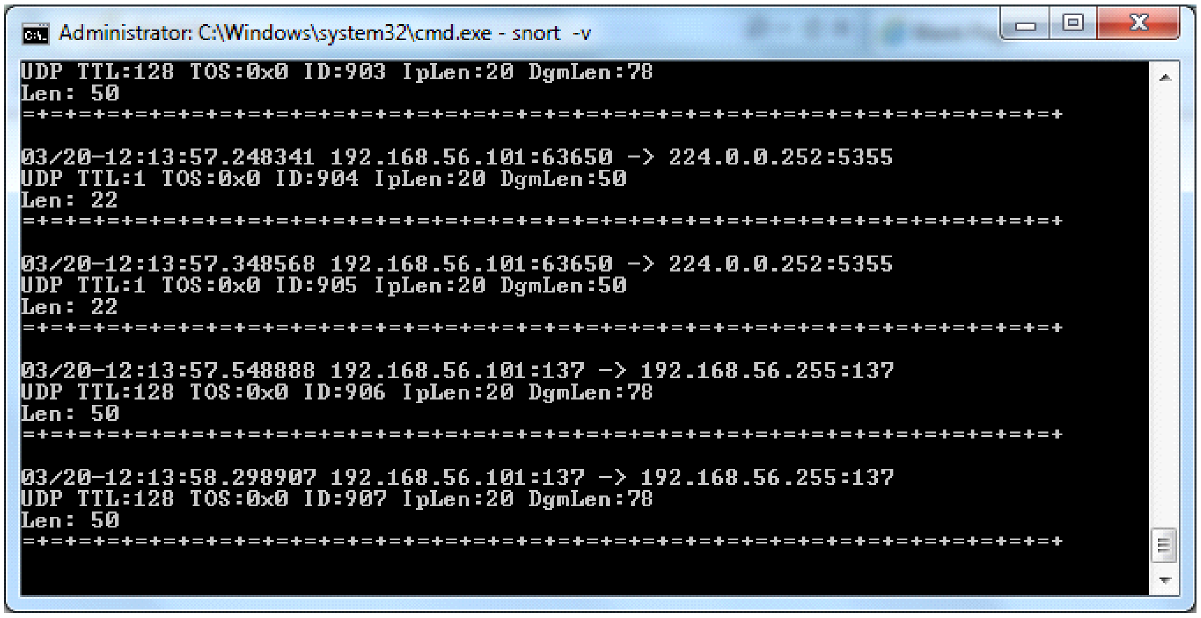
Nullsoft Install System v2.09



Variable	Value
CLASSPATH	.;.;C:\PROGRA~1\JMF21~1.1\lib\soun...
TEMP	%USERPROFILE%\AppData\Local\Temp
TMP	%USERPROFILE%\AppData\Local\Temp

Variable	Value
CLASSPATH	.;c:\Program Files\Java\jre1.6.0\lib\ext...
ComSpec	C:\Windows\system32\cmd.exe
FP_NO_HOST_C...	NO
NUMBER_OF_P...	4

- Click OK button and then close all dialog boxes.
- Open command prompt and type the following commands:



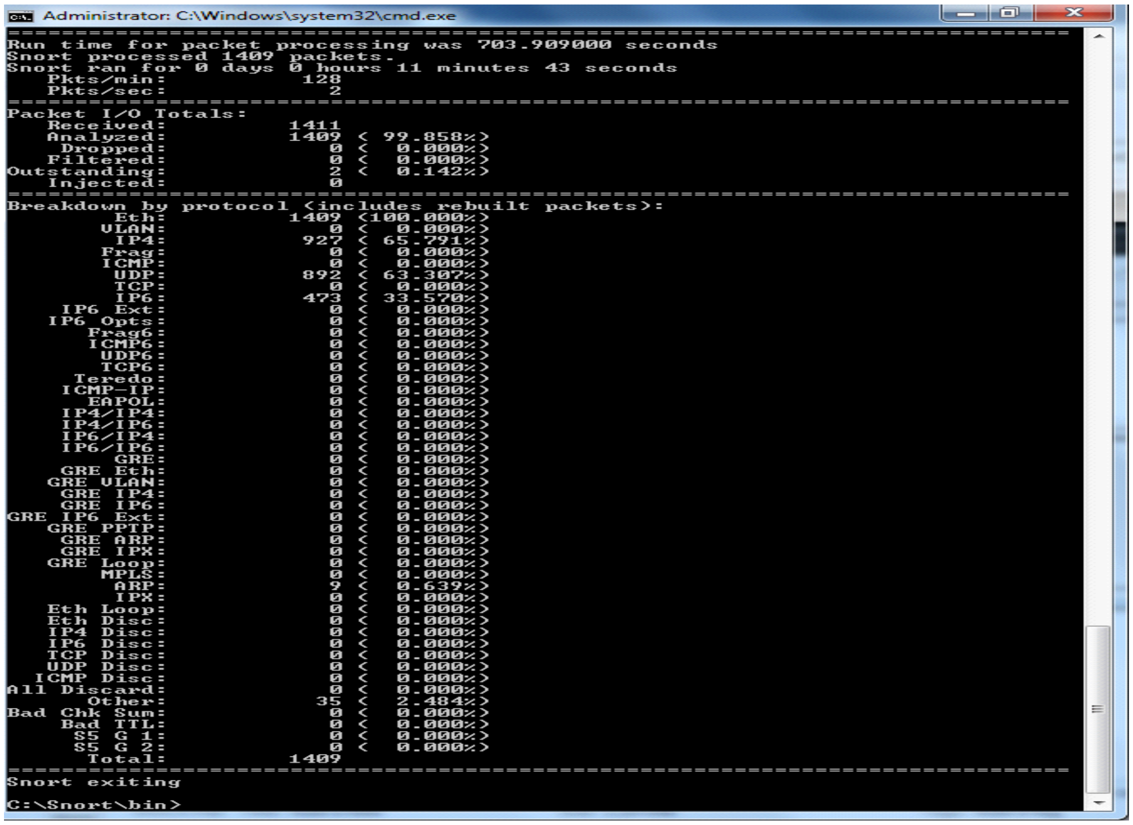
```
UDP TTL:128 TOS:0x0 ID:903 IpLen:20 DgmLen:78
Len: 50
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

03/20-12:13:57.248341 192.168.56.101:63650 -> 224.0.0.252:5355
UDP TTL:1 TOS:0x0 ID:904 IpLen:20 DgmLen:50
Len: 22
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

03/20-12:13:57.348568 192.168.56.101:63650 -> 224.0.0.252:5355
UDP TTL:1 TOS:0x0 ID:905 IpLen:20 DgmLen:50
Len: 22
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

03/20-12:13:57.548888 192.168.56.101:137 -> 192.168.56.255:137
UDP TTL:128 TOS:0x0 ID:906 IpLen:20 DgmLen:78
Len: 50
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

03/20-12:13:58.298907 192.168.56.101:137 -> 192.168.56.255:137
UDP TTL:128 TOS:0x0 ID:907 IpLen:20 DgmLen:78
Len: 50
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```



```
Administrator: C:\Windows\system32\cmd.exe
===============================================================================
Run time for packet processing was 703.909000 seconds
Snort processed 1409 packets.
Snort ran for 0 days 0 hours 11 minutes 43 seconds
   Pkts/min:          128
   Pkts/sec:            2
===============================================================================
Packet I/O Totals:
   Received:         1411
   Analyzed:         1409 ( 99.858%)
    Dropped:            0 (  0.000%)
   Filtered:            0 (  0.000%)
Outstanding:            2 (  0.142%)
   Injected:            0
===============================================================================
Breakdown by protocol (includes rebuilt packets):
        Eth:         1409 (100.000%)
       VLAN:            0 (  0.000%)
        IP4:          927 ( 65.791%)
       Frag:            0 (  0.000%)
       ICMP:            0 (  0.000%)
        UDP:          892 ( 63.307%)
        TCP:            0 (  0.000%)
        IP6:          473 ( 33.570%)
    IP6 Ext:            0 (  0.000%)
   IP6 Opts:            0 (  0.000%)
      Frag6:            0 (  0.000%)
      ICMP6:            0 (  0.000%)
       UDP6:            0 (  0.000%)
       TCP6:            0 (  0.000%)
     Teredo:            0 (  0.000%)
    ICMP-IP:            0 (  0.000%)
      EAPOL:            0 (  0.000%)
    IP4/IP4:            0 (  0.000%)
    IP4/IP6:            0 (  0.000%)
    IP6/IP4:            0 (  0.000%)
    IP6/IP6:            0 (  0.000%)
        GRE:            0 (  0.000%)
    GRE Eth:            0 (  0.000%)
   GRE VLAN:            0 (  0.000%)
    GRE IP4:            0 (  0.000%)
    GRE IP6:            0 (  0.000%)
GRE IP6 Ext:            0 (  0.000%)
   GRE PPTP:            0 (  0.000%)
    GRE ARP:            0 (  0.000%)
    GRE IPX:            0 (  0.000%)
   GRE Loop:            0 (  0.000%)
       MPLS:            0 (  0.000%)
        ARP:            9 (  0.639%)
        IPX:            0 (  0.000%)
   Eth Loop:            0 (  0.000%)
   Eth Disc:            0 (  0.000%)
   IP4 Disc:            0 (  0.000%)
   IP6 Disc:            0 (  0.000%)
   TCP Disc:            0 (  0.000%)
   UDP Disc:            0 (  0.000%)
  ICMP Disc:            0 (  0.000%)
All Discard:            0 (  0.000%)
      Other:           35 (  2.484%)
Bad Chk Sum:            0 (  0.000%)
    Bad TTL:            0 (  0.000%)
     S5 G 1:            0 (  0.000%)
     S5 G 2:            0 (  0.000%)
      Total:         1409
===============================================================================
Snort exiting

C:\Snort\bin>
```

RESULT:

Thus the Demonstrate Intrusion Detection System (Ids) Using Snort executed and verified successfully.

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

AIM:

To exploring a vulnerability assessment tool N-stalker,

DESCRIPTION:

VULNERABILITY ASSESSMENT TOOLS

Much has changed with regard to how we view vulnerability assessment software since its introduction in the early 1990s. At that time, two well-known security professionals, Dan Farmer and Wietse Venema, wrote a landmark paper entitled “Improving the Security of Your Site by Breaking Into It.” They went on to code the first automated penetration tool, known as SATAN (Security Administrator Tool for Analyzing Networks). Farmer was actually fired from his job at Sun Microsystems for developing this program.

Today, attack and penetration tools are viewed much differently. It is generally agreed that security professionals must look for vulnerabilities in their own networks and seek ways to mitigate the exposures they uncover. This brings up the question of what vulnerability assessment tools someone needs in their own network security lab. With so many tools available, where do you begin? I will start by looking at how these tools can be categorized. The three basic categories are as follows:

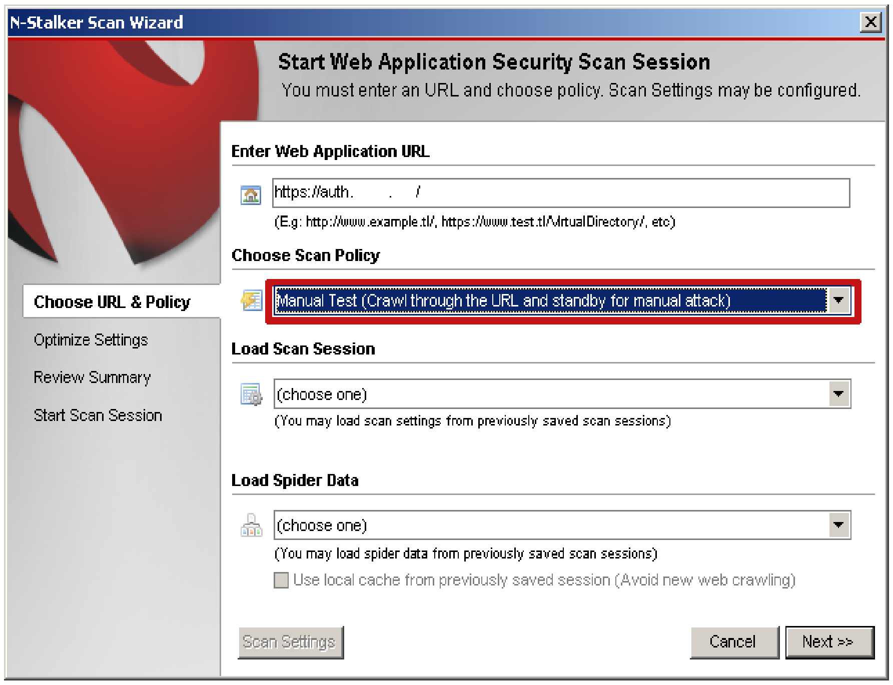
- Source code assessment tools examine the source code of an application.
- Application assessment tools examine a specific application or type of application.
- System assessment tools examine entire systems or networks for configuration or application-level problems.

Exploring N-Stalker, a Vulnerability Assessment Tool

The manual test is initiated by means of navigation through the application’s URLs with the Browser activated by N-Stalker’s internal Proxy which captures all requisitions and responses from the application (as in the configuration for recording of a navigation macro). Once the URLs you wish to analyze are captured, we can inform on how to use them for analysis, selecting and redirecting the attack mode.

1. The first stage is to know the application’s URL and the path you wish it to follow, and then choose the "Manual Test" policy to initiate the manual Scan manual, according to the steps below:

2. Type the URL of the web application and in "Choose Scan Policy", it is necessary to choose the "Manual Test” policy (crawl through the URL and standby for manual attack)".


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

https://auth.   .   /

(E.g: http://www.example.tl/, https://www.test.tl/VirtualDirectory/, etc)

Choose Scan Policy

Manual Test (Crawl through the URL and standby for manual attack)

Load Scan Session

(choose one)

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

Load Spider Data

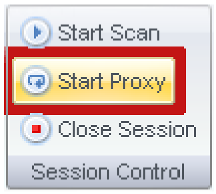
(choose one)

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

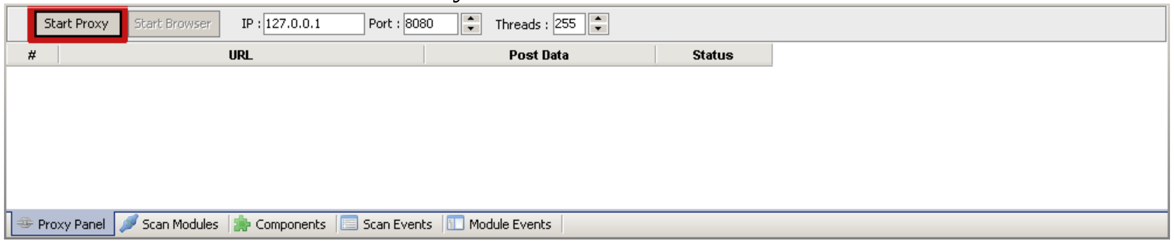
☐ Use local cache from previously saved session (Avoid new web crawling)

3. After finishing, please click on the "Next" button to proceed until reaching the "Review Summary" screen. If all is correct, press the "Start Session" button. The "Scanning Assistant" is closed and the "Scanning Interface Scanning" is open to initiate the manual analysis section.

4. In "Scan Options" click on the "Start Proxy" button. The Proxy will be iniatited.

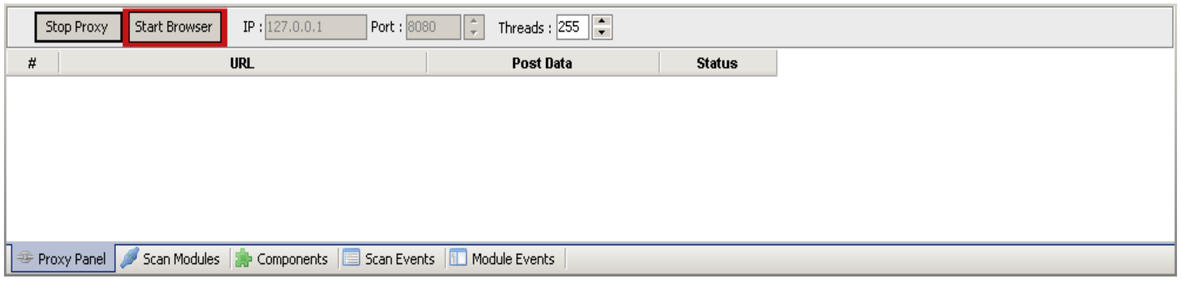


5. The Proxy options are available as an additional panel with the name "Proxy Panel", on the screen's lower corner. Just click on the "Start Proxy" button:

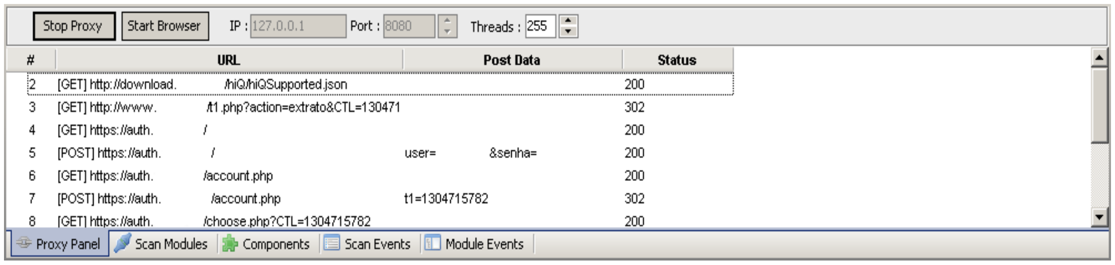

IP : 127.0.0.1 Port : 8080 Threads : 255

#	URL	Post Data	Status

6. After initiating, click on the "Start Browser" button. The N-Stalker will try to automatically configure your navigator for use inside the "Macro Recorder" tool. If you find problems doing so, please configure it manually (using the network connection of your browser).

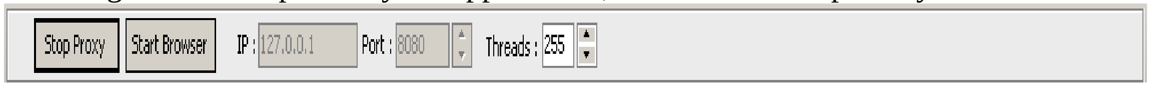


7. Start navigating throughout your application. Run through a path you would like the N-Stalker Spider engine would also follow. N-Stalker records all information within the "Proxy Panel" table:

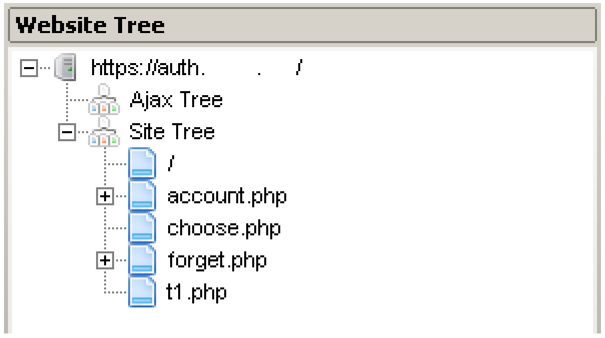


#	URL	Post Data	Status
2	[GET] http://download. /hiQ/hiQSupported.json		200
3	[GET] http://www. /t1.php?action=extrato&CTL=130471		302
4	[GET] https://auth. /		200
5	[POST] https://auth. /	user= &senha=	200
6	[GET] https://auth. /account.php		200
7	[POST] https://auth. /account.php	t1=1304715782	302
8	[GET] https://auth. /choose.php?CTL=1304715782		200

8. As soon as navigation is completed by the application , click on the "Stop Proxy" button.



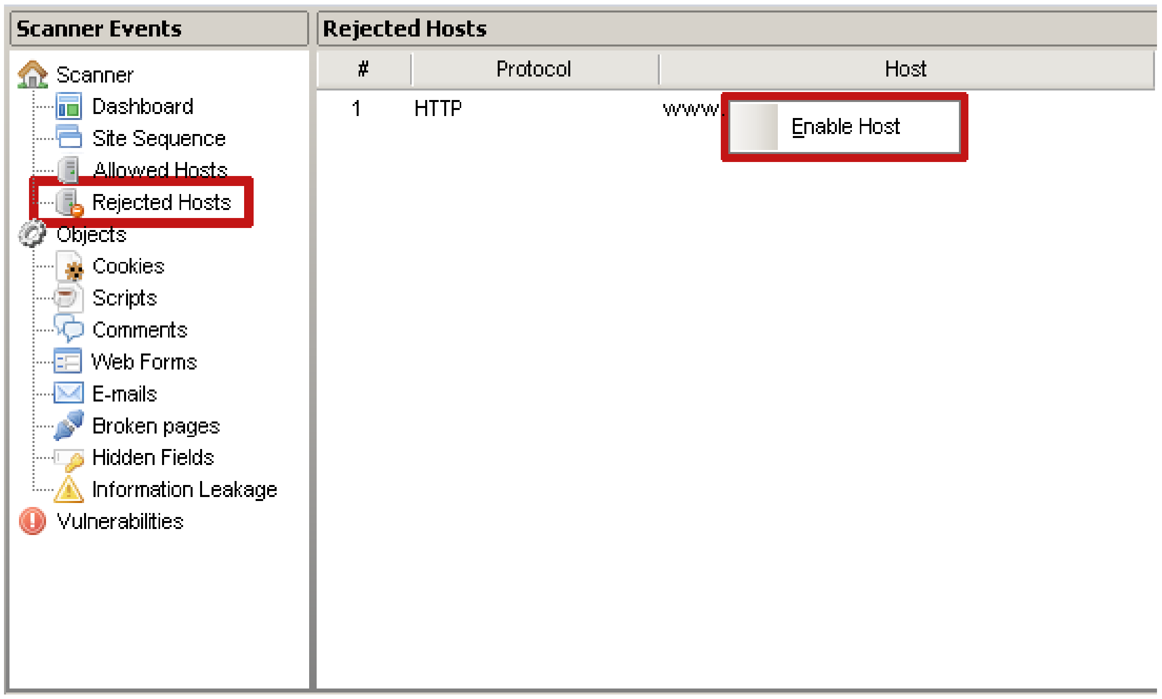
The “Website Tree” control allows user to visualize all the resources that were gathered during the trip of the N-Stalker spider/proxy engine, including web pages and transactions. The “Website Tree” is co-ordinated by web servers (URLs) which represent all the various hosts located on the crawler path (please remember that only the URL itself and the allowed hosts can be crawled through).



AUTHORIZING A REJECTED HOST

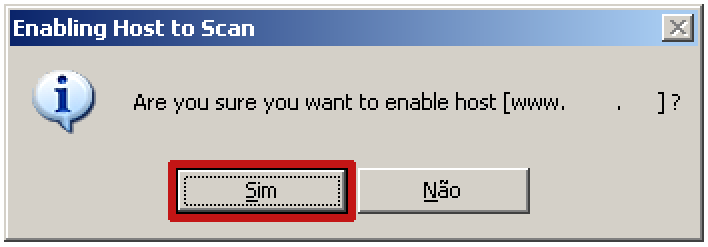
Sometimes the applications may refer to external web sites which are not relevant for the verification of global security, however, there are some situations where external web sites are part of the inspected application and thus must be assessed.

In the "Scanner Events" tree, on the "Rejected Hosts" option, it is possible to see a list of web sites that had been rejected by the N-Stalker spider engine due to configuration restrictions.



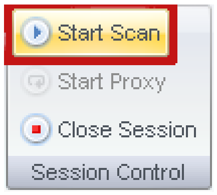
#	Protocol	Host
1	HTTP	www

7. In case you need to explicitly authorize a host from the list, please right click on it, then click on "Enable Host" and confirm by clicking on the "Yes" button.



Are you sure you want to enable host [www.   .   ]?

8. After navigating throughout your application and having accomplished the desired configurations, click on the "Start Scan" button to initiate the scanning engine.



Then a screen with 3 options related to the data stored by the proxy is displayed. Please select the option of your choice in case you wish to initiate Scan only with data stored by the proxy. Use the proxy data to track new URLs or to track all the URLs again including data stored by the proxy.

DETAILS:

- Use proxy data only and do not crawl for additional URLs

  If you select this option the Scanner will be executed only in the URLs stored by the proxy.

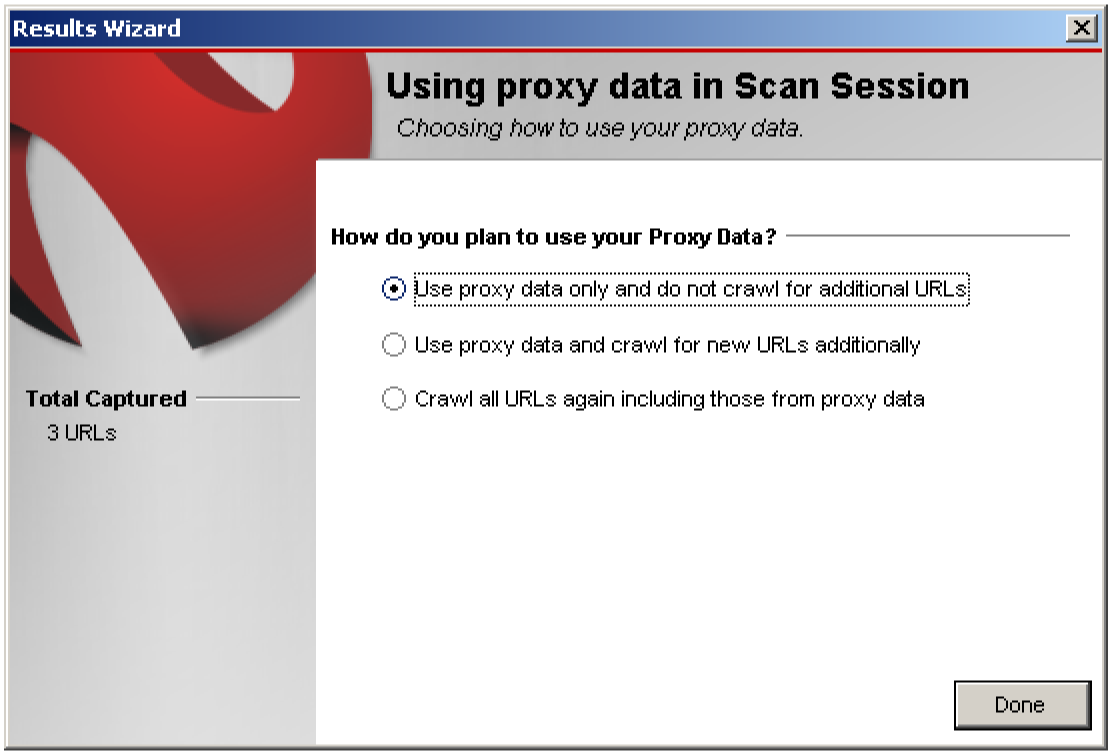
- Use proxy data and crawl for new URLs additionally

  If you select this option the Scanner will be executed in the URLs stored by the proxy but will also search for new URLs to be analyzed.

- Crawl all URLs again including those from proxy data

  If you select this option the Spider will track all URLs again, including the URLs stored beforehand by the proxy.

On the right corner of the screen, the quantity of URLs captured by the proxy at the time of navigation is then informed. Please select your option and click on the "Done" button to initiate execution of tests.


Results Wizard
Using proxy data in Scan Session
Choosing how to use your proxy data.
Done

How do you plan to use your Proxy Data?

[✓ Use proxy data only and do not crawl for additional URLs]

Use proxy data and crawl for new URLs additionally

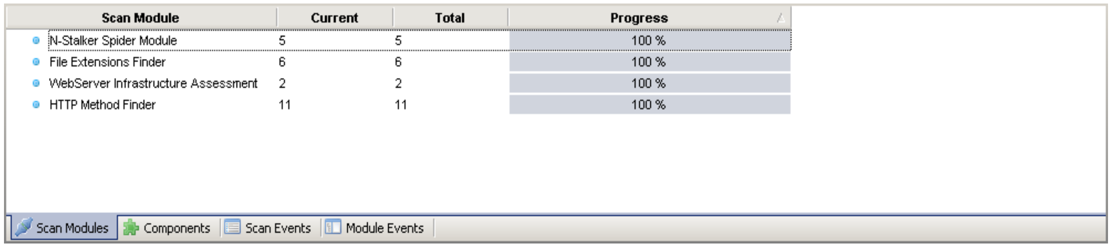
Crawl all URLs again including those from proxy data

Total Captured

3 URLs

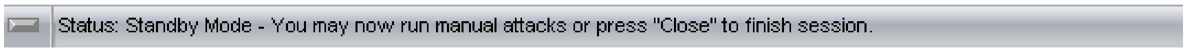
The "Scanning Session Interface" is open for execution of tests.

On the "Scan Modules" guide, on the footer of the N-Stalker scanning interface, there is a series of guides containing technical information about the scanning progress status including event items, components and depuration.



Scan Module	Current	Total	Progress
N-Stalker Spider Module	5	5	100 %
File Extensions Finder	6	6	100 %
WebServer Infrastructure Assessment	2	2	100 %
HTTP Method Finder	11	11	100 %

Before starting the manual test wait until the status shifts to the "Standby Mode - You may now run manual attacks or press "Close" to finish session" on the footer of the application.



Status: Standby Mode - You may now run manual attacks or press "Close" to finish session.

RESULT:

Thus the exploring of N-stalker a vulnerability assessment tool was completed successfully.

Ex. No. : 11 DEFEATING MALWARE - BUILDING TROJANS, ROOTKIT HUNTER

AIM:

To defeating malware by Building Trojans and Rootkit.

DESCRIPTION:

Prevention is better than a cure, and programs and executables should always be checked before use. Many sites provide an MD5sum with their programs to enable users to easily determine whether changes have been made. Email attachments should also always be scanned. In a high-security, controlled environment, a sheep dip system may even be used. (This term originates from the practice of completely immersing sheep in insecticide to make sure that they are free of pests.) A sheep dip computer can be used to screen suspect programs and is connected to a network only under controlled conditions. It can be used to further examine suspected files, incoming messages, and attachments.

KEY TERMS

Rootkit: A collection of tools that allows an attacker to take control of a system

Social engineering: A nontechnical attack that works by tricking or misleading an individual

Spyware: A type of malware that spies on the end user, sends pop-up messages, attempts to redirect the user to specifi c sites, or monitors their activity

Trojan: A type of malware known for tricking users into thinking it is something they want, while in reality malicious code is hidden inside

Virus: A piece of code that the user is tricked into installing that corrupts or destroys data

Worm: A self-propagating piece of malware that uses most of, if not all, available network bandwidth

Wrapper: A program that is used to combine a legitimate program with a piece of malware to create a single program that a user believes is safe to download and install

BUILDING TROJANS

Trojans and malware pose a real danger. This challenge highlights one of the ways that a hacker may distribute a Trojan. By default, older Windows systems automatically start a CD when it is inserted in the CD tray. You use this technique to distribute simulated malicious code. You need a blank CD and a CD burner for this exercise.

1. Create a text file named autorun.ini . Inside this text file, add the following contents:

```
[autorun]
Open paint.exe
Icon=paint.exe
```

2. Place the autorun.ini file and a copy of paint.exe into a folder to be burned to a CD.
3. After you have burned the CD, reinsert it in the CD-ROM drive and observe the results. The CD should autostart and automatically start the Paint program.
4. Think about the results. Although this exercise was benign, you could have easily used a Trojan program wrapped with a legitimate piece of software. Just leaving the CD lying around or giving it an attractive title, such as "pending 2006 bonuses," may lead someone to pick it up and view its contents. Anyone running the CD would then become infected. Even with AutoRun turned off, the user would only have to double-click the CD-ROM icon and the program would still run.

ROOTKITS

This exercise has you download a rootkit checker, install it, and examine its various options. Rootkit Hunter is an open source tool that checks Linux-based systems for the presence of rootkits and other unwanted tools. You can download and run this program on any Linux system, including the Kali OS found on the Wiley website wiley.com/go/networksecuritytestlab. Rootkit Hunter can be downloaded from http://rkhunter.sourceforge.net/.

1. Once you have started your Linux system, open a root terminal and download Rootkit Hunter. Enter the following at the command-line shell:

   wget http://downloads.rootkit.nl/rkhunter-<version>.tar.gz

   The <version> syntax requires you to enter the current version of the software. At the time of this writing, 1.3.0 is the most current version.

2. When the download has completed, unpack the archived file. You can do so by entering the following command:

   ```
   tar zxfrkhunter-<version>.tar.gz
   ```

   This command extracts Rootkit Hunter.

3. T o install Rootkit Hunter, change directories to the Rootkit Hunter folder:

   ```
   cdrkhunter
   ```

4. After you are in the proper directory, run the installer to complete the installation. To accomplish this, enter the following:

   ```
   ./installer.sh
   ```

   If everything goes correctly, the installation should finish successfully.
   The code listed here shows the output logfile of a successful installation:

```
 Rootkit Hunter installer 1.4.2 (Copyright 2003-2015, Michael Boelen)
 ---------------
 Starting installation/update

Checking  /usr/local... OK
Checking file retrieval tools... /usr/bin/wget
Checking installation directories...
- Checking /usr/local/rkhunter...Exists
- Checking /usr/local/rkhunter/etc...Exists
- Checking /usr/local/rkhunter/bin...Exists
- Checking /usr/local/rkhunter/lib/rkhunter/db...Exists
- Checking /usr/local/rkhunter/lib/rkhunter/docs...Exists
- Checking /usr/local/rkhunter/lib/rkhunter/scripts...Exists
- Checking /usr/local/rkhunter/lib/rkhunter/tmp...Exists
- Checking /usr/local/etc...Exists  - Checking /usr/local/bin...Exists
Checking system settings...
```

```
            ○ Perl... OK
Installing files...
Installing Perl module checker... OK
Installing Database updater... OK
Installing Portscanner... OK
Installing MD5 Digest generator... OK
Installing SHA1 Digest generator... OK
Installing Directory viewer... OK
Installing Database Backdoor ports... OK
Installing Database Update mirrors... OK
Installing Database Operating Systems... OK
Installing Database Program versions... OK
Installing Database Program versions... OK
Installing Database Default file hashes... OK
Installing Database MD5 blacklisted files... OK
Installing Changelog... OK
Installing Readme and FAQ... OK
Installing Wishlist and TODO... OK
Installing RK Hunter configuration file... Skipped (no overwrite)
Installing RK Hunter binary... OK
Configuration already updated.
Installation ready.
See /usr/local/rkhunter/lib/rkhunter/docs for more information.
Run 'rkhunter' (/usr/local/bin/rkhunter)
```

5. With Rootkit Hunter installed, run the program. There are a variety of options that you can use. To perform a complete system check, run this command:

```
Rkhunter–checkall
```

Rootkit Hunter can search for many different types of rootkits. A partial list is shown here:

55808 Trojan - Variant A
ADM W0rm
AjaKit
aPa Kit
Apache Worm
Ambient (ark) Rootkit
Balaur Rootkit
BeastKit beX2
BOBKit
CiNIK Worm (Slapper.B variant)
Danny-Boy's Abuse Kit
Devil RootKit
Dica
Dreams Rootkit

Duarawkz Rootkit
Flea Linux Rootkit
FreeBSD Rootkit
GasKit
Heroin LKM
HjC Rootkit
ignoKit
ImperalsS-FBRK
Irix Rootkit
Kitko
Knark
Li0n Worm
Lockit / LJK2
mod_rootme (Apache backdoor)
MRK
Ni0 Rootkit
NSDAP (RootKit for SunOS)
Optic Kit (Tux)
Oz Rootkit
Portacelo
R3dstorm Toolkit
RH-Sharpe's rootkit
RSHA's rootkit
Scalper Worm
Shutdown
SHV4 Rootkit
SHV5 Rootkit
Sin Rootkit
Slapper
Sneakin Rootkit
Suckit
SunOS Rootkit
Superkit
TBD (Telnet BackDoor)
TeLeKiT
T0rn Rootkit
Trojanit Kit
URK (Universal RootKit)
VcKit
Volc Rootkit
X-Org SunOS Rootkit
zaRwT.KiT Rootkit

OUTPUT:

When the scan is completed, you receive a message similar to the following:

```
---------------------------- Scan results ---------------------------
    MD5
    MD5 compared: 0
    Incorrect MD5 checksums: 0

    File scan
    Scanned files: 399
    Possible infected files: 0
    Application scan
    Vulnerable applications: 9

   Scanning took 15748 seconds

--------------------------------------------------------------------
Do you have some problems, undetected rootkits, false positives, ideas  or suggestions?
Please email me by filling in the contact form (@http://www.rootkit.nl)
------------------------------------------------------------------------
```

RESULT:

Thus the defeating Malware by Building Trojans and Rootkit was explored successfully.