

Poornima College of Engineering, Jaipur

Department of Computer Engineering



Lab Manual

7CS4-22: CYBER SECURITY LAB

Class: IV Year/ VII Sem

Poornima College of Engineering, Jaipur

Department of Computer Engineering

7CS4-22: Cyber Security Lab

List of Experiments

S. No.	Objective
1	Implement the following Substitution & Transposition Techniques concepts: a) Caesar Cipher b) Rail fence row & Column Transformation
2	Implement the Diffie-Hellman Key Exchange mechanism using HTML and JavaScript. Consider the end user as one of the parties (Alice) and the JavaScript application as other party (bob).
3	Implement the following Attack: a) Dictionary Attack b) Brute Force Attack
4	Installation of Wire shark and observe data transferred in client server communication using UDP/TCP and identify the UDP/TCP datagram.
5	Installation of rootkits and study about the variety of options.
6	Perform an Experiment to Sniff Traffic using ARP Poisoning.
7	Demonstrate intrusion detection system using Snort
8	Demonstrate how to provide secure data storage, secure data transmission and for creating digital signatures.
Beyond The Syllabus	
9	To develop a program to implement Advanced Encryption Standard for encryption and decryption
10	Develop secure coding practices to handle Code Injection Vulnerabilities such as SQL Injection
PROJECT Implementation	

PROJECT: In a small area location such as a house, office or in a classroom, there is a small network called a Local Area Network (LAN). The project aims to transfer a file peer-to-peer from one computer to another computer in the same LAN. It provides the necessary authentication for file transferring in the network transmission. By implementing the Server-Client technology, use a File Transfer Protocol mechanism and through socket programming, the end user is able to send and receive the encrypted and decrypted file in the LAN. An additional aim of the project is to transfer a file between computers securely in LANs. Elements of security are needed in the project because securing the files is an important task, which ensures files are not captured or altered by anyone on the same network. Whenever you transmit files over a network, there is a good chance your data will be encrypted by encryption technique.

Any algorithm like AES is used to encrypt the file that needs to transfer to another computer. The encrypted file is then sent to a receiver computer and will need to be decrypted before the user can open the file.

POORNIMA COLLEGE OF ENGINEERING, JAIPUR
DEPARTMENT OF COMPUTER ENGINEERING

Vision & Mission of Poornima College of Engineering

Vision

To create knowledge-based society with scientific temper, team spirit and dignity of labor to face the global competitive challenges.

Mission

To evolve and develop skill-based systems for effective delivery of knowledge so as to equip young professionals with dedication and commitment to excellence in all spheres of life

Vision & Mission of Department of Computer Engineering

Vision

Evolve as a centre of excellence with wider recognition and to adapt the rapid innovation in Computer Engineering.

Mission

- 1) To provide a learning-centered environment that will enable students and faculty members to achieve their goals empowering them to compete globally for the most desirable careers in academia and industry.
- 2) To contribute significantly to the research and the discovery of new arenas of knowledge and methods in the rapid developing field of Computer Engineering.
- 3) To support society through participation and transfer of advanced technology from one sector to another.

POORNIMA COLLEGE OF ENGINEERING, JAIPUR
DEPARTMENT OF COMPUTER ENGINEERING

PROGRAM EDUCATIONAL OBJECTIVES (PEO'S)

PEO1: Graduates will work productively as skillful engineers playing the leading roles in multifaceted teams

PEO2: Graduates will identify the solutions for challenging issues inspiring the upcoming generations leading them towards innovative, creative, and sophisticated technologies.

PEO3: Graduates will implement their pioneering ideas practically to create products and the feasible solutions of research oriented problems

PROGRAM OUTCOMES (POs)

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3: Design/development of solutions: Design solutions for 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.

PO4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: 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.

PO6: 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.

PO7: 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.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: 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.

PO11: 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.

PO12: 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: The ability to understand and apply knowledge of mathematics, system analysis & design, Data Modelling, Cloud Technology, and latest tools to develop computer based solutions in the areas of system software, Multimedia, Web Applications, Big data analytics, IOT, Business Intelligence and Networking systems.

PSO2: The ability to understand the evolutionary changes in computing, apply standards and ethical practices in project development using latest tools & Technologies to solve societal problems and meet the challenges of the future.

PSO3: The ability to employ modern computing tools and platforms to be an entrepreneur, lifelong learning and higher studies.

POORNIMA COLLEGE OF ENGINEERING, JAIPUR
DEPARTMENT OF COMPUTER ENGINEERING

**MAPPING OF KEY PHRASES OF THE INSTITUTES MISSION STATEMENT WITH
THE KEY PHRASES OF INSTITUTES VISION STATEMENT**

(Institution Mission Vs Institute Vision)

Key Phrases of the Mission Statement of the Institute	Key Phrases of the Vision Statement of the Institute		
	To create knowledge based society with scientific temper	Team spirit	To face the global competitive challenges
Skill based systems for effective delivery of knowledge	3		3
To equip young professionals with dedication		1	3
Excellence in all spheres of life	2		2

**MAPPING OF KEY PHRASES OF THE DEPARTMENTS VISION STATEMENT WITH
THE KEY PHRASES OF INSTITUTES MISSION STATEMENT**

(Department Vision Vs Institution Mission)

Key Phrases of the Vision Statement of the Department	Key Phrases of the Mission Statement of the Institute		
	Skill Based Systems	Delivery of Knowledge	Excellence in all spheres of life
Centre of Excellence	3	2	1
Wider recognition	2	2	1
Rapid innovation.	2	1	

POORNIMA COLLEGE OF ENGINEERING, JAIPUR
DEPARTMENT OF COMPUTER ENGINEERING

**MAPPING OF KEY PHRASES OF THE DEPARTMENTS MISSION STATEMENT
WITH THE KEY PHRASES OF DEPARTMENTS VISION STATEMENT
(Department Mission Vs Department Vision)**

Key Phrases of the Mission Statement of the Department	Key Phrases of the Vision Statement of the Department		
	Centre of Excellence	Wider recognition	Rapid innovation.
Learning-centered environment	3		2
Research and Discovery	2		2
Social Responsibility	1	2	1

**MAPPING OF KEY PHRASES PEOS WITH KEY PHRASES OF DEPARTMENTS
MISSION STATEMENT
(PEO Vs Department Mission)**

Key Phrases of PEO Statements	Key Phrases of the Mission of the Department		
	Learning-centered environment	Research and Discovery	Social Responsibility
Skillful engineers	3	2	
Innovative, Creative, and Sophisticated Technologies	3		1
Pioneering Ideas	2	2	

POORNIMA COLLEGE OF ENGINEERING, JAIPUR
DEPARTMENT OF COMPUTER ENGINEERING

MAPPING OF KEY PHRASES OF PSO WITH KEY PHRASES OF DEPARTMENTS

MISSION STATEMENT

(PSO Vs Department Mission)

Key Phrases of PSO Statement	Key Phrases of the Mission Department		
	Learning-centred environment	Research and Discovery	Social Responsibility
Technical Knowledge	3	2	
Standards, Ethic, Tools, Challenges Societal Problems	1	2	2
Entrepreneur, Lifelong Learning and Higher Studies.	2		

MAPPING OF KEY PHRASES OF PEO WITH KEY PHRASES OF PO

(PEO Vs PO)

Key Phrases of PEO Statement	Key Phrases of PO Statement											
	Engineering knowledge:	Problem analysis:	Design/development of solutions:	Conduct investigations of complex problems:	Modern tool usage:	The engineer and society:	Environment and sustainability:	Ethics:	Individual and teamwork:	Communication:	Project management and finance:	Lifelong learning:
Skillful engineers	3	3	3	3	2		1	1	2	2	1	2
Innovative, Creative, and Sophisticated Technologies		3	2	2		1	1	1				
Pioneering Ideas	2	1	1	2	2			1	1	2	1	2

POORNIMA COLLEGE OF ENGINEERING, JAIPUR
DEPARTMENT OF COMPUTER ENGINEERING

MAPPING OF KEY PHRASES OF PSO WITH KEY PHRASES PEO
(PSO Vs PEO)

Key Phrases of the PEO Department	Key Phrases of the PSO Department		
	Technical Knowledge	Standards, Ethic, Tools, Challenges Societal Problems	Entrepreneur, Lifelong Learning and Higher Studies.
Skillful engineers	3	2	2
Innovative, Creative, and Sophisticated Technologies	2	2	2
Pioneering Ideas		1	2

COURSE OUTCOMES

Course: 7CS4-22: CYBER SECURITY LAB

On completion of this course:	
CO1	Graduate will be able to implement the substitution and transposition techniques for plain text encryption and decryption.
CO2	Graduate will be able to design a solution for Key Exchange problem and understand the general attacks on system.
CO3	Graduate will be able to analyse the data transferred in client server communication and working of various network protocol using different security-based tools like Wire shark, tcpdump, rootkits, snort etc.
CO4	Graduate will be able to demonstrate encryption and decryption techniques for secure data transmission across network and creating digital signatures

MAPPING OF CO WITH PO AND PSO

CO's	PO's												PSO's		
	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	-	-	-	2	-	-	-	-	-	-	-	-	2	3	-
CO2	-	-	-	-	-	3	-	-	-	-	-	-	-	2	-
CO3	-	-	-	-	3	-	-	-	-	-	-	-	3	-	-
CO4	3	-	-	-	-	-	-	-	-	-	-	-	-	2	2

Mapping Levels: 1- Low, 2- Moderate, 3-Strong



RAJASTHAN TECHNICAL UNIVERSITY, KOTA

Scheme & Syllabus

IV Year- VII Semester: B. Tech. (Computer Science & Engineering)

7CS4-22: Cyber Security Lab

Credit: 2

Max. Marks: 100(IA:60, ETE:40)

OL+OT+4P

End Term Exam: 2 Hours

SN	List of Experiments
1	Implement the following Substitution & Transposition Techniques concepts: a) Caesar Cipher b) Rail fence row & Column Transformation
2	Implement the Diffie-Hellman Key Exchange mechanism using HTML and JavaScript. Consider the end user as one of the parties (Alice) and the JavaScript application as other party (bob).
3	Implement the following Attack: a) Dictionary Attack b) Brute Force Attack
4	Installation of Wire shark, tcpdump, etc and observe data transferred in client server communication using UDP/TCP and identify the UDP/TCP datagram.
5	Installation of rootkits and study about the variety of options.
6	Perform an Experiment to Sniff Traffic using ARP Poisoning.
7	Demonstrate intrusion detection system using any tool (snort or any other s/w).
8	Demonstrate how to provide secure data storage, secure data transmission and for creating digital signatures.

Experiment 1:

Obj: Implement the following Substitution & Transposition Techniques concepts:

a) Caesar Cipher

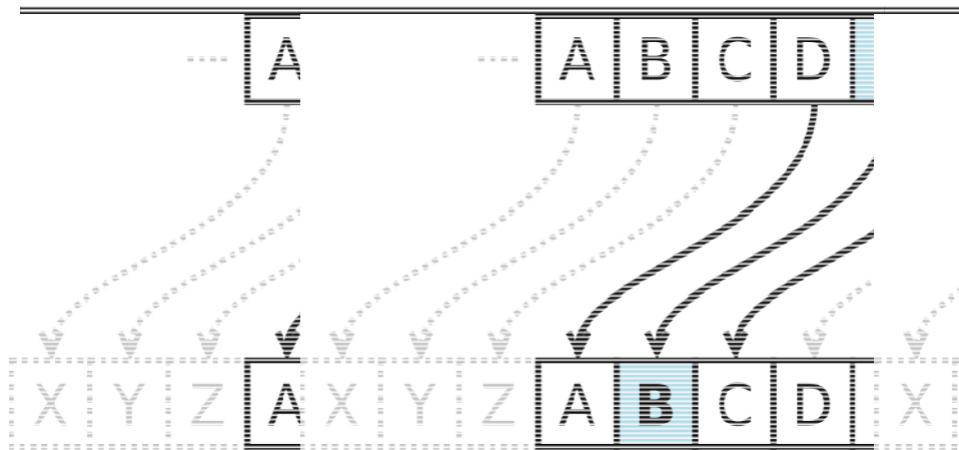
b) Rail fence row & Column Transformation

DESCRIPTION:

To encrypt a message with a Caesar cipher, each letter in the message is changed using a simple rule: shift by three. Each letter is replaced by the letter three letters ahead in the alphabet. A becomes D, B becomes E, and so on. For the last letters, we can think of the alphabet as a circle and "wrap around".

W becomes Z, X becomes A, Y becomes B, and Z becomes C. To change a message back, each letter is replaced by the one three before it.

EXAMPLE:



Algorithm:

a) Caesar Cipher

1. Caesar cipher is an example of a substitution cipher in which plaintext letters in the original message are replaced (substituted for) by cipher text letters

2. The easiest way to understand this is to consider that there are two alphabets:

PLAIN_ALPHABET: ABCDEFGHIJKLMNOPQRSTUVWXYZ CIPHER_ALPHABET:
DEFGHIJKLMNOPQRSTUVWXYZABC

3. The cipher alphabet is a shifted version of the plain alphabet. In this case, each letter in the cipher alphabet has to be shifted by 3 places to the right

4. The shift -- (i.e., the number 3) is the secret key which must be shared by Alice and Bob if they want to send secret messages using this cipher

5. To encrypt the message MEET ME AT THE DOCK we would replace all the Ms in the message with the corresponding letter from the cipher alphabet

6. So M is replaced by P. And we would replace all the Es by H and so on. Thus, the encryption of our message would be PHHW PH DW WLH GRFN

Sample Output: Enter any String: Hello World Enter the Key: 5 Encrypted String is:
MjqqtBtwqi Decrypted String is: Hello World

Encryption :

```
class CaesarCipher
{

    public static StringBuffer encrypt(String text, int s)
    {
        StringBuffer result= new StringBuffer();

        for (int i=0; i<text.length(); i++)
        {
            if (Character.isUpperCase(text.charAt(i)))
            {
                char ch = (char)((((int)text.charAt(i) +
                                     s - 65) % 26 + 65));
                result.append(ch);
            }
        }
    }
}
```

```

        }
        else
        {
            char ch = (char)((((int)text.charAt(i) +
                                s - 97) % 26 + 97);

            result.append(ch);
        }
    }
    return result;
}

public static void main(String[] args)
{
    String text = "ATTACKATONCE";
    int s = 4;
    System.out.println("Text : " + text);
    System.out.println("Shift : " + s);
    System.out.println("Cipher: " + encrypt(text, s));
}
}

```

Output: Text: ABCDEF

Shift: 4

Cipher: DEFGHI

Decryption :

```
import java.util.Scanner;
```

```
public class CaesarCipherJava {
```

```
public static void main(String...s){
String message, decryptedMessage = "";
int key;
char ch;
Scanner sc = new Scanner(System.in);
System.out.println("Enter a message: ");
message = sc.nextLine();
System.out.println("Enter key: ");
key = sc.nextInt();

for(int i = 0; i < message.length(); ++i){
ch = message.charAt(i);
if(ch >= 'a' && ch <= 'z'){
    ch = (char)(ch - key);

    if(ch < 'a'){
        ch = (char)(ch + 'z' - 'a' + 1);
    }

    decryptedMessage += ch;
}
else if(ch >= 'A' && ch <= 'Z'){
    ch = (char)(ch - key);

    if(ch < 'A'){
        ch = (char)(ch + 'Z' - 'A' + 1);
    }

    decryptedMessage += ch;
}
else {
```



```

        decryptedMessage += ch;
    }
}
System.out.println("Decrypted Message = " + decryptedMessage);
}
}

```

Output: Enter a message: DEFGHI

Enter key: 4

Encrypted msg: ABCDEF

b) Rail fence row & Column Transformation

Rail fence ciphers are examples of transposition ciphers: The characters in the plaintext message are permuted to create the ciphertext. In the rail fence cipher, the permutation is obtained from a very simple pattern.

Other transposition ciphers use other manipulations to permute the characters. The encryption key for a rail fence cipher is a positive integer.

In the rail fence cipher, the plain text 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.

EXAMPLE:

	A	U	T	H	O	R	
	1	6	5	2	3	4	
W	E	A	R	E	D		
I	S		C	O	V	E	
R	E	D	S	A	V		
E	Y	O	U	R	S		
E	L	F	A	B	C		

yields the cipher

WIREEROSUA EVARBDEVSCACDOFESEYL.

Suppose we want to encrypt the message “buy your books in August” using a rail fence cipher with encryption key 3. Here is how we would proceed.

i. Arrange the plaintext characters in an array with 3 rows (the key determines the number of rows),

forming a zig-zag pattern:

```
b - - - o - - - o - - - i - - - g - - -  
- u - y - u - b - o - s - n - u - u - t  
- - y - - - r - - - k - - - a - - - s -
```

ii. Then concatenate the non-empty characters from the rows to obtain the ciphertext:

BOOIGUYUBOSNUUTYRKAS

For practice, try encrypting the plaintext “a new semester begins with football and moving vans” using a rail fence cipher with key 5 and see if you obtain the ciphertext

ASNODANETISOTNMVNEGWFBAOGSWEREIHALVNSBTLI

The following ciphertext was produced with a rail fence cipher with key 4.

EOCSNYUWLEJYREASONS

See if you can decrypt it.

Because transposition ciphers permute the letters of the plaintext, the plaintext and ciphertext will share the same frequency distribution of characters. Because the frequencies of characters in English (and other languages) are well-known, with messages of sufficient length a transposition cipher can be identified with reasonable confidence using frequency analysis.

If a rail fence cipher is suspected, it is feasible to test it by exhaustive search: In order to avoid undesirable results, the choice of key for a rail fence cipher is quite restricted. Of course, if one were to choose 1 as the key, the plaintext and ciphertext would coincide. The reader should consider the following questions about other possible key choices: What happens if the key is roughly the same as (or larger than) the number of characters in the plaintext? What does this tell you about the number of viable keys for a given message? The rail fence cipher is a very old encryption scheme, pre-dating the Middle Ages. It was used as a field cipher by both sides in the US Civil War.

```
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 RailFence{  
    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);

}
}
```

Experiment 2:

Implement the Diffie-Hellman Key Exchange mechanism using HTML and JavaScript. Consider the end user as one of the parties (Alice) and the JavaScript application as other party (bob).

The Diffie–Hellman (DH) Algorithm is a key-exchange protocol that **enables two parties communicating over public channel** to establish a mutual secret without it being transmitted over the Internet. DH enables the two to use a public key to encrypt and decrypt their conversation or data using symmetric cryptography.

Step by Step Explanation

Alice	Bob
Public Keys available = P, G	Public Keys available = P, G
Private Key Selected = a	Private Key Selected = b
Key generated = $x = G^a \text{ mod } P$	Key generated = $y = G^b \text{ mod } P$
Exchange of generated keys takes place	
Key received = y	key received = x
Generated Secret Key = $k_a = y^a \text{ mod } P$	Generated Secret Key = $k_b = x^b \text{ mod } P$
Algebraically, it can be shown that $k_a = k_b$	
Users now have a symmetric secret key to encrypt	

Example: Step 1: Alice and Bob get public numbers $P = 23$, $G = 9$

Step 2: Alice selected a private key $a = 4$ and

Bob selected a private key $b = 3$

Step 3: Alice and Bob compute public values

Alice: $x = (9^4 \bmod 23) = (6561 \bmod 23) = 6$

Bob: $y = (9^3 \bmod 23) = (729 \bmod 23) = 16$

Step 4: Alice and Bob exchange public numbers

Step 5: Alice receives public key $y = 16$ and

Bob receives public key $x = 6$

Step 6: Alice and Bob compute symmetric keys

Alice: $k_a = y^a \bmod p = 65536 \bmod 23 = 9$

Bob: $k_b = x^b \bmod p = 216 \bmod 23 = 9$

Step 7: 9 is the shared secret.

HTML FILE

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<h1> Diffie-Hellman Key Exchange mechanism </h1>
```

```
<p> Exp 2 :Implement the Diffie-Hellman Key Exchange mechanism using HTML and  
JavaScript. Consider the end user as one of the parties (Alice) and the JavaScript application as  
other party (bob). </p>
```

```
</body>
```

```
<script>
```

```
function power( m,n,o)
```

```
{
```

```
        if (n==1)
            return m;
        else
            return(Math.pow(m,n) % o);
    }
```

```
var P, G, Aice_private, Bob_private, Alice_x, Bob_y, Ka, Kb
```

```
// Alice and Bob select two prime numbers P and G publicly
```

```
P = 23;
```

```
G = 9;
```

```
document.write ("The value of first prime number P:=" +P + "<br>");
```

```
document.write ("The value of Second prime number G:=" +G + "<br>");
```

```
// Alice and Bob select their private key
```

```
Aice_private=4;
```

```
Bob_private =3;
```

```
// Alice and Bob create x and y using formula and exchange the values
```

```
Alice_x = power( G, Aice_private, P);
```

```
Bob_y = power ( G, Bob_private, P);
```

```
// now Alice and Bob create secret key Ka and Kb
```

```
Ka= power (Bob_y, Aice_private, P);
```



```
Kb = power(Alice_x, Bob_private, P);  
document.write ("The value of Alice Screte key Ka:=" +Ka +"<br>");  
document.write("<br>");  
document.write("<br>");
```

```
document.write ("The value of Bob Screte key Kb:=" +Kb +"<br>");
```

```
</script>
```

```
</html>
```

Experiment 3:

Implement the following Attack:

- a) **Dictionary Attack**
- b) **Brute Force Attack**

Dictionary Attack Definition:

“A type of brute force attack where an intruder attempts to crack a password-protected security system with a “dictionary list” of common words and phrases used by businesses and individuals.”

Both are common types of cybersecurity attacks in which an attacker tries to log in to a user’s account by systematically checking and attempting all possible passwords and passphrases until the correct one is found. These brute-force and dictionary attacks are common, due to large quantities of individuals reusing common password variations.

After all, the easiest way to attack a system is through the front door, and there must be *some* way to log in. If you have credentials, you can log in as a normal user would, likely without generating suspicious log entries, tripping IDS signatures, or needing an unpatched vulnerability. If you have the credentials for the system administrator, life is even easier. Attackers have neither of these luxuries; here’s an overview of how they utilize brute-force and dictionary attacks to gain access.

In a dictionary attack, the attacker utilizes a wordlist in the hopes that the user’s password is a commonly used word (or a password seen in previous sites). Dictionary attacks are optimal for passwords that are based on a simple word (e.g. 'cowboys' or 'longhorns'). Wordlists aren’t restricted to English words; they often also include common passwords (e.g. 'password,' 'letmein,' or 'iloveyou,' or '123456'). But modern systems restrict their users from such simple passwords, requiring users to come up with strong passwords that would hopefully not be found in a wordlist.

```
import java.security.*;
```

```

import java.io.*;
import java.util.*;
import java.lang.StringBuilder;
import javax.xml.bind.DatatypeConverter;

public class DictionaryAttack {
    //Extracted from http://www.jmdoudoux.fr/java/dej/chap-jca.htm
    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();
    }

    //This method takes a string, computes its SHA-1 hash,
    //and converts it into HEX using the bytesToHex method
    public static String stringToSha1(String input) throws Exception {
        //Setup a MessageDigest for SHA1
        MessageDigest md = MessageDigest.getInstance("SHA1");
        md.reset();

        //Setup the MessageDigest with our input string
        md.update(input.getBytes("UTF-8"));

        //Convert the string's digest to HEX
        String sha1 = bytesToHex(md.digest());
        return sha1;
    }
}

```

```
}
```

```
//This method takes a byte array holding a salt and a string input
```

```
//and returns the concatenated salt || input in byte array format
```

```
public static byte[] concatenate_salt_with_string(byte[] salt, String input) throws
```

```
Exception {
```

```
    //Convert input string to bytes
```

```
    byte[] input_byte = input.getBytes("UTF-8");
```

```
    //Create byte array sufficiently large
```

```
    byte[] concatenated = new byte[salt.length + input_byte.length];
```

```
    //Insert the salt first
```

```
    System.arraycopy(salt, 0, concatenated, 0, salt.length);
```

```
    //Insert the input string converted to bytes
```

```
    System.arraycopy(input_byte, 0, concatenated, salt.length, input_byte.length);
```

```
    //Return the concatenated salt and string in a byte array
```

```
    return concatenated;
```

```
}
```

```
//This method takes a string, a salt, computes its salted SHA-1 hash,
```

```
//and converts it into HEX using the bytesToHex method
```

```
public static String stringToSha1_salted(byte[] salt, String input) throws Exception {
```

```
    //Setup a MessageDigest for SHA1
```

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

```
    md.reset();
```

```
    //Use the concatenate_salt_with_string method to concatenate the salt with the
```

```
input
```

```
    byte[] concatenated = concatenate_salt_with_string(salt, input);
```

```
    //Setup the MessageDigest with our input string
```

```
    md.update(concatenated);
```

```

//Convert the string's digest to HEX
String sha1 = bytesToHex(md.digest());
    return sha1;
}

public static void main(String[] args) throws Exception {
    //Notify the user the program is starting.
    System.out.println("Let's get things started.");

    //Load the provided password file into stream and buffer
    File passwords_file = new File("/E:/Cyber Security Lab 7th sem/LAB
PROGRAMS/password.txt");
    FileInputStream password_stream = new FileInputStream(passwords_file);
    BufferedReader password_buffer = new BufferedReader(new
InputStreamReader(password_stream));

    //Initialize 3 hashmaps, one for non-salted passwords, one for salted passwords,
    //and one for the salts of salted passwords.
    Map<String, String> non_salted_passwords = new HashMap<String, String>();
    Map<String, String> salted_passwords = new HashMap<String, String>();
    Map<String, String> salted_passwords_salts = new HashMap<String, String>();

    //We parse the buffer to extract user account names and passwords
    String password_file_line = null;
    while ((password_file_line = password_buffer.readLine()) != null) {
        String[] splited = password_file_line.split("\\s+");

        //First case: password hashed with no salt
        if(splited.length == 3){
            non_salted_passwords.put(splited[0], splited[2]);

```

```

    }

    //Second case: password hashed with a salt
    else{
        salted_passwords.put(splited[0], splited[3]);
        salted_passwords_salts.put(splited[0], splited[2]);
    }
}

//We are done reading the password file, we can close its buffer
password_buffer.close();

//Load the provided Dictionary into stream and buffer
File fin = new
File("/Users/nicolas/Documents/eclipseworkspace/dictionaryattack/src/english.0");
FileInputStream fis = new FileInputStream(fin);

//Construct BufferedReader from InputStreamReader
BufferedReader br = new BufferedReader(new InputStreamReader(fis));

//We parse the buffer to test matches for hashed password,
//reversed passwords, non vowel passwords, and salted versions of password (if
required).
String line = null;
while ((line = br.readLine()) != null) {
    //We first iterate through the non salted passwords
    Iterator non_salted_passwords_it = non_salted_passwords.entrySet().iterator();
    while (non_salted_passwords_it.hasNext()) {
        //We extract the key,value pair from the HashTable entry
        Map.Entry pair = (Map.Entry)non_salted_passwords_it.next();
        String account_name = pair.getKey().toString();

```

```

String account_password_hash = pair.getValue().toString();

//We test if the password matches an unmodified dictionary entry
if(account_password_hash.equals(stringToSha1(line))) {
    System.out.println(account_name + "'s password is '" + line + "'");
}

//We test if the password matches a reversed dictionary entry
String reversed_line = new StringBuilder(line).reverse().toString();
if(account_password_hash.equals(stringToSha1(reversed_line))) {
    System.out.println(account_name + "'s password is '" + reversed_line + "'");
}

//We test if the password matches a dictionary entry without its vowels
String line_without_vowels = line.replaceAll("[AEIOUaeiou]", "");
if(account_password_hash.equals(stringToSha1(line_without_vowels))) {
    System.out.println(account_name + "'s password is '" + line_without_vowels +
    "'");
}

//We then iterate through the salted passwords
Iterator salted_passwords_it = salted_passwords.entrySet().iterator();
while (salted_passwords_it.hasNext()) {
    //We extract the key,value pair from the HashTable entry
    Map.Entry salted_pair = (Map.Entry)salted_passwords_it.next();
    String account_name = salted_pair.getKey().toString();
    String account_password_hash = salted_pair.getValue().toString();
    //We extract the corresponding salt from the HashTable of salts
    byte[] account_password_hash_salt =
    DatatypeConverter.parseHexBinary(salted_passwords_salts.get(account_name));

```

```
//We test if the password matches an unmodified dictionary entry

if(account_password_hash.equals(stringToSha1_salted(account_password_hash_salt,line))) {
    System.out.println(account_name + "'s password is '" + line + "'");
}

//We test if the password matches a reversed dictionary entry
String reversed_line = new StringBuilder(line).reverse().toString();

if(account_password_hash.equals(stringToSha1_salted(account_password_hash_salt,reversed_line))) {
    System.out.println(account_name + "'s password is '" + reversed_line + "'");
}

//We test if the password matches a dictionary entry without its vowels
String line_without_vowels = line.replaceAll("[AEIOUaeiou]", "");

if(account_password_hash.equals(stringToSha1_salted(account_password_hash_salt,line_without_vowels))) {
    System.out.println(account_name + "'s password is '" + line_without_vowels + "'");
}

}

//We are done using the dictionary file, we can close its buffer
br.close();

//Notify the user our program is done running.
System.out.println("The program terminated.");
```



```
}  
}
```

Brute Force Attacks Defined:

- A brute force attack, also known as an exhaustive search, is a cryptographic hack that relies on guessing possible combinations of a targeted password until the correct password is discovered.
- The longer the password, the more combinations that will need to be tested.
- A brute force attack can be time consuming, difficult to perform if methods such as data obfuscation are used, and at times downright impossible.
- However, if the password is weak it could merely take seconds with hardly any effort.
- Weak passwords are like shooting fish in a barrel for attackers, which is why all organizations should enforce a strong password policy across all users and systems.

How are Brute Force Attacks Used?

Brute force attacks are usually used to obtain personal information such as passwords, passphrases, usernames and Personal Identification Numbers (PINs), and use a script, hacking application, or similar process to carry out a string of continuous attempts to get the information required.

Goals of a brute force attack include:

- Theft of personal information such as passwords, passphrases and other information used to access online accounts and network resources
- Harvesting credentials to sell to third parties
- Posing as users to send phishing links or spread fake content
- Defacement of websites and other information in the public domain that could damage the reputation of the organization
- Redirecting domains to sites holding malicious content

They can also be used for positive gains. Many IT specialists use this method of attack to test network security and more specifically, the strength of the encryption used on the network.

Brute Force Attack Tools

An attacker is usually aided by automated software that uses computing to systematically check password combinations until the correct one is identified. Using a brute force password cracking application is required in order to go through numerous combinations and possibilities that can be difficult or impossible to calculate by a human alone. Popular examples of brute force attack tools include:

- Aircrack-ng
- John the Ripper
- L0phtCrack
- RainbowCrack

Types of Brute Force Attack

There are a number of different types of brute force attack, each of which has the same goals detailed above.

Hybrid Brute Force Attacks

You may have heard of dictionary attacks. These are one of the most common forms of brute force attack and use a list of words in a dictionary to crack passwords. Other types of attack may use a list of commonly used passwords. If your password is 'password', for example, a brute force bot would be able to crack your password within seconds.

Reverse Brute Force Attack

Reverse brute force attacks don't target a specific username, but instead, use a common group of passwords or an individual password against a list of possible usernames.

Credential Stuffing

When a username and password pairing is known by the attacker, they can use this information to gain access to multiple websites and network resources. For example, many users choose the same password to access many different websites for the sake of simplicity. Taking precautions like using two-factor authentication and using different passwords for every different network resources can help to prevent brute force attacks that rely on credential stuffing.

How to Prevent Brute Force Attacks

Brute force attacks typically rely on weak passwords and careless network administration.

Fortunately, these are both areas that can be improved easily in order to prevent vulnerabilities that could bring your network or website resources to their knees. For example, utilizing strong

passwords, allowing a limited number of login attempts and enabling two-factor authentication can help to prevent brute force attacks.

Ultimately, it is important to educate your organization on the importance of password strength and the general information security habits. Even with a strong password, employees can fall victim to insider threats if security is not a strong part of your culture. Learn more about Forcepoint's Insider Threat Program offerings.

```
import java.util.Scanner;

public class BruteForce {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int i, j, l, k = 97, key = 0, flag = 0, index = 0, keyVal;
        String pt;
        char[] ct1 = new char[10];
        char[] pt1 = new char[10];
        char temp;
        System.out.println("ENTER PLAIN TEXT");
        pt = sc.next();
        System.out.println("ENTER KEY VALUE :");
        key = sc.nextInt();
        for (i = 0; i < pt.length(); i++) {
            for (j = 0; j < 26; j++) {

                if (pt.charAt(i) == ' ') {
                    flag = 0;
                    break;
                }
                temp = (char) (j + k);
                if (pt.charAt(i) == temp) {
```

```

        flag = 1;
        index = j;
        break;
    }
}
if (flag == 1) {

    char c = (char) (((index + key) % 26) + 97);
    ct1[i] = c;

}
}
System.out.println("ENCRYPTED DATA:");
for (i = 0; i < pt.length(); i++) {
    System.out.print(ct1[i]);
}
System.out.println("\n" + "DECRYPTION OF DATA USING BRUTE-FORCE ATTACK
:");
key = 1;
while (key <= 26) {
    for (i = 0; i < pt.length(); i++) {
        for (j = 0; j < 26; j++) {
            if (ct1[i] == ' ') {
                flag = 0;
                break;
            }
            temp = (char) (j + k);
            if (ct1[i] == temp) {
                flag = 1;
                index = j;
                break;
            }
        }
    }
}

```

```
        }  
    }  
    keyVal = index - key;  
    if (flag == 1 & keyVal > 0) {  
        pt1[i] = (char) ((keyVal % 26) + 97);  
  
    } else if (flag == 1) {  
        pt1[i] = (char) ((26 + keyVal) + 97);  
    }  
  
    }  
    System.out.print("\n" + "DECRYPTED DATA:");  
    for (i = 0; i < pt.length(); i++) {  
        System.out.print(pt1[i]);  
    }  
    key++;  
    }  
    }  
}
```

Experiment 4:

Installation of Wire shark, tcpdump, etc and observe data transferred in client server communication using UDP/TCP and identify the UDP/TCP datagram.

To install Wireshark:

1. Open Windows Explorer.
2. Select the Downloads folder.
3. Locate the version of Wireshark you downloaded in Activity 2. ...
4. If you see a User Account Control dialog box, select Yes to allow the program to make changes to this computer.
5. Select Next > to start the Setup Wizard.
6. Review the license agreement.

tcpdump installation steps:

1. Download the rpm package for tcpdump.
2. Log in to DSVA via SSH as DSVA user. The default password is “dsva”.
3. Switch to root user using this command: \$sudo -s.
4. Upload the package to DSVA under path:/home/dsva. ...
5. Unpack the tar package: ...
6. Install the rpm packages:

Experiment 5:

Installation of rootkits and study about the variety of options.

- A rootkit is a collection of programs/software tools — typically malicious — that gives a threat actor remote administrative access to and control over a computer while hiding its presence on that machine. In simpler words, a rootkit is typically associated with malware that you can't see but make sure that the cyber-criminal sees your computer and, possibly, your actions as well.
- Cybercriminals use **rootkits** to hide and protect malware on a computer.
- The rootkit itself isn't necessarily harmful; what's dangerous is the various forms of malware inside them.
- Many rootkits are designed to invade the “root,” or kernel, of the program, and therefore operate without announcing their presence to the owner of the computer.
- Malware in a rootkit can steal data and take over a system for malicious purposes, all while remaining undetected.
- Installed in the core operating system of a computer, rootkits are difficult to detect and potentially harmful to a system. They can block some antivirus and antimalware software, rendering them ineffective, in part because rootkits activate before an operating system boots up.
- Rootkits could remain in place for years. Their core role is to hide any trail of their existence. They can even alter data reports from a system to avoid detection.
- The Alureon and Sinowal Trojans and Sirefef, Rustock and Cutwail, are all well-known malware associated with rootkits.
- Windows operating systems have some built-in technologies to help protect them from rootkits, but they are not perfect. Criminal programmers can design a rootkit virus to change how an operating system processes information and commands, including login protocols.

Breaking the term rootkit into the two component words, root and kit, is a useful way to define it. Root is a UNIX/Linux term that's the equivalent of Administrator in Windows. The word kit denotes programs that allow someone to obtain root/admin-level access to the computer by executing the programs in the kit — all of which is done without end-user consent or knowledge.

A rootkit is a type of malicious software that is activated each time your system boots up. Rootkits are difficult to detect because they are activated before your system's Operating System has completely booted up. A rootkit often allows the installation of hidden files, processes, hidden user accounts, and more in the systems OS. Rootkits are able to intercept data from terminals, network connections, and the keyboard.

Rootkits have two primary functions: remote command/control (back door) and software eavesdropping. Rootkits allow someone, legitimate or otherwise, to administratively control a computer. This means executing files, accessing logs, monitoring user activity, and even changing the computer's configuration. Therefore, in the strictest sense, even versions of VNC are rootkits. This surprises most people, as they consider rootkits to be solely malware, but in of themselves they aren't malicious at all.

The presence of a rootkit on a network was first documented in the early 1990s. At that time, Sun and Linux operating systems were the primary targets for a hacker looking to install a rootkit. Today, rootkits are available for a number of operating systems, including Windows, and are increasingly difficult to detect on any network.

PROCEDURE:

STEP-1: Download Rootkit Tool from GMER website www.gmer.net.

STEP-2: This displays the Processes, Modules, Services, Files, Registry, RootKit / Malwares, Autostart, CMD of local host.

STEP-3: Select Processes menu and kill any unwanted process if any.

STEP-4: Modules menu displays the various system files like .sys, .dll

STEP-5: Services menu displays the complete services running with Autostart, Enable, Disable, System, Boot.

STEP-6: Files menu displays full files on Hard-Disk volumes.

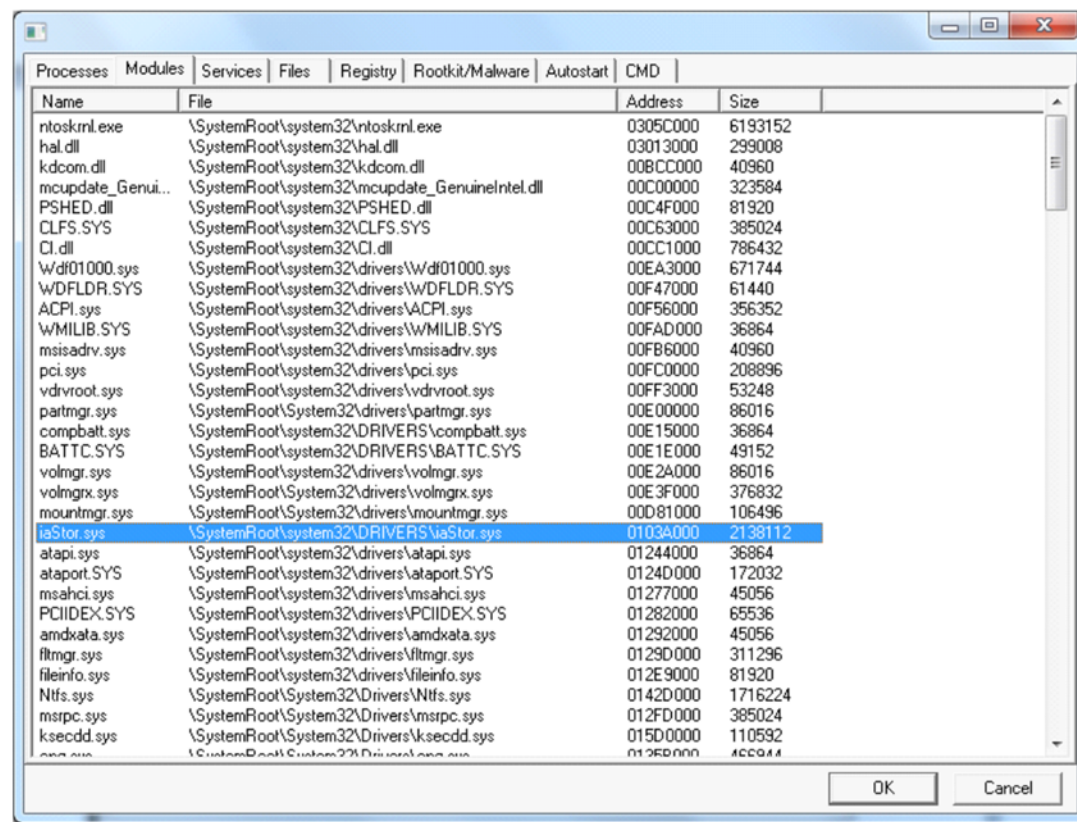
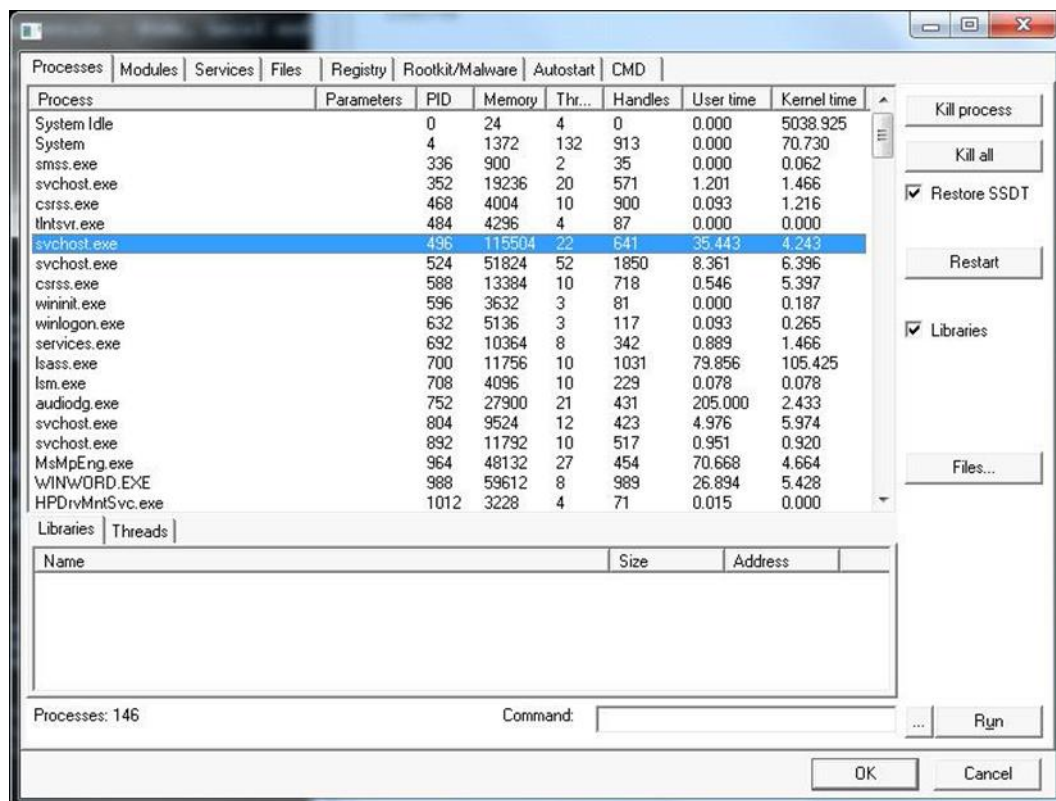
STEP-7: Registry displays Hkey_Current_user and Hkey_Local_Machine.

STEP-8: Rootkits / Malwares scans the local drives selected.

STEP-9: Autostart displays the registry base Autostart applications.

STEP-10: CMD allows the user to interact with command line utilities or Registry

SCREENSHOTS:



Experiment 6:

Perform an Experiment to Sniff Traffic using ARP Poisoning.

Address Resolution Protocol (ARP) is a stateless protocol used for resolving IP addresses to machine MAC addresses. All network devices that need to communicate on the network broadcast ARP queries in the system to find out other machines' MAC addresses. ARP Poisoning is also known as **ARP Spoofing**.

Here is how ARP works –

- When one machine needs to communicate with another, it looks up its ARP table.
- If the MAC address is not found in the table, the **ARP_request** is broadcasted over the network.
- All machines on the network will compare this IP address to MAC address.
- If one of the machines in the network identifies this address, then it will respond to the **ARP_request** with its IP and MAC address.
- The requesting computer will store the address pair in its ARP table and communication will take place.

What is ARP Spoofing?

ARP packets can be forged to send data to the attacker's machine.

- ARP spoofing constructs a large number of forged ARP request and reply packets to overload the switch.
- The switch is set in **forwarding mode** and after the **ARP table** is flooded with spoofed ARP responses, the attackers can sniff all network packets.

Attackers flood a target computer ARP cache with forged entries, which is also known as **poisoning**. ARP poisoning uses Man-in-the-Middle access to poison the network.

What is MITM?

The Man-in-the-Middle attack (abbreviated MITM, MitM, MIM, MiM, MITMA) implies an active attack where the adversary impersonates the user by creating a connection between the victims and sends messages between them. In this case, the victims think that they are communicating with each other, but in reality, the malicious actor controls the communication.

A third person exists to control and monitor the traffic of communication between two parties. Some protocols such as **SSL** serve to prevent this type of attack.

ARP Poisoning – Exercise

In this exercise, we have used **BetterCAP** to perform ARP poisoning in LAN environment using VMware workstation in which we have installed **Kali Linux** and **Etmetercap** tool to sniff the local traffic in LAN.

For this exercise, you would need the following tools –

- VMware workstation
- Kali Linux or Linux Operating system
- Ettercap Tool
- LAN connection

Note – This attack is possible in wired and wireless networks. You can perform this attack in local LAN.

Step 1 – Install the VMware workstation and install the Kali Linux operating system.

Step 2 – Login into the Kali Linux using username pass “root,toor”.

Step 3 – Make sure you are connected to local LAN and check the IP address by typing the command **ifconfig** in the terminal.

Step 4 – Open up the terminal and type “Ettercap –G” to start the graphical version of Ettercap.

Step 5 – Now click the tab “sniff” in the menu bar and select “unified sniffing” and click OK to select the interface. We are going to use “eth0” which means Ethernet connection.

Step 6 – Now click the “hosts” tab in the menu bar and click “scan for hosts”. It will start scanning the whole network for the alive hosts.

Step 7 – Next, click the “hosts” tab and select “hosts list” to see the number of hosts available in the network. This list also includes the default gateway address. We have to be careful when we select the targets.

Step 8 – Now we have to choose the targets. In MITM, our target is the host machine, and the route will be the router address to forward the traffic. In an MITM attack, the attacker intercepts the network and sniffs the packets. So, we will add the victim as “target 1” and the router address as “target 2.”

In VMware environment, the default gateway will always end with “2” because “1” is assigned to the physical machine.

Step 9 – In this scenario, our target is “192.168.121.129” and the router is “192.168.121.2”. So we will add target 1 as **victim IP** and target 2 as **router IP**.

Step 10 – Now click on “MITM” and click “ARP poisoning”. Thereafter, check the option “Sniff remote connections” and click OK.

Step 11 – Click “start” and select “start sniffing”. This will start ARP poisoning in the network which means we have enabled our network card in “promiscuous mode” and now the local traffic can be sniffed.

Note – We have allowed only HTTP sniffing with Ettercap, so don’t expect HTTPS packets to be sniffed with this process.

Step 12 – Now it’s time to see the results; if our victim logged into some websites. You can see the results in the toolbar of Ettercap.

Experiment 7:

Demonstrate intrusion detection system using any tool (snort or any other s/w).

Intrusion Detection with Snort: The general thought is that if a firewall is protecting one's network, the network is considered secure. However, that is not entirely true. Firewalls are a fundamental component of a network, but they cannot fully protect the network from forced entries or hostile intent. **Intrusion Detection Systems** are used to evaluate aggressive or unexpected packets and generate an alert before these programs can harm the network. A host-based Intrusion Detection System runs on all the devices in a network or connects to an organization's internal network. A network-based Intrusion Detection System is instead deployed at a certain point or group of points from which all ingoing and outgoing traffic can be monitored. An advantage of a host-based Intrusion Detection System is that it also can detect anomalies or malicious traffic being generated from the host itself, i.e., if the host is affected by malware, etc.

Intrusion Detection Systems (IDS) work by monitoring and analyzing network traffic and comparing it with an established ruleset, determining what should be taken as normal for the network (i.e., for ports, bandwidths, etc.) and what to take a closer look at.

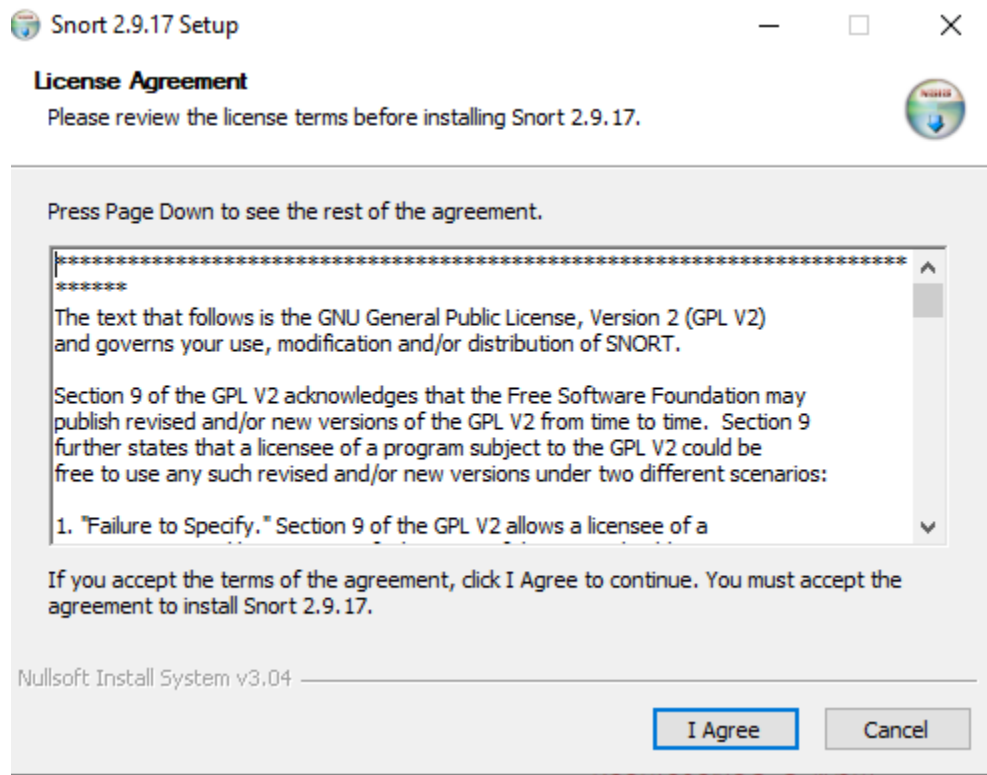
An Intrusion Detection System can be deployed depending upon the size of the network. There are dozens of quality commercial IDSs, but many companies and small businesses cannot afford them. **Snort** is a flexible, lightweight, and popular Intrusion Detection System that can be deployed according to the needs of the network, ranging from small to large networks, and provides all the features of a paid IDS. **Snort** does not cost anything but that does not mean that it cannot provide the same functionalities as an elite, commercial IDS.

Snort is considered a passive IDS, which means it sniffs network packets, compares with the ruleset, and, in the case of detecting a malicious log or entry (i.e., detecting an intrusion), generates an alert or places an entry in a log file.

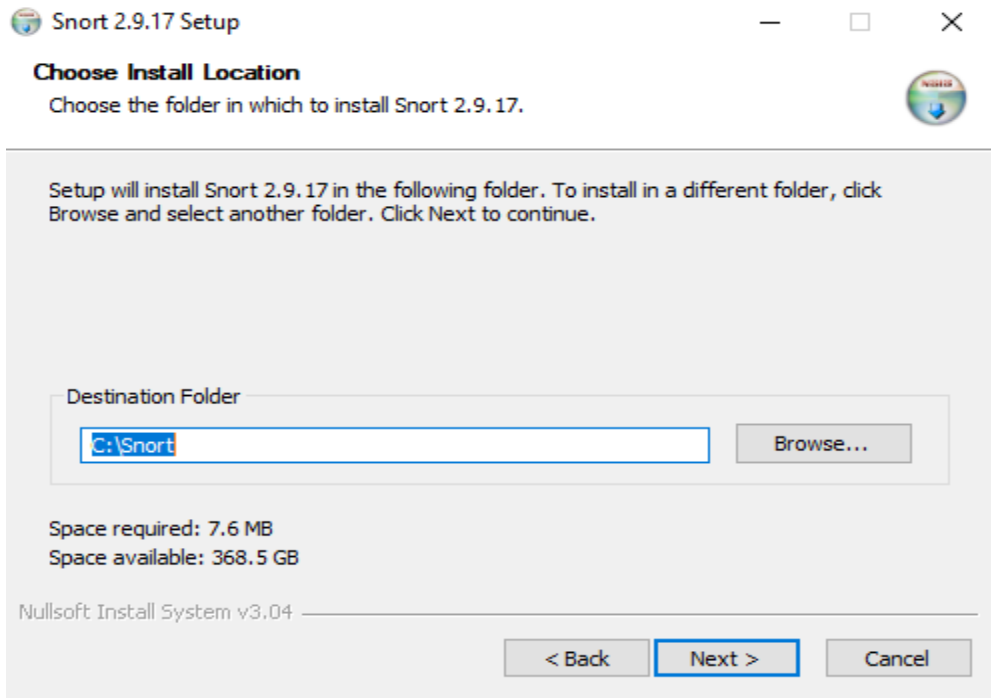
Snort is used for monitoring the operations and activities of routers, firewalls, and servers. Snort provides a user-friendly interface, containing a chain of rulesets that can be very helpful to a person who is unfamiliar with IDSs. Snort generates an alarm in case of an intrusion (buffer overflow attacks, DNS poisoning, OS fingerprinting, port scans, and much more), giving an organization greater visibility of the network traffic and making it much easier to meet security regulations.

Installing Snort on Windows 10 A Step By Step Process:

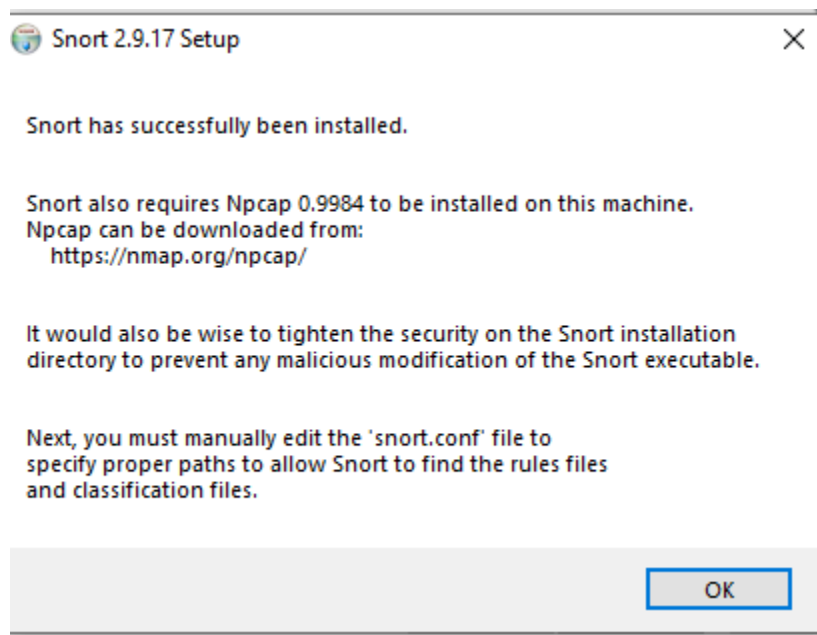
1. For Windows 10 64 bit supported SNORT's executable file can be downloaded from <https://www.snort.org/downloads>.
2. Open the downloaded snort executable file.
3. Click On 'I Agree' on the license agreement.



4. Choose components of Snort to be installed.
5. Click "Next" and then choose install location for snort preferably a separate folder in Windows C Drive.

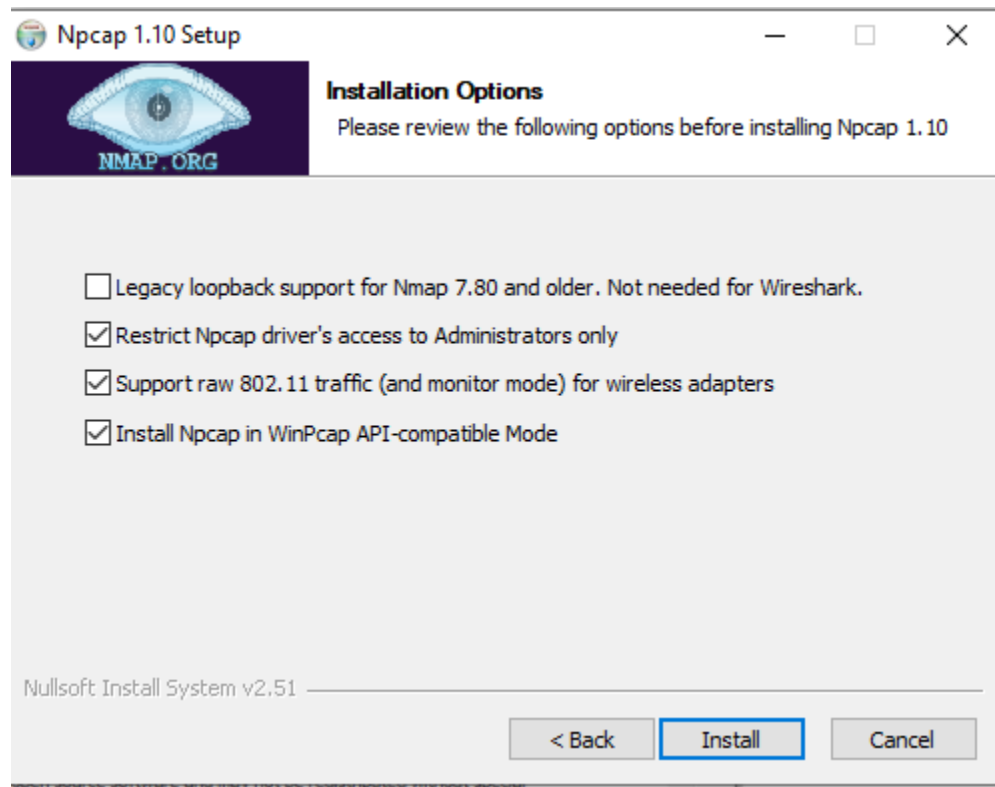


6. Click “Next” Installation process starts and then it completes
7. When you click “Close” you are prompted with this dialogue box:



8. Installing Npcap is required by snort for proper functioning.
9. Npcap for Windows 10 can be downloaded from <https://nmap.org/npcap/>
10. Opening Npcap setup file, Click on ‘I Agree’ To license agreement

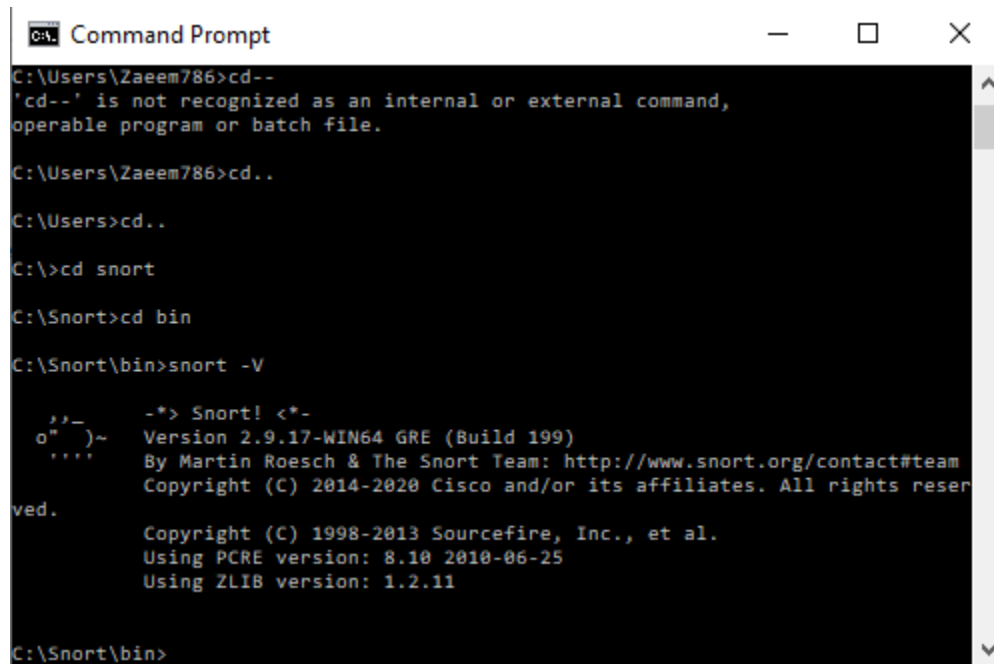
11. Now we proceed to choose which components of Npcap are to be installed and then clicking on “Install”.



12. Installation process starts and completes. Clicking on “Next”

13. Now the window for installation of Npcap shows it has been installed. Clicking “Finish”

14. After installing Snort and Npcap enter these commands in windows 10 Command prompt to check snorts working



```
Command Prompt
C:\Users\Zaeem786>cd--
'cd--' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\Zaeem786>cd..

C:\Users>cd..

C:\>cd snort

C:\Snort>cd bin

C:\Snort\bin>snort -V

_*> Snort! <*-
o" )~
' ' '
Version 2.9.17-WIN64 GRE (Build 199)
By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
Copyright (C) 2014-2020 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: 1.2.11

C:\Snort\bin>
```

15. As you can see in the above figure that snort runs successfully.

Configuring Snort 2.9.17 on Windows 10:

After installing Snort on Windows 10, Another important step to get started with Snort is configuring it on Windows 10.

Note: The italicized portion with a left hand side border states commands which were pre-written in the configuration file of Snort so we need to make changes according to the commands mentioned in the images, to be precise we need to enter configuration commands as shown in the images to configure snort.

1. Go to <https://www.snort.org/downloads/#rule-downloads> and download latest snort rule file.

2. Extract 3 folders from the downloaded snortrules-snapshot-29170.tar folder into the Snorts corresponding folders in C drive.

Folders to be extracted are: rules , preproc_rules , etc

- **rules folder** contains the rules files and the most important **local.rules** file. Which we will use to enter all our rules.
- **etc folder** contains all configuration files and the most important file is **snort.conf** file which we will use for configuration

3. Now open the **snort.conf** file through the notepad++ editor or any other text editor to edit configurations of snort to make it work like we want it to.

4. Setup the network addresses you are protecting

```
ipvar HOME_NET any
```

Note: Mention your own host IP addresses that you want to protect.

```
44 # Setup the network addresses you are protecting
45 ipvar HOME_NET 192.168.100.27/24
46
```

5. Setup the external network into anything that is not the home network. That is why ! is used in the command it denotes 'not'.

```
# Set up the external network addresses. Leave as "any" in most situationsipvar EXTERNAL_NET any
```

```
47 # Set up the external network addresses. Leave as "any" in most situations
48 ipvar EXTERNAL_NET !$HOME_NET
49
```

6. Now we have to define the directory for our rules and preproc rules folder

```
# Path to your rules files (this can be a relative path)# Note for Windows users: You are advised to make
this an absolute path,# such as: c:\snort\rulesvar RULE_PATH ../rulesvar SO_RULE_PATH
../so_rulesvar PREPROC_RULE_PATH ../preproc_rules
```

```
101 # Path to your rules files (this can be a relative path)
102 # Note for Windows users: You are advised to make this an absolute path,
103 # such as: c:\Snort\rules
104 var RULE_PATH c:\Snort\rules
105 # var SO_RULE_PATH ../so_rules
106 var PREPROC_RULE_PATH c:\Snort\preproc_rules
```

Setting up path to our rules files and preproc rules folder in Snort

7. Now we have to setup our white list and black list path it will be in our snorts' rule folder

```
# If you are using reputation preprocessor set thesevar WHITE_LIST_PATH ../rulesvar
BLACK_LIST_PATH ../rules
```

```
113 var WHITE_LIST_PATH c:\Snort\rules
114 var BLACK_LIST_PATH c:\Snort\rules
```

: Setting up our White List and Black List files paths in Snort

8. Next we have to enable to log directory, so that we store logs in our log folder. Uncomment this line and set absolute path to log directory

```
# Configure default log directory for snort to log to. For more information see snort -h command line
options (-l)## config logdir:
```

```
186 config logdir: c:\Snort\log
```

Setting up Log Directory Path in Snort

9. Now we will set the path to dynamic preprocessors and dynamic engine

```
# path to dynamic preprocessor libraries
```

```
dynamic preprocessor directory/usr/local/lib/snort_dynamicpreprocessor/
```

```
246 # path to dynamic preprocessor libraries
247 dynamicpreprocessor directory c:\Snort\lib\snort_dynamicpreprocessor
```

Setting up path to dynamic preprocessors and dynamic engine in Snort

10. We will do same thing for dynamic preprocessor engine

```
# path to base preprocessor engine dynamicengine /usr/local/lib/snort_dynamicengine/libs_f_engine.so
```

```
249 # path to base preprocessor engine
250 dynamicengine c:\Snort\lib\snort_dynamicengine\sf_engine.dll
```

Setting up the path to dynamic preprocessor engine in Snort

11. Now lets set our reputation preprocessors:

```
# path to dynamic rules libraries# dynamicdetection directory /usr/local/lib/snort_dynamicrules
```

```
252 # path to dynamic rules libraries
253 # dynamicdetection directory /usr/local/lib/snort_dynamicrules
```

Path to dynamic rules libraries in Snort

12. Just comment out these lines as shown in figure 19 in doing so we are excluding packet normalization of different packets.

```
263 # Inline packet normalization. For more information, see README.normalize
264 # Does nothing in IDS mode
265 # preprocessor normalize_ip4
266 # preprocessor normalize_tcp: ips ecn stream
267 # preprocessor normalize_icmp4
268 # preprocessor normalize_ip6
269 # preprocessor normalize_icmp6
```

Commenting out packet normalization commands in Snort

13. Scroll down to the reputation preprocessors. We will just change the name of the files since white list , black list are not rules they are just the list of IP addresses labelled as black or white

```
# Reputation preprocessor. For more information see README.reputationpreprocessor reputation:
\memcap 500, \priority whitelist, \nested_ip inner, \whitelist $WHITE_LIST_PATH/whitelist,
\blacklist $BLACK_LIST_PATH/black.list
```

```
511     whitelist $WHITE_LIST_PATH/white.list, \
512     blacklist $BLACK_LIST_PATH/black.list
```

Whitelisting and Blacklisting IPs through the command as shown in figure

14. Converted back slashes to forward slashes in lines 546–651.

```
545 # site specific rules
546 include $RULE_PATH\local.rules
547
548 include $RULE_PATH\app-detect.rules
549 include $RULE_PATH\attack-responses.rules
550 include $RULE_PATH\backdoor.rules
551 include $RULE_PATH\bad-traffic.rules
552 include $RULE_PATH\blacklist.rules
553 include $RULE_PATH\botnet-cnc.rules
554 include $RULE_PATH\browser-chrome.rules
555 include $RULE_PATH\browser-firefox.rules
556 include $RULE_PATH\browser-ie.rules
557 include $RULE_PATH\browser-other.rules
558 include $RULE_PATH\browser-plugins.rules
559 include $RULE_PATH\browser-webkit.rules
560 include $RULE_PATH\chat.rules
561 include $RULE_PATH\content-replace.rules
562 include $RULE_PATH\ddos.rules
563 include $RULE_PATH\dns.rules
564 include $RULE_PATH\dos.rules
565 include $RULE_PATH\experimental.rules
566 include $RULE_PATH\exploit-kit.rules
567 include $RULE_PATH\exploit.rules
568 include $RULE_PATH\file-executable.rules
569 include $RULE_PATH\file-flash.rules
570 include $RULE_PATH\file-identify.rules
571 include $RULE_PATH\file-image.rules
572 include $RULE_PATH\file-multimedia.rules
573 include $RULE_PATH\file-office.rules
574 include $RULE_PATH\file-other.rules
575 include $RULE_PATH\file-pdf.rules
576 include $RULE_PATH\finger.rules
577 include $RULE_PATH\ftp.rules
578 include $RULE_PATH\icmp-info.rules
579 include $RULE_PATH\icmp.rules
```

Converted back slashes to forward slashes in specific lines in snort.conf file

```

621 include $RULE_PATH\rservices.rules
622 include $RULE_PATH\scada.rules
623 include $RULE_PATH\scan.rules
624 include $RULE_PATH\server-apache.rules
625 include $RULE_PATH\server-iis.rules
626 include $RULE_PATH\server-mail.rules
627 include $RULE_PATH\server-mssql.rules
628 include $RULE_PATH\server-mysql.rules
629 include $RULE_PATH\server-oracle.rules
630 include $RULE_PATH\server-other.rules
631 include $RULE_PATH\server-webapp.rules
632 include $RULE_PATH\shellcode.rules
633 include $RULE_PATH\smtp.rules
634 include $RULE_PATH\snmp.rules
635 include $RULE_PATH\specific-threats.rules
636 include $RULE_PATH\spyware-put.rules
637 include $RULE_PATH\sql.rules
638 include $RULE_PATH\telnet.rules
639 include $RULE_PATH\tftp.rules
640 include $RULE_PATH\virus.rules
641 include $RULE_PATH\voip.rules
642 include $RULE_PATH\web-activex.rules
643 include $RULE_PATH\web-attacks.rules
644 include $RULE_PATH\web-cgi.rules
645 include $RULE_PATH\web-client.rules
646 include $RULE_PATH\web-coldfusion.rules
647 include $RULE_PATH\web-frontpage.rules
648 include $RULE_PATH\web-iis.rules
649 include $RULE_PATH\web-misc.rules
650 include $RULE_PATH\web-php.rules
651 include $RULE_PATH\xll.rules

```

Converted back slashes to forward slashes in specific lines in snort.conf file

15. Again just convert forward slashes to backslashes and uncomment the lines below:

```

# decoder and preprocessor event rules# include $PREPROC_RULE_PATH/preprocessor.rules#
include $PREPROC_RULE_PATH/decoder.rules# include $PREPROC_RULE_PATH/sensitive-
data.rules

```

```

657
658 # decoder and preprocessor event rules
659 include $PREPROC_RULE_PATH\preprocessor.rules
660 include $PREPROC_RULE_PATH\decoder.rules
661 include $PREPROC_RULE_PATH\sensitive-data.rules

```

Converted back slashes to forward slashes in specific lines and uncommenting specific lines in snort.conf file

16. Now we just need to verify the presence of this command at the bottom of **snort.conf** file.

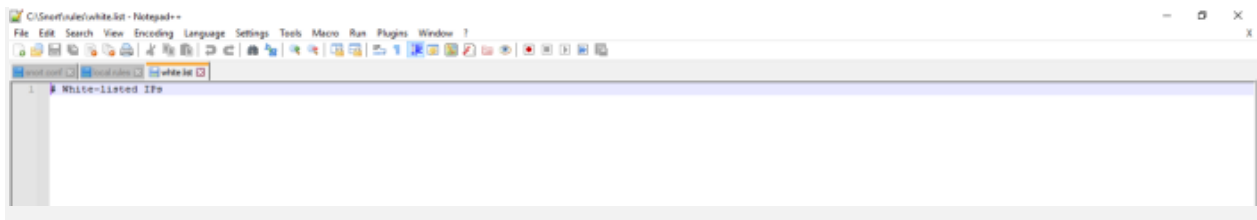
```
688 # Event thresholding or suppression commands. See threshold.conf
689 include threshold.conf
```

verifying presence of “include threshold.conf” command in snort.conf file

17. Click on Save file and save all changes to save the configuration file (**snort.conf**).

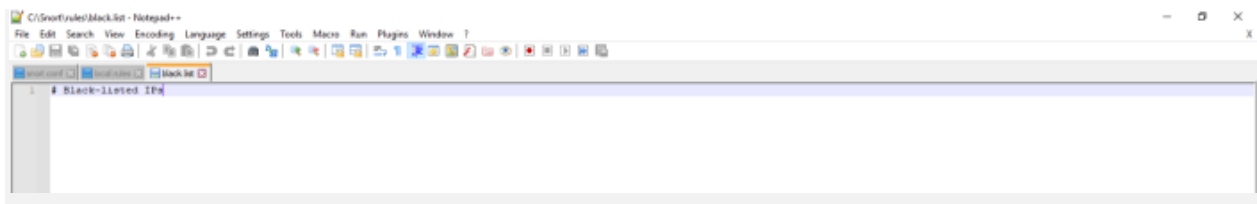
18. Now recalling the **Step 13** white list , black list are not rules they are just the list of IP addresses labelled as black or white right now these files don't exist in our rule path which is why we have to create them manually , save them in this folder **C:\Snort\rules**.

- Go to Notepad++ and create new file.
- Comment it #White-listed IPs.
- Name the file white.list and save the file.



Creating White List IPs file

- Create another new file.
- Comment it #Black-listed IPs.
- Name the file black.list and save the file.



Creating Black List IPs file in Snort

19. Now we test snort again by running Command prompt as admin. To check if it's running fine after all the configurations.



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.17763.1282]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\>cd..
C:\Users>cd..
C:\>cd snort
C:\Snort>cd bin
C:\Snort\bin>snort -V

  o"~
  ....
  -*) Snort! <*-
  Version 2.9.17-WIN32 GRE (Build 199)
  By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
  Copyright (C) 2014-2020 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

C:\Snort\bin>
```

Test Running of Snort in Windows 10 after Configuration

20. We can also check the wireless interface cards from which we will be using snort by using the command below. We can see the list of our wireless interface cards through entering this command in command prompt.

Snort — W

21. configuration validation check command:

Now we will enter a command to check validation of snort's configuration by choosing a specific wireless interface card (1). The rest of the command shows the config file path. The command is :

```
snort -i 1 -c C:\Snort\etc\snort.conf -T
```

```
---- Initialization Complete ----

o''~)~
'''~

-*> Snort! <*-
Version 2.9.17-WIN32 GRE (Build 199)
By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
Copyright (C) 2014-2020 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

Rules Engine: SF_SNORT_DETECTION_ENGINE Version 3.1 <Build 1>
Preprocessor Object: SF_SSLPP Version 1.1 <Build 4>
Preprocessor Object: SF_SSH Version 1.1 <Build 3>
Preprocessor Object: SF_SMTP Version 1.1 <Build 9>
Preprocessor Object: SF_SIP Version 1.1 <Build 1>
Preprocessor Object: SF_SDF Version 1.1 <Build 1>
Preprocessor Object: SF_REPUTATION Version 1.1 <Build 1>
Preprocessor Object: SF_POP Version 1.0 <Build 1>
Preprocessor Object: SF_MODBUS Version 1.1 <Build 1>
Preprocessor Object: SF_IMAP Version 1.0 <Build 1>
Preprocessor Object: SF_GTP Version 1.1 <Build 1>
Preprocessor Object: SF_FTPTELNET Version 1.2 <Build 13>
Preprocessor Object: SF_DNS Version 1.1 <Build 4>
Preprocessor Object: SF_DNP3 Version 1.1 <Build 1>
Preprocessor Object: SF_DCERPC2 Version 1.0 <Build 3>

Snort successfully validated the configuration!
Snort exiting

C:\Snort\bin>
```

Checking Validation of Snort Configuration in Command Prompt

It can be seen in the given figure that Snort successfully validates our configuration. This brings us to the end of our installation and configuration tutorial.

Experiment 8:

Demonstrate how to provide secure data storage, secure data transmission and for creating digital signatures.

INSTALLING THE SOFTWARE:

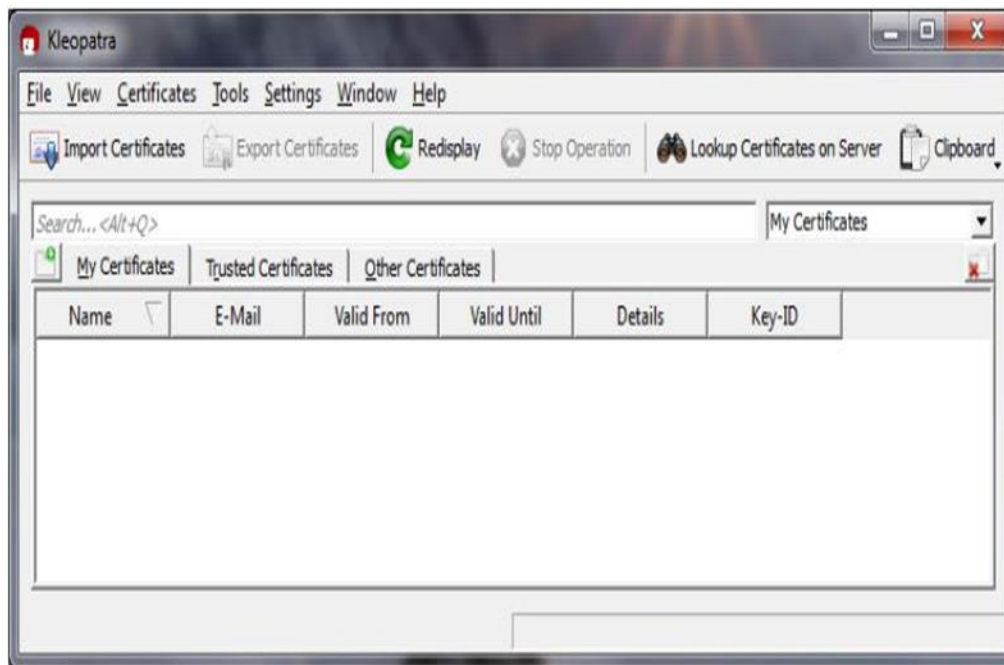
1. Visit www.gpg4win.org. Click on the “Gpg4win 2.3.0” button
2. On the screen, click the “Download Gpg4win” button.
3. When the “Welcome” screen is displayed, click the “Next” button
4. When the “License Agreement” page is displayed, click the “Next” button
5. Set the check box values as specified below, then click the “Next” button
6. Set the location where you want the software to be installed. The default location is fine. Then, click the “Next” button.
7. Specify where you want shortcuts to the software placed, then click the “Next” button.
8. If you selected to have a GPG shortcut in your Start Menu, specify the folder in which it will be placed. The default “Gpg4win” is OK. Click the “Install” button to continue
9. A warning will be displayed if you have Outlook or Explorer opened. If this occurs, click the “OK” button.
10. The installation process will tell you when it is complete. Click the “Next” button
11. Once the Gpg4win setup wizard is complete, the following screen will be displayed. Click the “Finish” button
12. If you do not uncheck the “Show the README file” check box, the README file will be displayed. The window can be closed after you’ve reviewed it.

CREATING YOUR PUBLIC AND PRIVATE KEYS

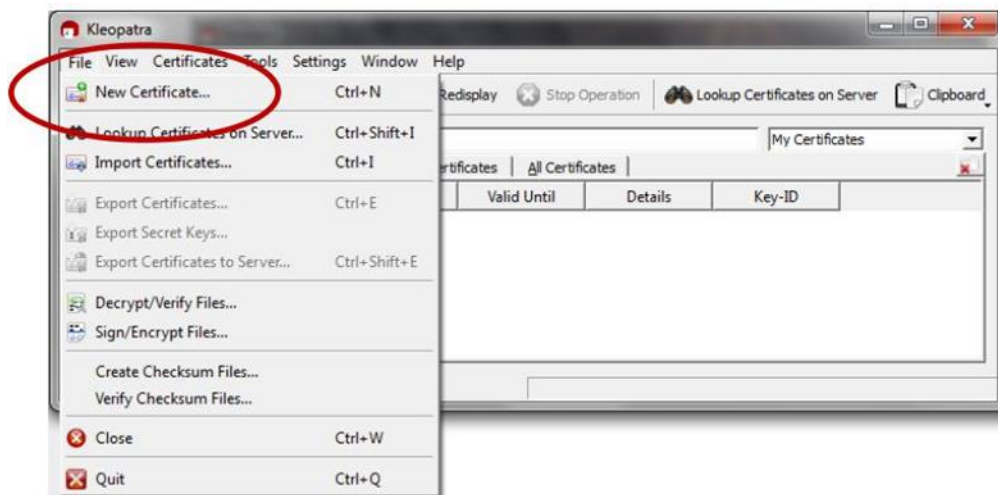
GPG encryption and decryption is based upon the keys of the person who will be receiving the encrypted file or message. Any individual who wants to send the person an encrypted file or message must possess the recipient’s public key certificate to encrypt the message. The recipient must have the associated private key, which is different than the public key, to be able to decrypt

the file. The public and private key pair for an individual is usually generated by the individual on his or her computer using the installed GPG program, called “Kleopatra” and the following procedure:

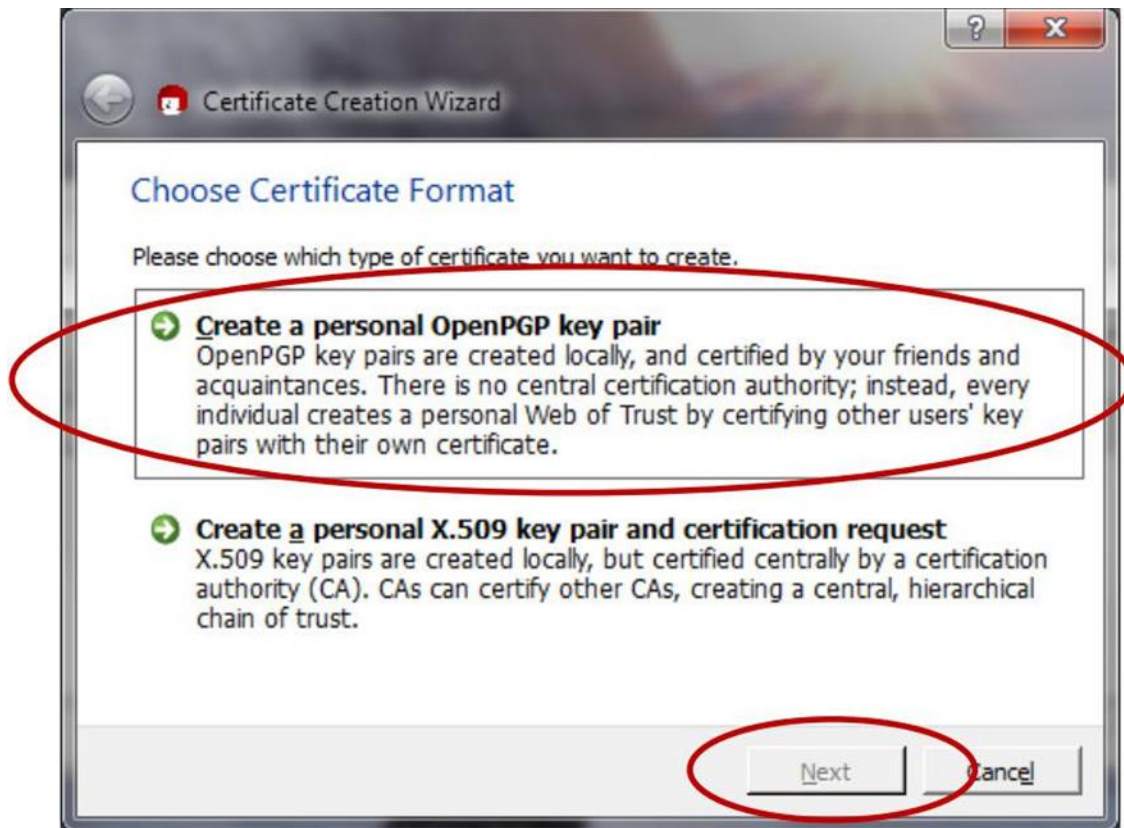
1. From your start bar, select the “Kleopatra” icon to start the Kleopatra certificate management software
2. The following screen will be displayed



3. From the “File” dropdown, click on the “New Certificate” option



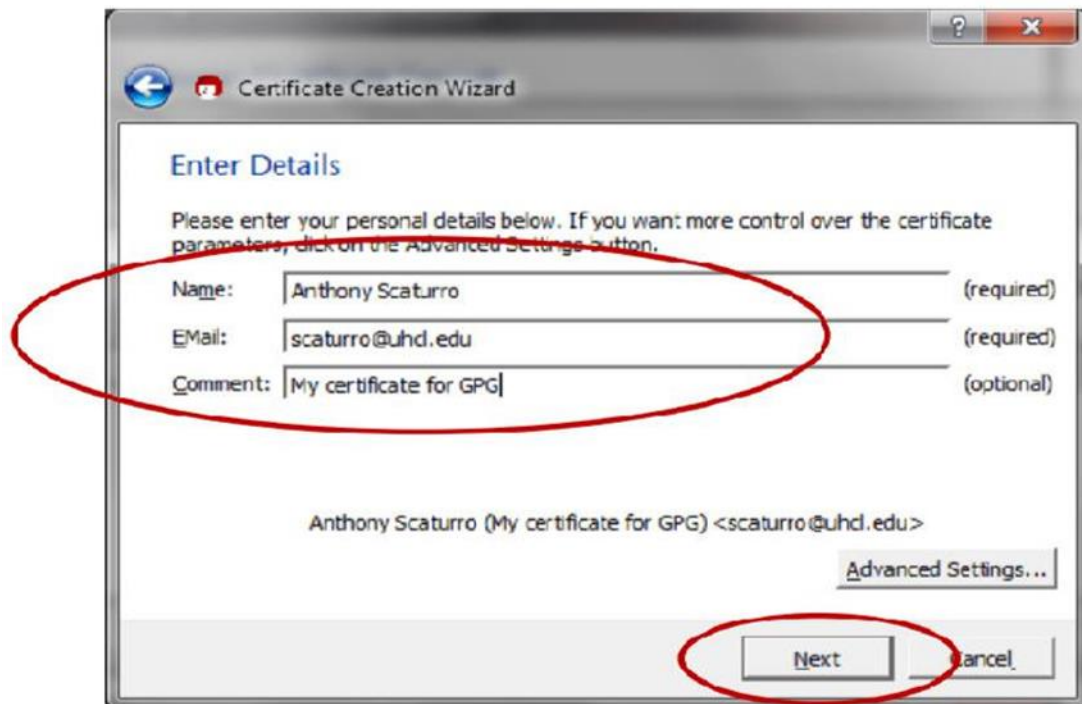
4. The following screen will be displayed. Click on “Create a personal OpenPGP key pair” and the “Next” button



5. The Certificate Creation Wizard will start and display the following:

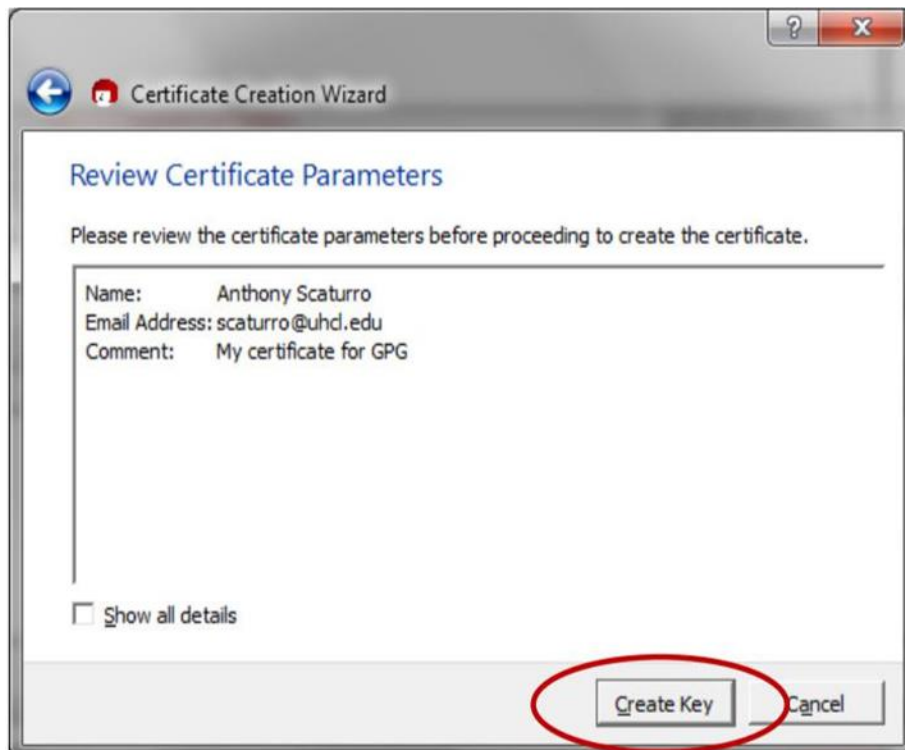
The screenshot shows the 'Enter Details' window of the Certificate Creation Wizard. The title bar includes a back arrow, a red icon, and the text 'Certificate Creation Wizard'. The main heading is 'Enter Details'. Below it, a message says 'Please enter your personal details below. If you want more control over the certificate parameters, click on the Advanced Settings button.' There are three input fields: 'Name:' (required), 'Email:' (required), and 'Comment:' (optional). The 'Name' field is empty, and a red error message 'Name is required, but empty.' is displayed below it. To the right of the fields is an 'Advanced Settings...' button. At the bottom right, the 'Next' button is highlighted with a black border, and the 'Cancel' button is also visible.

6. Enter your name and e-mail address. You may also enter an optional comment. Then, click the “Next” button



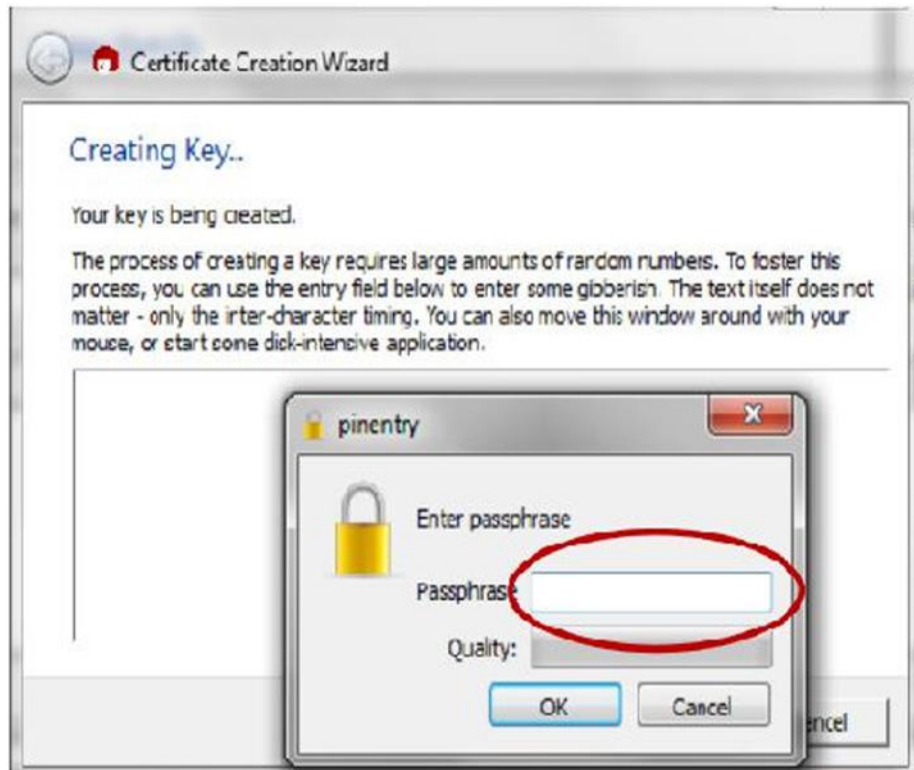
The screenshot shows the 'Enter Details' window of the 'Certificate Creation Wizard'. The window has a title bar with a question mark and a close button. Below the title bar is a back arrow icon and the text 'Certificate Creation Wizard'. The main heading is 'Enter Details'. Below this is a paragraph: 'Please enter your personal details below. If you want more control over the certificate parameters, click on the Advanced Settings button.' There are three input fields: 'Name:' with the value 'Anthony Scaturro' (required), 'Email:' with the value 'scaturro@uhd.edu' (required), and 'Comment:' with the value 'My certificate for GPG' (optional). Below these fields is a summary line: 'Anthony Scaturro (My certificate for GPG) <scaturro@uhd.edu>'. To the right of this line is a button labeled 'Advanced Settings...'. At the bottom right are two buttons: 'Next' and 'Cancel'. A red oval highlights the 'Name', 'Email', and 'Comment' input fields. Another red oval highlights the 'Next' button.

7. Review your entered values. If OK, click the “Create Key” button



The screenshot shows the 'Review Certificate Parameters' window of the 'Certificate Creation Wizard'. The window has a title bar with a question mark and a close button. Below the title bar is a back arrow icon and the text 'Certificate Creation Wizard'. The main heading is 'Review Certificate Parameters'. Below this is a paragraph: 'Please review the certificate parameters before proceeding to create the certificate.' There is a text box containing the following information: 'Name: Anthony Scaturro', 'Email Address: scaturro@uhd.edu', and 'Comment: My certificate for GPG'. Below the text box is a checkbox labeled 'Show all details'. At the bottom right are two buttons: 'Create Key' and 'Cancel'. A red oval highlights the 'Create Key' button.

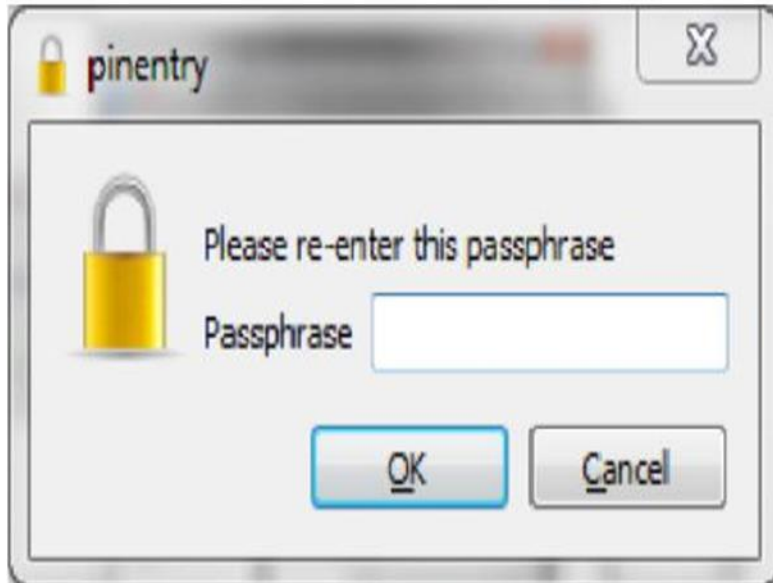
8. You will be asked to enter a passphrase



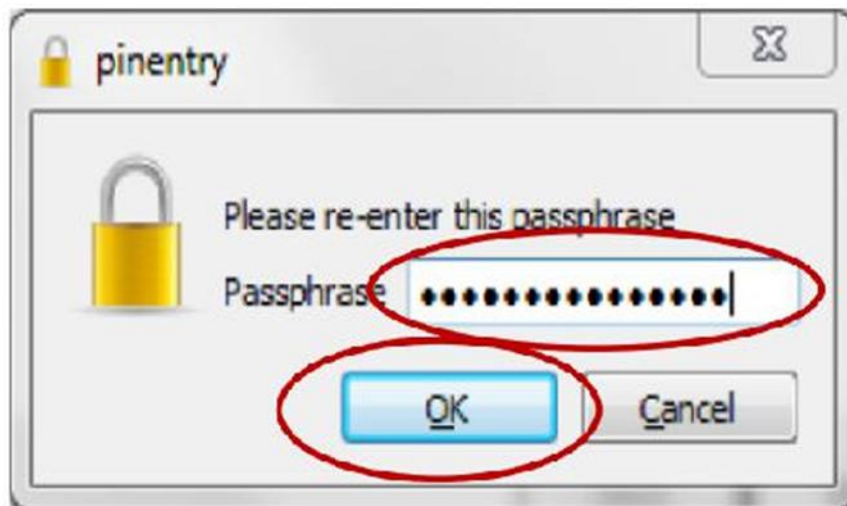
9. The passphrase should follow strong password standards. After you've entered your passphrase, click the "OK" button.



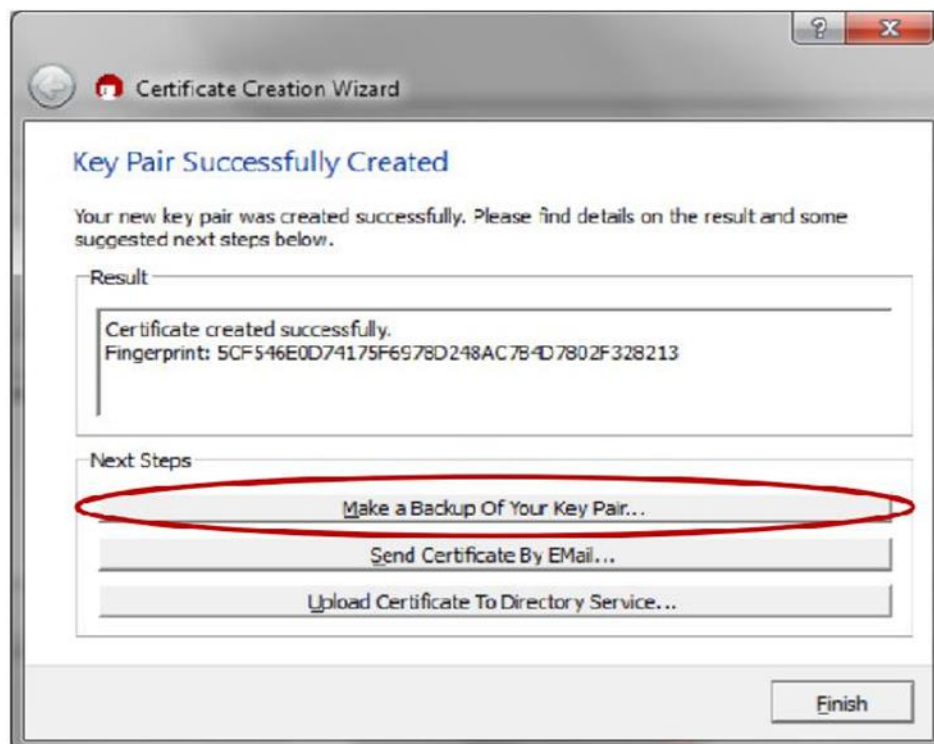
10. You will be asked to re-enter the passphrase



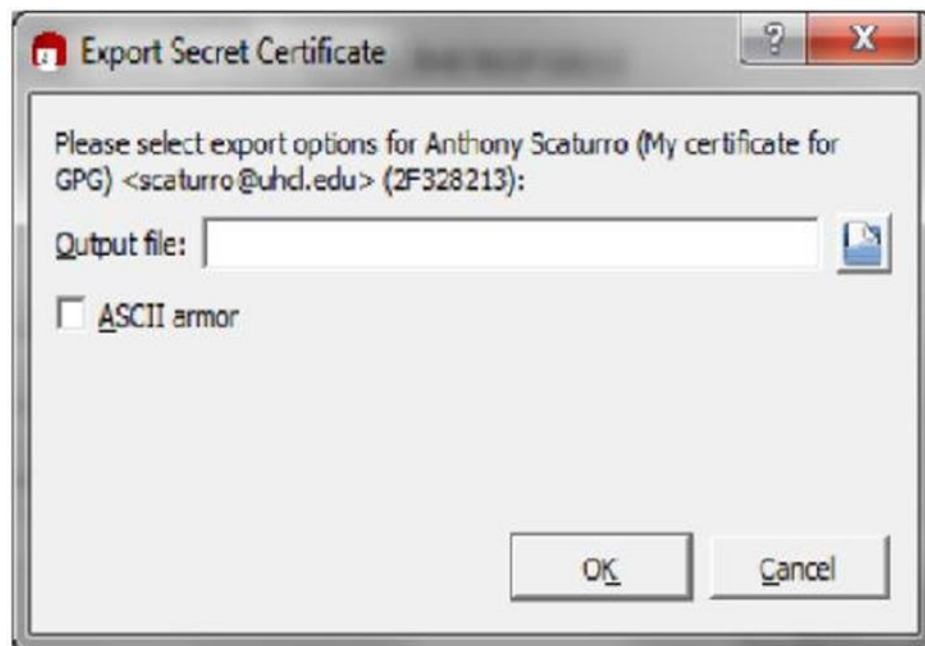
11. Re-enter the passphrase value. Then click the “OK” button. If the passphrases match, the certificate will be created.



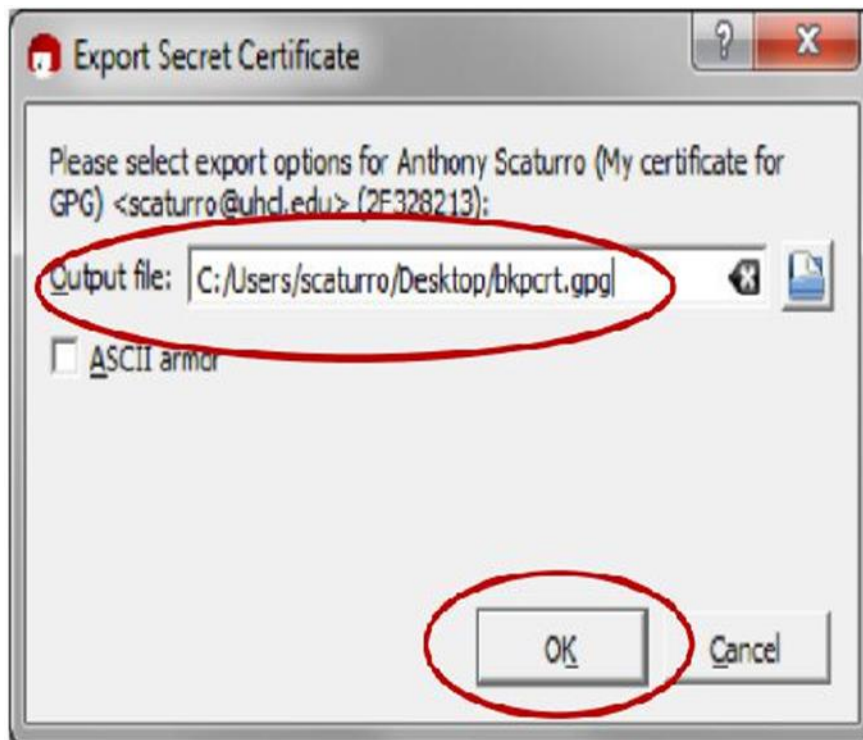
12. Once the certificate is created, the following screen will be displayed. You can save a backup of your public and private keys by clicking the “Make a backup Of Your Key Pair” button. This backup can be used to copy certificates onto other authorized computers.



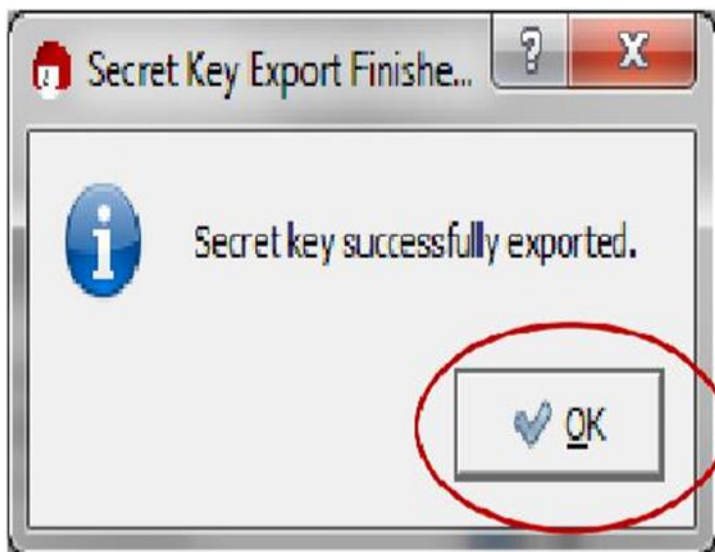
13. If you choose to backup your key pair, you will be presented with the following screen:



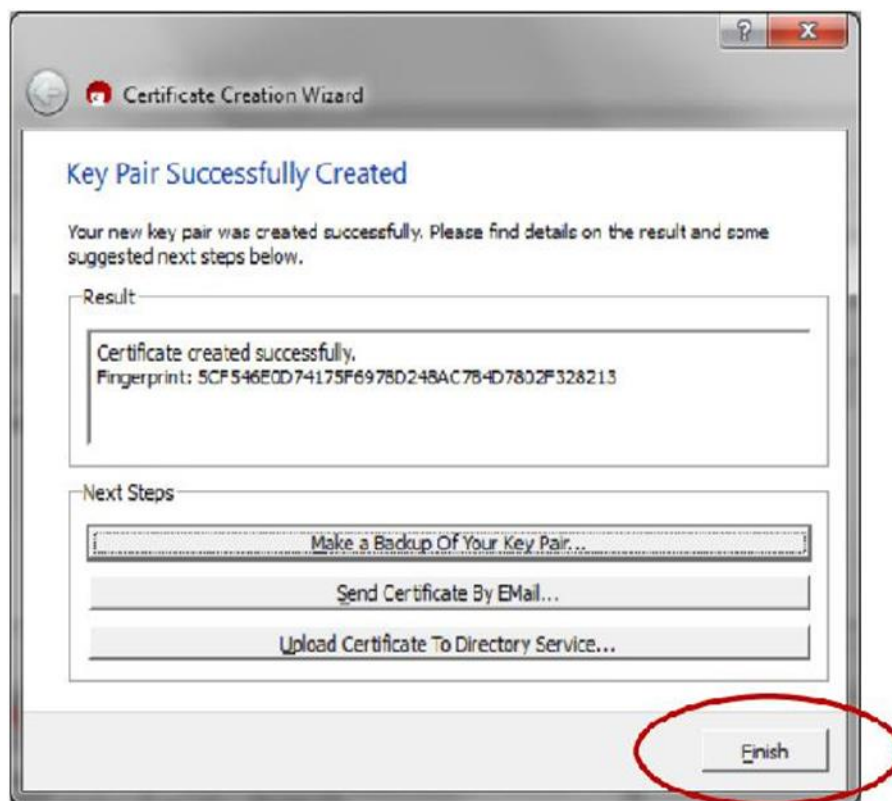
14. Specify the folder and name the file. Then click the "OK" button.



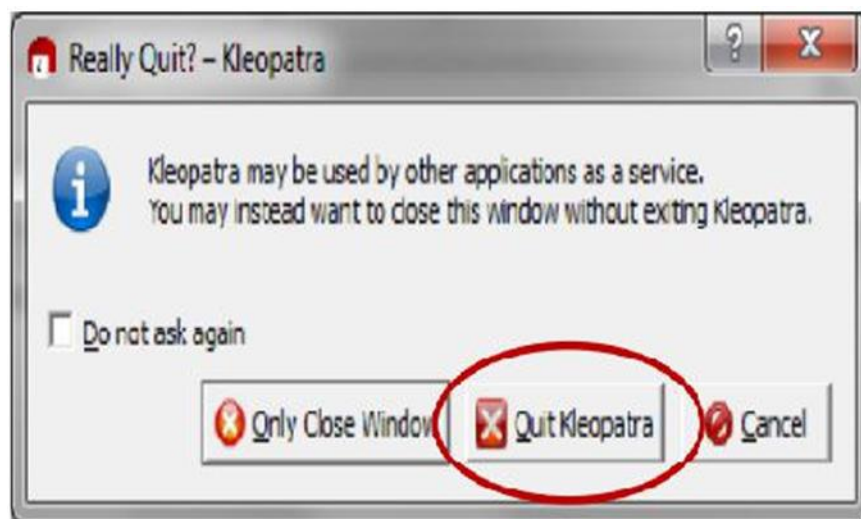
15. After the key is exported, the following will be displayed. Click the “OK” button.



16. You will be returned to the “Key Pair Successfully Created” screen. Click the “Finish” button.



17. Before the program closes, you will need to confirm that you want to close the program by clicking on the “Quit Kleopatra” button



Beyond The Syllabus

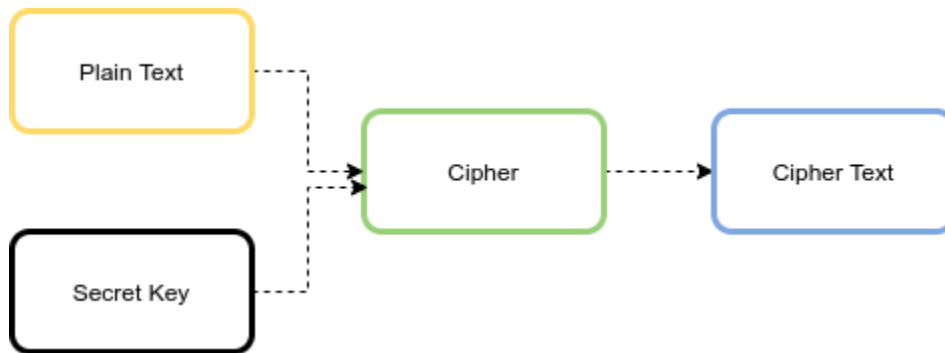
Experiment: 9

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

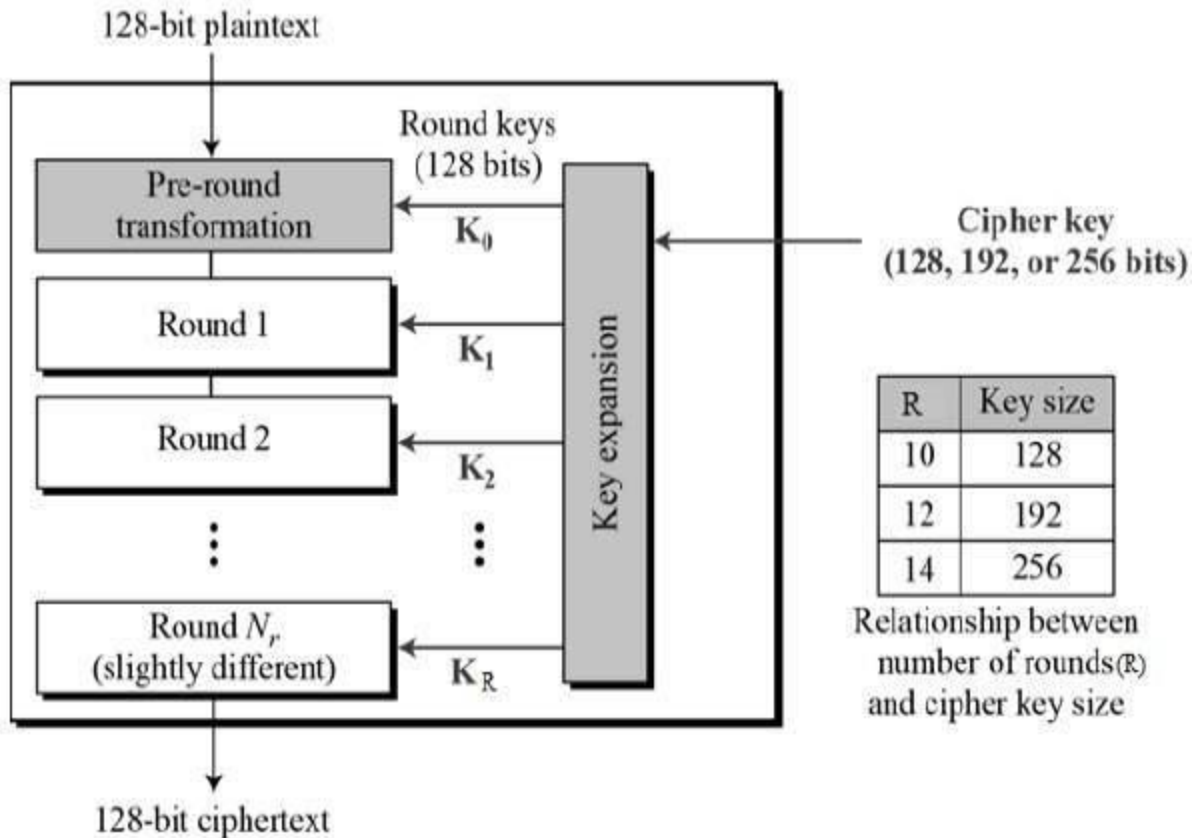
The symmetric-key block cipher plays an important role in data encryption. It means that the same key is used for both encryption and decryption. The Advanced Encryption Standard (AES) is a widely used symmetric-key encryption algorithm.

The AES algorithm is an iterative, symmetric-key block cipher that supports cryptographic keys (secret keys) of 128, 192, and 256 bits to encrypt and decrypt data in blocks of 128 bits.

The below figure shows the high-level AES algorithm:



The schematic of AES structure is given in the following illustration –



AES Analysis:

In present day cryptography, AES is widely adopted and supported in both hardware and software. Till date, no practical cryptanalytic attacks against AES has been discovered. Additionally, AES has built-in flexibility of key length, which allows a degree of 'future-proofing' against progress in the ability to perform exhaustive key searches.

However, just as for DES, the AES security is assured only if it is correctly implemented and good key management is employed.

Program:

AES Encryption and Decryption

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

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

public class AES {

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

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

    public static String encrypt(String strToEncrypt, String secret)
    {
        try
        {
            setKey(secret);
            Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
            cipher.init(Cipher.ENCRYPT_MODE, secretKey);
            return Base64.getEncoder().encodeToString(cipher.doFinal(strToEncrypt.getBytes("UTF-8")));
        }
        catch (Exception e)
        {
            System.out.println("Error while encrypting: " + e.toString());
        }
    }
}
```

```

        return null;
    }

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

```

Encryption and decryption example

```

public static void main(String[] args)
{
    final String secretKey = "ssshhhhhhhhhh!!!";

    String originalString = "howtodoinjava.com";
    String encryptedString = AES.encrypt(originalString, secretKey) ;
    String decryptedString = AES.decrypt(encryptedString, secretKey) ;

    System.out.println(originalString);
    System.out.println(encryptedString);
    System.out.println(decryptedString);
}

```

Beyond The Syllabus

Experiment: 10

Develop secure coding practices to handle Code Injection Vulnerabilities such as SQL Injection

What is SQL injection

SQL injection, also known as SQLI, is a common attack vector that uses malicious SQL code for backend database manipulation to access information that was not intended to be displayed. This information may include any number of items, including sensitive company data, user lists or private customer details.

The impact SQL injection can have on a business is far-reaching. A successful attack may result in the unauthorized viewing of user lists, the deletion of entire tables and, in certain cases, the attacker gaining administrative rights to a database, all of which are highly detrimental to a business.

When calculating the potential cost of an SQLi, it's important to consider the loss of customer trust should personal information such as phone numbers, addresses, and credit card details be stolen.

While this vector can be used to attack any SQL database, websites are the most frequent targets.

What are SQL queries

SQL is a standardized language used to access and manipulate databases to build customizable data views for each user. SQL queries are used to execute commands, such as data retrieval, updates, and record removal. Different SQL elements implement these tasks, e.g., queries using the SELECT statement to retrieve data, based on user-provided parameters.

SQL Injection

SQL injection is a code injection technique that might destroy your database.

SQL injection is one of the most common web hacking techniques.

SQL injection is the placement of malicious code in SQL statements, via web page input.

SQL in Web Pages

SQL injection usually occurs when you ask a user for input, like their username/userid, and instead of a name/id, the user gives you an SQL statement that you will **unknowingly** run on your database.

Look at the following example which creates a **SELECT** statement by adding a variable (txtUserId) to a select string. The variable is fetched from user input (getRequestString):

Example

```
txtUserId = getRequestString("UserId");  
txtSQL = "SELECT * FROM Users WHERE UserId = " + txtUserId;
```

The rest of this chapter describes the potential dangers of using user input in SQL statements.

SQL Injection Based on 1=1 is Always True

Look at the example above again. The original purpose of the code was to create an SQL statement to select a user, with a given user id.

If there is nothing to prevent a user from entering "wrong" input, the user can enter some "smart" input like this:

UserId:

Then, the SQL statement will look like this:

```
SELECT * FROM Users WHERE UserId = 105 OR 1=1;
```

The SQL above is valid and will return ALL rows from the "Users" table, since **OR 1=1** is always TRUE.

Does the example above look dangerous? What if the "Users" table contains names and passwords?

The SQL statement above is much the same as this:

```
SELECT UserId, Name, Password FROM Users WHERE UserId = 105 or 1=1;
```

A hacker might get access to all the user names and passwords in a database, by simply inserting 105 OR 1=1 into the input field.

SQL Injection Based on ""="" is Always True

Here is an example of a user login on a web site:

Username:

Password:

Example

```
uName = getQueryString("username");
```

```
uPass = getQueryString("userpassword");
```

```
sql = 'SELECT * FROM Users WHERE Name ='' + uName + '' AND Pass ='' + uPass + '''
```

Result

```
SELECT * FROM Users WHERE Name ="John Doe" AND Pass ="myPass"
```

A hacker might get access to user names and passwords in a database by simply inserting " OR ""=" into the user name or password text box:

User Name:

Password:

The code at the server will create a valid SQL statement like this:

Result

```
SELECT * FROM Users WHERE Name ="" or ""="" AND Pass ="" or ""=""
```

The SQL above is valid and will return all rows from the "Users" table, since **OR ""=""** is always TRUE.