

VIDYA PRATISHTHAN'S COLLEGE OF ENGINEERING
(Affiliated to University of Pune, Approved by AICTE New Delhi & Govt. of Maharashtra)
Vidyanagari, Baramati Dist. Pune

Information Assurance and Security Laboratory Lab Manual

Department of Information Technology
B.E (IT)



Prepared By:

Manik.K.Chavan

(M.Tech Computer Engineering)

Assistant Professor

Department of Information Technology

Practical: 1

RSA Algorithm

AIM: Write a program in C++ or Java to implement RSA algorithm for key generation and cipher verification.

Theory: In cryptography, **RSA** (which stands for Rivest, Shamir and Adleman who first publicly described it) is an algorithm for public-key cryptography.^[1] It is the first algorithm known to be suitable for signing as well as encryption, and was one of the first great advances in public key cryptography. RSA is widely used in electronic commerce protocols, and is believed to be sufficiently secure given sufficiently long keys and the use of up-to-date implementations.

Operation

The RSA algorithm involves three steps: key generation, encryption and decryption.

Key generation

RSA involves a **public key** and a **private key**. The public key can be known to everyone and is used for encrypting messages. Messages encrypted with the public key can only be decrypted using the private key. The keys for the RSA algorithm are generated the following way:

1. Choose two distinct prime numbers p and q .
 - For security purposes, the integers p and q should be chosen at random, and should be of similar bit-length. Prime integers can be efficiently found using a primality test.
2. Compute $n = pq$.
 - n is used as the modulus for both the public and private keys
3. Compute $\phi(n) = (p-1)(q-1)$, where ϕ is Euler's totient function.
4. Choose an integer e such that $1 < e < \phi(n)$ and $\gcd(e, \phi(n)) = 1$, i.e. e and $\phi(n)$ are coprime.
 - e is released as the public key exponent.
 - e having a short bit-length and small Hamming weight results in more efficient encryption - most commonly $0x10001 = 65537$. However, small values of e (such as 3) have been shown to be less secure in some settings.^[4]
5. Determine $d = e^{-1} \bmod \phi(n)$; i.e. d is the multiplicative inverse of $e \bmod \phi(n)$.
 - This is more clearly stated as solve for d given $(d \cdot e) \bmod \phi(n) = 1$
 - This is often computed using the extended Euclidean algorithm.
 - d is kept as the private key exponent.

The **public key** consists of the modulus n and the public (or encryption) exponent e . The **private key** consists of the private (or decryption) exponent d which must be kept secret.

Notes:

- An alternative, used by PKCS#1, is to choose d matching $de \equiv 1 \pmod{\lambda}$ with $\lambda = \text{lcm}(p-1, q-1)$, where lcm is the least common multiple. Using λ instead of $\phi(n)$ allows more choices for d . λ can also be defined using the Carmichael function, $\lambda(n)$.
- The ANSI X9.31 standard prescribes, IEEE 1363 describes, and PKCS#1 allows, that p and q match additional requirements: be strong primes, and be different enough that Fermat factorization fails.

Encryption

Alice transmits her public key (n, e) to Bob and keeps the private key secret. Bob then wishes to send message \mathbf{M} to Alice.

He first turns \mathbf{M} into an integer m , such that $0 < m < n$ by using an agreed-upon reversible protocol known as a padding scheme. He then computes the ciphertext c corresponding to

$$c = m^e \pmod{n}.$$

This can be done quickly using the method of exponentiation by squaring. Bob then transmits c to Alice.

Decryption

Alice can recover m from c by using her private key exponent d via computing

$$m = c^d \pmod{n}.$$

Given m , she can recover the original message \mathbf{M} by reversing the padding scheme.

A worked example

Here is an example of RSA encryption and decryption. The parameters used here are artificially small, but one can also use OpenSSL to generate and examine a real keypair.

1. Choose two distinct prime numbers, such as

$$p = 61 \text{ and } q = 53.$$

2. Compute $n = pq$ giving

$$n = 61 \cdot 53 = 3233.$$

3. Compute the totient of the product as $\phi(n) = (p-1)(q-1)$ giving

$$\phi(3233) = (61 - 1)(53 - 1) = 3120.$$

4. Choose any number $1 < e < 3120$ that is coprime to 3120. Choosing a prime number for e leaves us only to check that e is not a divisor of 3120.

Let $e = 17$.

5. Compute d , the modular multiplicative inverse of $e \pmod{\phi(n)}$, yielding

$$d = 2753.$$

The **public key** is $(n = 3233, e = 17)$. For a padded plaintext message m , the encryption function is $m^{17} \pmod{3233}$.

The **private key** is $(n = 3233, d = 2753)$. For an encrypted ciphertext c , the decryption function is $c^{2753} \pmod{3233}$.

For instance, in order to encrypt $m = 65$, we calculate

$$c = 65^{17} \pmod{3233} = 2790.$$

To decrypt $c = 2790$, we calculate

$$m = 2790^{2753} \pmod{3233} = 65.$$

Both of these calculations can be computed efficiently using the square-and-multiply algorithm for modular exponentiation. In real life situations the primes selected would be much larger; in our example it would be relatively trivial to factor n , 3233, obtained from the freely available public key back to the primes p and q . Given e , also from the public key, we could then compute d and so acquire the private key.

Practical implementations use Chinese Remainder theorem to speed up the calculation using modulus of factors ($\pmod{p \cdot q}$ using \pmod{p} and \pmod{q}).

The values dp , dq and $qInv$, which are part of the private key are computed as follows:

- $dp = d \pmod{p - 1} = 2753 \pmod{61 - 1} = 53$
- $dq = d \pmod{q - 1} = 2753 \pmod{53 - 1} = 49$
- $qInv = q^{-1} \pmod{p} = 53^{-1} \pmod{61} = 38$ (Hence: $qInv * q \pmod{p} = 38 * 53 \pmod{61} = 1$)

Here is how dp, dq and qInv are used for efficient decryption. (Encryption is efficient by choice of public exponent e)

- $m1 = c^{dp} \bmod p = 2790^{53} \bmod 61 = 4$
- $m2 = c^{dq} \bmod q = 2790^{49} \bmod 53 = 12$
- $h = (qInv * (m1 - m2)) \bmod p = (38 * -8) \bmod 61 = 1$
- $m = m2 + h * q = 12 + 1 * 53 = 65$ (same as above but computed more efficiently)

```
/* C program for the Implementation Of RSA Algorithm */
```

```
#include< stdio.h>
```

```
#include< conio.h>
```

```
int phi,M,n,e,d,C,FLAG;
```

```
int check()
```

```
{
    int i;
    for(i=3;e%i==0 && phi%i==0;i+2)
    {
        FLAG = 1;
        return;
    }
    FLAG = 0;
}
```

```
void encrypt()
```

```
{
    int i;
    C = 1;
    for(i=0;i< e;i++)
        C=C*M%n;
    C = C%n;
    printf("\n\tEncrypted keyword : %d",C);
}
```

```
void decrypt()
```

```
{
    int i;
    M = 1;
    for(i=0;i< d;i++)
```

```

        M=M*C%n;
    M = M%n;
    printf("\n\tDecrypted keyword : %d",M);
}

void main()
{
    int p,q,s;
    clrscr();
    printf("Enter Two Relatively Prime Numbers\t: ");
    scanf("%d%d",&p,&q);
    n = p*q;
    phi=(p-1)*(q-1);
    printf("\n\tF(n)\t= %d",phi);
    do
    {
        printf("\n\nEnter e\t: ");
        scanf("%d",&e);
        check();
    }while(FLAG==1);
    d = 1;
    do
    {
        s = (d*e)%phi;
        d++;
    }while(s!=1);
    d = d-1;
    printf("\n\tPublic Key\t: { %d,%d}",e,n);
    printf("\n\tPrivate Key\t: { %d,%d}",d,n);
    printf("\n\nEnter The Plain Text\t: ");
    scanf("%d",&M);
    encrypt();
    printf("\n\nEnter the Cipher text\t: ");
    scanf("%d",&C);
    decrypt();
    getch();
}

```

/****** OUTPUT *****/

Enter Two Relatively Prime Numbers : 7 17

$F(n) = 96$

Enter e : 5

Public Key : {5,119}

Private Key : {77,119}

Enter The Plain Text : 19

Encrypted keyword : 66

Enter the Cipher text : 66

Decrypted keyword : 19 */

Practical: 2

Client Server Authentication Verification

Aim: Write a Client – Server program in C++ or Java for authentication verification.

Theory: Client/server describes the relationship between two computer programs in which one program, the client, makes a service request from another program, the server, which fulfills the request. Although the client/server idea can be used by programs within a single computer, it is a more important idea in a network. In a network, the client/server model provides a convenient way to interconnect programs that are distributed efficiently across different locations. Computer transactions using the client/server model are very common. For example, to check your bank account from your computer, a client program in your computer forwards your request to a server program at the bank. That program may in turn forward the request to its own client program that sends a request to a database server at another bank computer to retrieve your account balance. The balance is returned back to the bank data client, which in turn serves it back to the client in your personal computer, which displays the information for you.

Authentication

The authentication service is concerned with assuring that a communication is authentic. In the case of a single message, such as a warning or alarm signal, the function of the authentication service is to assure the recipient that the message is from the source that it claims to be from. In the case of an ongoing interaction, such as the connection of a terminal to a host, two aspects are involved. First, at the time of connection initiation, the service assures that the two entities are authentic, that is, that each is the entity that it claims to be. Second, the service must assure that the connection is not interfered with in such a way that a third party can masquerade as one of the two legitimate parties for the purposes of unauthorized transmission or reception.

Two specific authentication services are defined in X.800:

Peer entity authentication: Provides for the corroboration of the identity of a peer entity in an association. It is provided for use at the establishment of, or at times during the data transfer phase of, a connection. It attempts to provide confidence that an entity is not performing either a masquerade or an unauthorized replay of a previous connection.

Data origin authentication: Provides for the corroboration of the source of a data unit. It does not provide protection against the duplication or modification of data units. This type of service supports applications like electronic mail where there are no prior interactions between the communicating entities.

Client Server Authentication Program in java

```
import java.io.*;
import java.net.*;
class Server_1
{
    public static String user = new String("Jyoti");
    public static String password = new String("MIT");
    public static void main(String ar[])throws Exception
    {
        ServerSocket ss=new ServerSocket(10000);
        Socket s=ss.accept();
        DataInputStream temp_1=new DataInputStream(s.getInputStream());
        String sUNAME=temp_1.readLine();

        DataInputStream temp_2=new DataInputStream(s.getInputStream());
        String sPSWD=temp_2.readLine();

        PrintStream ps=new PrintStream(s.getOutputStream());
        if (user.equals(sUNAME) && password.equals(sPSWD)) {
            ps.println("Success.");
        }
        else {
            ps.println("Failed.");
        }
    }
}
```

```
import java.io.*;
import java.net.*;
class Client_1
{
    public static void main(String ar[])throws Exception
    {
        Socket s=new Socket(InetAddress.getLocalHost(),10000);

        System.out.println("enter the Username :");
        DataInputStream uname=new DataInputStream(System.in);
        String sUNAME=uname.readLine();
```

```
System.out.println("enter the password :");  
DataInputStream pswd=new DataInputStream(System.in);  
String sPSWD=pswd.readLine();
```

```
PrintStream ps=new PrintStream(s.getOutputStream());  
ps.println(sUNAME);  
ps.println(sPSWD);
```

```
DataInputStream dis1=new DataInputStream(s.getInputStream());  
System.out.println("password status:"+dis1.readLine());
```

```
}
```

```
}
```

Practical: 3

The Extended Euclidean Algorithm

Aim: Develop a program in C++ or Java based on number theory such as Chinese remainder or Extended Euclidean algorithm. (Or any other to illustrate number theory for security)

Theory:

The *Extended Euclidean Algorithm* is just a fancier way of using the Euclidean algorithm above. It involves using extra variables to compute $ax + by = \gcd(a, b)$ as we go through the Euclidean algorithm in a single pass. It's more efficient to use in a computer program.

Euclidean algorithm

The *Euclidean algorithm* is an efficient method to compute the *greatest common divisor* (gcd) of two integers. It was first published in Book VII of Euclid's *Elements* sometime around 300 BC.

We write $\gcd(a, b) = d$ to mean that d is the largest number that will divide both a and b . If $\gcd(a, b) = 1$ then we say that a and b are *coprime* or *relatively prime*. The gcd is sometimes called the *highest common factor* (hcf).

Algorithm: (Euclidean algorithm) Computing the greatest common divisor of two integers.

INPUT: Two non-negative integers a and b with $a \geq b$.
OUTPUT: $\gcd(a, b)$.

1. While $b > 0$, do
 - a. Set $r = a \bmod b$,
 - b. $a = b$,
 - c. $b = r$
2. Return a .
3. **Question 1(a):** Find $\gcd(421, 111)$.
4. **Answer:**

We use the Euclidean algorithm as follows:

$$421 = 111 \times 3 + 88 \quad (\text{larger number on left})$$

$$111 = 88 \times 1 + 23 \quad (\text{shift left})$$

$$88 = 23 \times 3 + 19 \quad (\text{note how 19 moves down the "diagonal"})$$

$$23 = 19 \times 1 + 4$$

$$19 = 4 \times 4 + 3$$

$$4 = 3 \times 1 + 1 \quad (\text{last non-zero remainder is 1})$$

$$3 = 1 \times 3 + 0$$

5. The last non-zero remainder is 1 and therefore $\gcd(421, 111) = 1$.

The Extended Euclidean Algorithm

The *Extended Euclidean Algorithm* is just a fancier way of doing what we did [Using the Euclidean algorithm](#) above. It involves using extra variables to compute $ax + by = \gcd(a, b)$ as we go through the Euclidean algorithm in a single pass. It's more efficient to use in a computer program. If you are doing a calculation by hand, honestly, it's simpler just to use the [method above](#).

Algorithm: Extended Euclidean algorithm.

INPUT: Two non-negative integers a and b with $a \geq b$.
OUTPUT: $d = \gcd(a, b)$ and integers x and y satisfying $ax + by = d$.

1. If $b = 0$ then set $d = a$, $x = 1$, $y = 0$, and return(d, x, y).
2. Set $x_2 = 1$, $x_1 = 0$, $y_2 = 0$, $y_1 = 1$
3. While $b > 0$, do
 - a. $q = \text{floor}(a/b)$, $r = a - qb$, $x = x_2 - qx_1$, $y = y_2 - qy_1$.
 - b. $a = b$, $b = r$, $x_2 = x_1$, $x_1 = x$, $y_2 = y_1$, $y_1 = y$.
4. Set $d = a$, $x = x_2$, $y = y_2$, and return(d, x, y).

with input $a = 4864$, $b = 3458$ we get following values

q	r	x	y	a	b	x ₂	x ₁	y ₂	y ₁
1	1406	1	-1	3458	1406	0	1	1	-1
2	646	-2	3	1406	646	1	-2	-1	3
2	114	5	-7	646	114	-2	5	3	-7
5	76	-27	38	114	76	5	-27	-7	38
1	38	32	-45	76	38	-27	32	38	-45
2	0	-91	128	38	0	32	-91	-45	128

$x = 32$ $y = -45$ $d = 38$

That is, $\gcd(4864, 3458) = 38$ and $32 \times 4864 - 45 \times 3458 = 38$.

**// Program to implement Algorithm for Extended Euclidian Algorithm to find GCD
// of two numbers.**

```
#include<stdio.h>
#include<conio.h>
void main()
{

    int a,A,b,B,c,d,m,n,q,r=1;
    int t,x;
```

```

printf("\n\n\n\nThis program will find the Greatest Common Factor of two
numbers\n");
printf("m = ");
scanf("%d",&m);
printf("n = ");
scanf("%d",&n);
printf("\n\nWorking numbers\n\n");
printf("A\t a\t B\t b\t c\t d\t q\t r\n");
printf("-----\n");

A=b=1;
a=B=0;
c=m;
d=n;
//a' <- b <- 1, a <- b' <- 0, c <- m, d <- n

while(r!=0)
{
    q=(c/d);
    r=(c%d);

    printf("%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\n",A,a,B,b,c,d,q,r);

    x=d;
    c=d;
    d=r;
    t=A;
    A=a;
    a=(t-(q*a));
    t=B;
    B=b;
    b=(t-(q*b));
}

printf("-----\n");
printf("\nThe Greatest Common Divisor of %d and %d is %d\n\n", m, n, x);

//return 0;
}

```

Practical: 4

Snort IDS

AIM: Configure and demonstrate use of IDS tool such as snort.

Theory: In this lab, we will use a Network Intrusion Detection System (NIDS) to detect attacks through the network. Our tool of choice is [Snort](#), an open source signature-based NIDS (remind yourself about the differences between anomaly-based and signature-based intrusion detection systems).

- 1) **Architecture of IDS**
- 2) **Misuse based IDS**
- 3) **Anomaly based IDS**

Note: Explain each of the above in detail.

Step 1:

Install and configure snort

NOTE: Write down the steps for installation and configuration of snort for WindowsXP.

\$sudo apt-get install snort

Read the comments in `/etc/snort/snort.conf` carefully and pay attention to the definition of variable `HOME_NET` and `EXTERNAL_NET`. Then try running Snort as root:

\$sudo snort -c /etc/snort/snort.conf

Watch the output carefully, and address any errors in your config file. (Hint: some default `.rules` files contain deprecated format, try to comment those files in the config file). Continue re-running snort until you get it working correctly.

Step 2:

Read about Snort's signature syntax in the Snort User's Manual. In particular, be sure to review the metadata options reference and `sid`. Once you are somewhat familiar with the rule language, read through some of the web attacks rules files. These are files named in the form `web-*.rules` under `/etc/snort/rules/`.

Follow the references listed in a few of the rules and read about the type of attack the specific signatures are designed to detect.

Now, select two web attack signatures that seem straight-forward to understand. It would be simpler if you select a signature that looks for "evil" data in an HTTP URL string. Log into your Windows server and open a browser. Based on the documentation provided with the signature you have selected, attempt to trigger the Snort signature by making a HTTP request to which contains an attack string which should be detected.

Next, verify in your Snort logs that your attack triggered an alert based on that. (Hint: `/var/log/snort/`)

Step 3:

Snort also allows us to write custom rules. Open the file `/etc/snort/rules/local.rules` and add one rule that detects each visit to `www.google.com` that is made by the virtual machine.

The rule should look for any outbound TCP traffic that is going to port 80 and contains the pattern "`www.google.com`" in the URL and trigger an alert when it gets a match. Give the rule an SID of 1000000 or higher. Then visit Google with a web browser and check if your rule triggered an alert.

Questions:

1. In step 1, how did you modify the config file to make it work?
2. In step 2, describe the two attack signatures you chose and explain the corresponding rules against them.

Practical: 5

Ethereal Packet Sniffer and Analysis Tool

AIM: Configure and demonstrate use of Traffic monitoring tool such as Wireshark with security perspective.

Introduction

Packet sniffers, sometimes referred to as protocol or network analyzers, are invaluable tools for network and systems administrators. Ethereal is a free packet sniffer that not only decodes network traffic, but can also filter and analyze it. Additionally, Ethereal can read the data files from a multitude of other packet sniffers, letting you analyze previously collected data. The files can even be compressed with gzip, and Ethereal will read and write to them invisibly. Ethereal currently runs on most UNIX platforms and various Windows platforms. In Linux, it has an advanced, GTK-based GUI. Ethereal works on the following Windows platforms:

- Windows ME / 98 /95
- Windows Server 2003/ XP / 2000 / NT 4.0

Here are some examples people use Ethereal for:

- Network administrators use it to **troubleshoot network problems**
- Network security engineers use it to **examine security problems**
- Developers use it to **debug protocol implementations**
- People use it to **learn network protocol** internals

2.0 Installation

Ethereal can be downloaded from the main Web site:

<http://www.ethereal.com/download.html>

or from any of their mirrors worldwide. Installing Ethereal under Windows from the binary packages requires two steps:

1. Install WinPcap. You will find a single installer exe called something like "auto-installer", which can be installed under various Windows systems, including 9x/Me/NT4.0/2000/XP. This installer is located at:

<http://winpcap.polito.it/install/Default.htm>

2. Install Ethereal. You may acquire a binary installable of Ethereal at

<http://www.ethereal.com/download.html#binaries>

Download the installer and execute it.

3.0 Using Ethereal

3.1 Starting it up

After compiling the software (or installing the binary packages available), the program can be started by double-clicking on the icon. If a user who is not the root is allowed to run the program with root privileges, he can not only capture data on the network, but should any security holes be found in Ethereal, an attacker could gain control through it. Once Ethereal starts up, a standard packet-sniffer GUI with three panes is seen:

- (i) The Packet-List pane, where captured packets are displayed. Each line in the packet list corresponds to one packet in the capture file. If you select a line in this pane, more details will be displayed in the "Packet Details" and "Packet Bytes" panes. While dissecting a packet, Ethereal will place information from the protocol dissectors into the columns. As higher level protocols might overwrite information from lower levels, you will typically see the information from the highest possible level only.
- (ii) The Packet-Details pane, which contains the protocol tree for the currently selected packet and displays each field and value for the packet. This pane shows the protocols and protocol fields of the packet selected in the "Packet List" pane. The protocols and fields of the packet are displayed using a tree, which can be expanded and collapsed.
- (iii) The Packet-Bytes pane, which contains a hex dump of the selected packet. As usual for a hexdump, the left side shows the offset in the packet data, in the middle the packet data is shown in a hexadecimal representation and on the right the corresponding ASCII characters (or . if not appropriate) are displayed.

There is also a small text-entry box above the top pane, with the title "Filter". It is used to provide filters for displaying the packets (Figure 1).

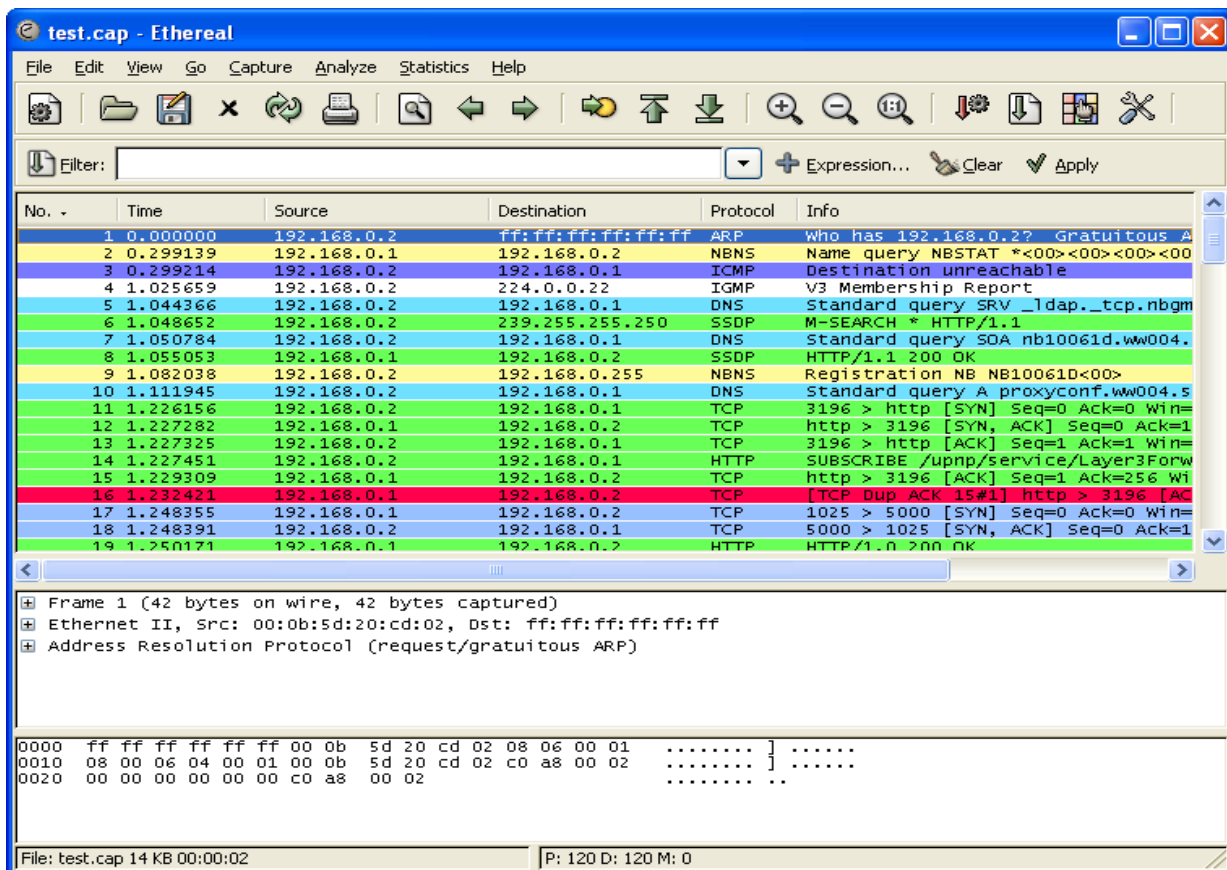


Figure 1

3.2 Capture Options

To capture packets, Ethereal must put your Ethernet card into promiscuous mode, which is why it must be executed with root user privileges. (Some cards do not support promiscuous mode, but these are extremely rare).

Putting a card into promiscuous mode and capturing traffic on the network may not only set off various intrusion detection systems (IDS), but may cause general discontent to the network administrator. The presence of a sniffer can be a sign that one of the machines in the network has been cracked. So before proceeding, make sure you have the system or network administrator's permission.

To start capturing, go to the "Capture" menu and select "Start". A new dialog box will appear asking for information (Figure 2).

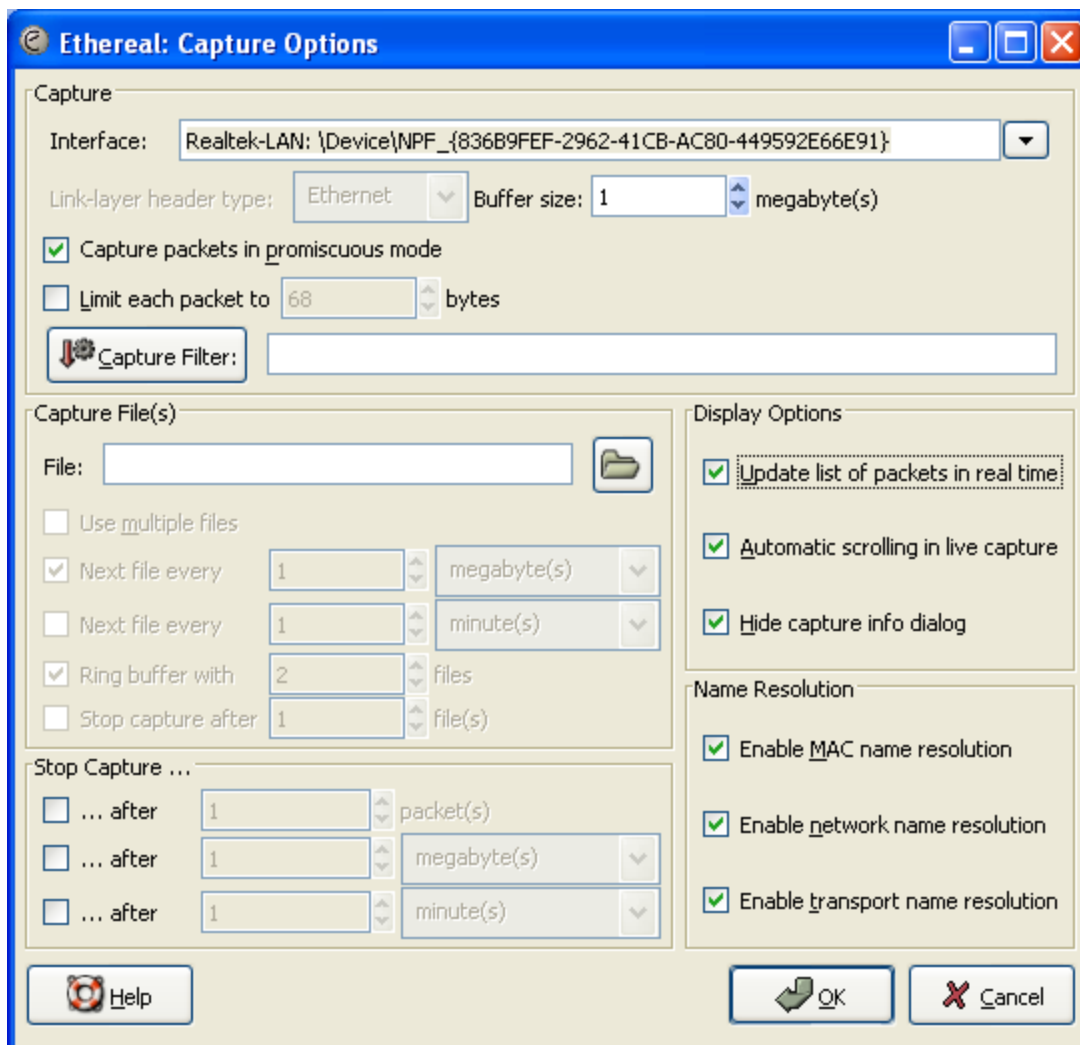


Figure 2

3.2.1 Capture Frame

3.2.1.1 Interface

This field specifies the interface you want to capture on. You can only capture on one interface, and you can only capture on interfaces that Ethereal has found (auto – detected) on the system. It is a drop-down list, so simply click on the button on the right hand side and select the interface you want. It defaults to the first non-loopback interface that supports capturing, and if there are none, the first loopback interface. On some systems, loopback interfaces cannot be used for capturing (loopback interfaces are not available on Windows platforms). This field performs the same function as the **-i <interface>** command line option.

3.2.1.2 Link-layer header type

Unless you are in the rare situation that you need this, just keep the default.

3.2.1.3 Buffer size: n megabyte(s)

Enter the buffer size to be used while capturing. This is the size of the kernel buffer which will keep the captured packets, until they are written to disk. If you encounter packet drops, try increasing this value.

3.2.1.4 Capture packets in promiscuous mode

This checkbox allows you to specify that Ethereal should put the interface in promiscuous mode when capturing. If you do not specify this, Ethereal will only capture the packets going to or from your computer (and not all packets on your LAN segment).

3.2.1.5 Limit each packet to n bytes

This field allows you to specify the maximum amount of data that will be captured for each packet, and is sometimes referred to as the **snaplen**. If disabled, the default is 65535, which will be sufficient for most protocols. Some rules of thumb:

- If you are unsure, just keep the default value.
- If you don't need all of the data in a packet - for example, if you only need the link-layer, IP, and TCP headers - you might want to choose a small snapshot length, as less CPU time is required for copying packets, less buffer space is required for packets, and thus perhaps fewer packets will be dropped if traffic is very heavy.
- If you don't capture all of the data in a packet, you might find that the packet data you want is in the part that's dropped, or that reassembly isn't possible as the data required for reassembly is missing.

3.2.1.6 Capture Filter

This option allows a tcpdump-style capture filter to be used. It defaults to empty, or no filter. You can also click on the button labelled Capture Filter, and Ethereal will bring up the Capture Filters dialog box and allow you to create and/or select a filter.

3.2.2 Capture File(s) frame

3.2.2.1 File

This field allows you to specify the file name that will be used for the capture file. The next two options, "Update list of packets in real time" and "Automatic scrolling in live capture" are not necessary if you are saving the capture to a file, but are very useful for watching the network on the fly. This field is left blank by default. If the field is left blank, the capture data will be stored in a temporary file. You can also click on the button to the right of this field to browse through the file system.

3.2.2.2 Use multiple files

Instead of using a single file, Ethereal will automatically switch to a new one, if a specific trigger condition is reached.

3.2.2.3 Next file every n megabyte(s)

Multiple files only: Switch to the next file after the given number of byte(s) / kilobyte(s) / megabyte(s) / gigabyte(s) have been captured.

3.2.2.4 Next file every n minute(s)

Multiple files only: Switch to the next file after the given number of second(s) / minutes(s) / hours(s) / days(s) have elapsed.

3.2.2.5 Ring buffer with n files

Multiple files only: Form a ring buffer of the capture files, with the given number of files.

3.2.2.6 Stop capture after n file(s)

Multiple files only: Stop capturing after switching to the next file the given number of times.

3.2.3 Stop Capture... frame

3.2.3.1 ... after n packet(s)

Stop capturing after the given number of packets have been captured.

3.2.3.2 ... after n megabytes(s)

Stop capturing after the given number of byte(s) / kilobyte(s) / megabyte(s) / gigabyte(s) have been captured. This option is greyed out, if "Use multiple files" is selected.

3.2.3.3 ... after n minute(s)

Stop capturing after the given number of second(s) / minutes(s) / hours(s) / days(s) have elapsed.

3.2.4. Display Options frame

3.2.4.1 Update list of packets in real time

This option allows you to specify that Ethereal should update the packet list pane in real time. If you do not specify this, Ethereal does not display any packets until you stop the capture. When you check this, Ethereal captures in a separate process and feeds the captures to the display process.

3.2.4.2 Automatic scrolling in live capture

This option allows you to specify that Ethereal should scroll the packet list pane as new packets come in, so you are always looking at the last packet. If you do not specify this, Ethereal simply adds new packets onto the end of the list, but does not scroll the packet list pane. This option is greyed out if "Update list of packets in real time" is disabled.

3.2.4.3 Hide capture info dialog

If this option is checked, the following capture info dialog (Figure 3) will be hidden. This option is greyed out if "Update list of packets in real time" is disabled.

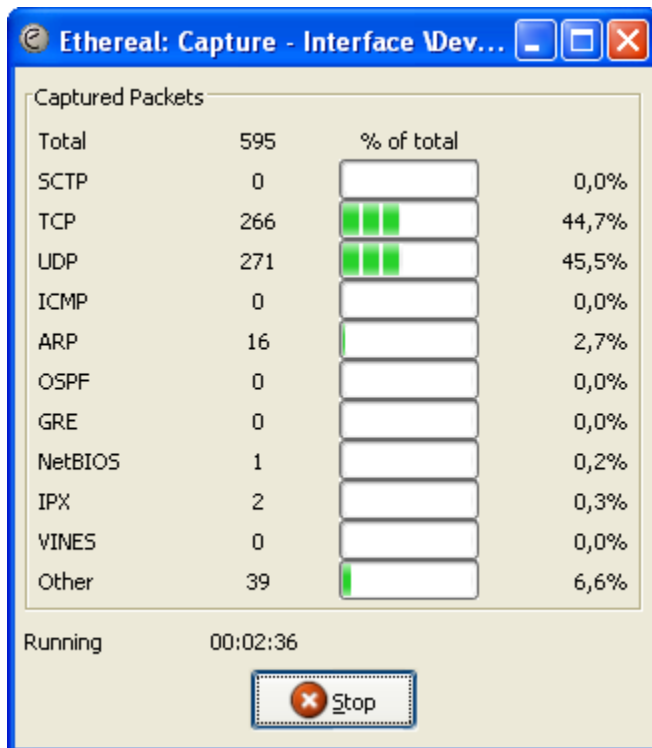


Figure 3

3.2.5 Name Resolution frame

The last three options all deal with name resolution: "Enable MAC name resolution", "Enable network name resolution", and "Enable transport name resolution". These three options can create additional traffic that grows with the amount of data collected, so users that do not want to disturb the network may want to turn these options off.

3.2.5.1 Enable MAC name resolution

This option allows you to control whether or not Ethereal translates MAC addresses into names.

3.2.5.2 Enable network name resolution

This option allows you to control whether or not Ethereal translates network addresses into names.

3.2.5.3 Enable transport name resolution

This option allows you to control whether or not Ethereal translates transport addresses into protocols.

3.3 Saving captured packets

You can save captured packets simply by using the Save As... menu item from the File menu under Ethereal. You can choose which packets to save and which file format to be used. The "Save Capture File As" dialog box allows you to save the current capture to a file with this dialog box, you can perform the following actions:

1. Type in the name of the file you wish to save the captured packets in, as a standard file name in your file system.
2. Select the directory to save the file into.
3. Select the range of the packets to be saved.
4. Specify the format of the saved capture file by clicking on the File type drop down box. You can choose from the types shown.
6. Use "Browse for other folders" to browse files and folders in your file system.
7. Click on the Save button to accept your selected file and save to it. If Ethereal has a problem saving the captured packets to the file you specified, it will display an error dialog box. After clicking OK on this error dialog box, you can try again.
8. Click on the Cancel button to go back to Ethereal and not save the captured packets.

3.4 Opening captured files

Ethereal can read in previously saved capture files. To read them, simply select the **Open** menu item from the **File** menu. Ethereal will then pop up the File Open dialog box (Figure 4).

With this dialog box, you can perform the following actions:

1. The "+ Add" button allows you to add a directory, selected in the right-hand pane, to the favorites list. Those changes are persistent.
2. The "- Remove" button allows you to remove a selected directory from that list again (the items like: "Home", "Desktop", and "Filesystem" cannot be removed).
3. Select files and directories with the list boxes.
4. View file preview information (like the filesize, the number of packets, ...), while browsing the filesystem.
5. Specify a display filter with the Filter button and filter field. This filter will be used when opening the new file. Clicking on the Filter button causes Ethereal to pop up the Filters dialog box.
6. Specify which name resolution is to be performed for all packets by clicking on one of the "Enable name resolution" check buttons.
7. Click the Open button to accept your selected file and open it. If Ethereal doesn't recognize the capture format, it will grey out this button.
8. Click the Cancel button to go back to Ethereal and not load a capture file.

Thus, a network packet analyzer is a measuring device used to examine what's going on inside a network cable. In the past, such tools were either very expensive, proprietary, or both. However, with the advent of Ethereal, all that has changed, making Ethereal perhaps one of the best open source packet analyzers available today.

Part 3: Questions

1. Provide a screen shot for the TCP Stream where it shows the username and password.
2. Under the menu bar, you will see the word 'Filter'. That's where you place filters to the packet captures. (You will see a filter that you have placed when you chose to follow the TCP stream). Click on 'clear' to see all traffic again. Now look for the DNS query response corresponding to

the query for mail.com. What is answer returned? Provide a screen shot that shows the DNS header with the returned answer.

3. Now you need to perform another packet capture. Go to 'mail.gmu.edu'. Then start your capture. Sign in to your GMU mail account, and then stop the capture. Go back and perform another 'Follow TCP stream' from under the 'Analyze' menu. Can you find your username or password? Why or why not?

4. Do some reading on SSL, and provide a good definition of SSL and a short explanation on how it works.

Practical: 6

Study of vulnerability scanning tool- Nessus

Aim: Configure and demonstrate use of vulnerability assessment tool such as NESSUS

Theory:

What is Nessus?

Nessus is a great tool designed to automate the testing and discovery of known security problems. Typically someone, a hacker group, a security company, or a researcher discovers a specific way to violate the security of a software product.

Nessus is the world's most popular open-source vulnerability scanner used in over 75,000 organizations worldwide. Many of the world's largest organizations are realizing significant cost savings by using Nessus to audit business-critical enterprise devices and applications.

The "Nessus" Project was started by Renaud Deraison in 1998 to provide to the Internet community a free, powerful, up-to-date and easy to use remote security scanner. Nessus is currently rated among the top products of its type throughout the security industry and is endorsed by professional information security organizations such as the SANS Institute. It is estimated that the Nessus scanner is used by 75,000 organizations worldwide.

One of the very powerful features of Nessus is its client server technology. Servers can be placed at various strategic points on a network allowing tests to be conducted from various points of view. A central client or multiple distributed clients can control all the servers. Nessus is designed to help identify and solve these known problems, before a hacker takes advantage of them.

Installation:

1. An installed version of UNIX is required.
2. Prior installation of several external programs is recommended:

[NMAP](#) is the industry standard for port scanners

[Hydra](#) is a weak password tester and

[Nikto](#) is a cgi/.script checker.

They are included because they are the best applications in their class. If installed in the PATH\$ before Nessus installation, they will automatically be available.

3. The simplest installation method is using the Lynx automatic install. Lynx is included on many of the Linux versions. The Lynx command is (logged in as a user, and not root)

```
lynx -source http://install.nessus.org | sh
```

Using Nessus:

This section shows how to configured Nessus server (nessusd) to scan a part of network (eight hosts actually). Seven of the tested hosts are on a local network, whereas the last is located somewhere else, across the Internet.

1.The configuration of the server by the administrator

Nessus is made up of two parts: a client and a server. You need a Unix-like system to use the server (Linux is just fine). In this test, I used the standard client nessus, mainly because I wrote it and because it is the only one that supports the cipher layer.

1.1 Download and install nessusd and nessus

You can download the latest version of Nessus from (www.nessus.org) See the installation instructions to find out how to compile it.

1.2 Create a nessusd account

The nessusd server has its own users database, each user having a set of restrictions. This allows you to share a single nessusd server for a whole network and different administrators who will only test their part of the network.

2. The use of the Unix client (including screenshots)

2.1The utility nessus-adduser takes care of the creation of a new account:

```
# nessus-adduser
```

Addition of a new nessusd user

Login : renaud

Authentication (pass/cert) [pass] : pass

Password: secret

User rules

Enter the rules for this user, and hit ctrl-D once you are done :

(the user can have an empty rules set)

deny 10.163.156.1

accept 10.163.156.0/24

default deny

Login : renaud

Password : secret

DN :

Rules :

deny 10.163.156.1

accept 10.163.156.0/24

default deny

Is that ok (y/n) ? [y] y

user added.

2.3 Configure your nessus daemon

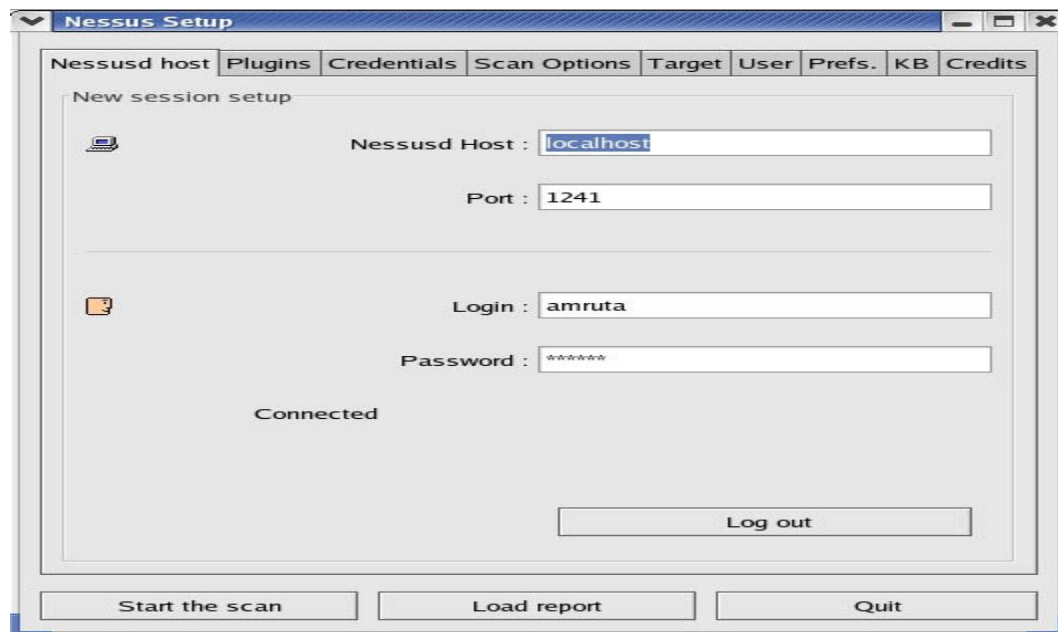
In the file /usr/local/etc/nessus/nessusd.conf, I can set several options for nessusd.

Start nessusd

Once all of this is done, I can safely start nessusd as root :

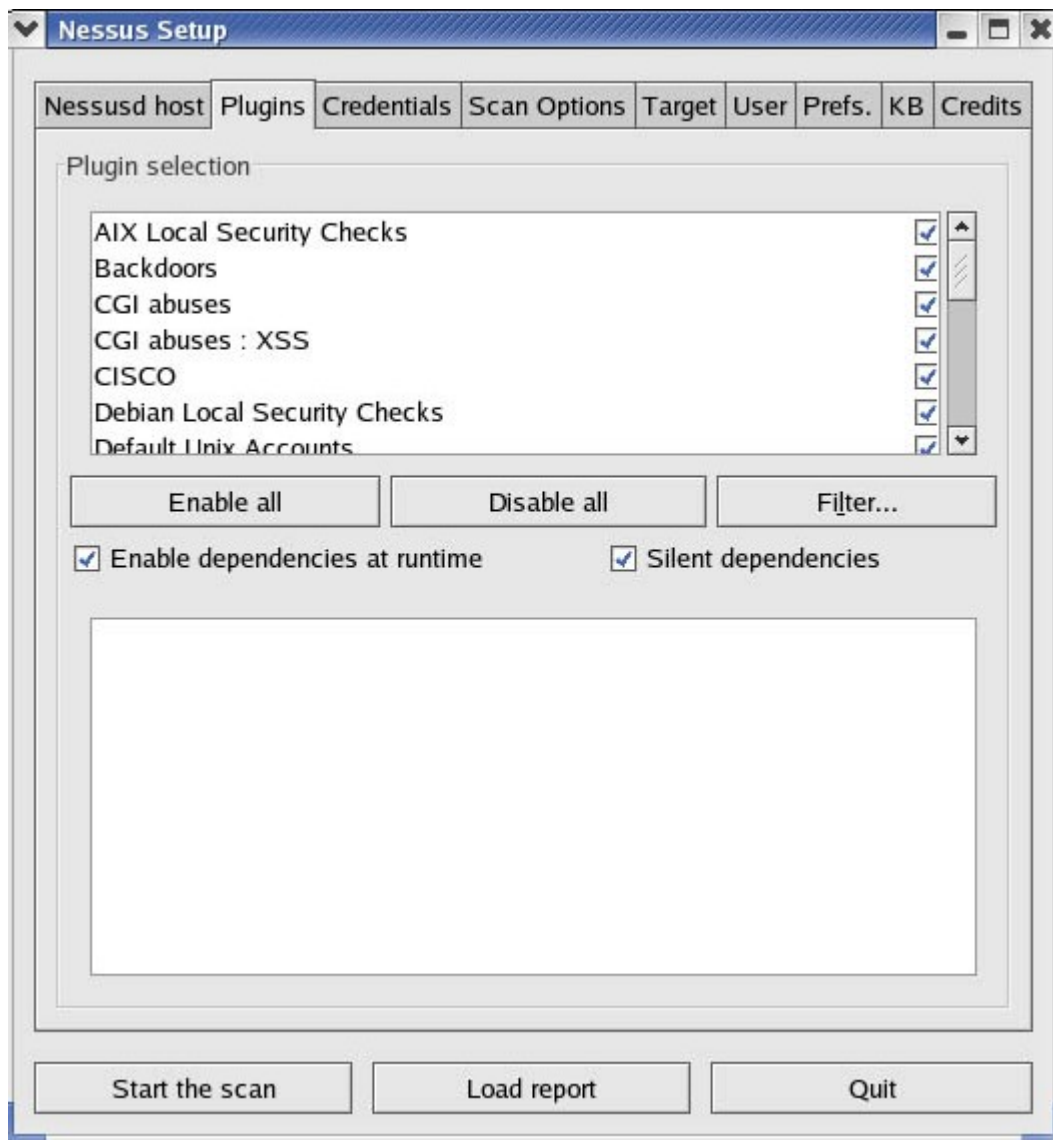
nessusd -D

2.4 The client configuration



Once I am connected, the **Log in** button changes to **Log out**, and a **Connected** label appears at its left.

2.5 The security checks configuration



Clicking on a plugin name will pop up a window explaining what the plugin does.

2.6 The plugins preferences

The image shows a screenshot of the 'Nessus Setup' window. The window has a title bar with a dropdown arrow, the text 'Nessus Setup', and standard window control buttons (minimize, maximize, close). Below the title bar is a tabbed interface with the following tabs: 'Nessusd host', 'Plugins', 'Credentials', 'Scan Options', 'Target', 'User', 'Prefs.', 'KB', and 'Credits'. The 'Credentials' tab is currently selected. Inside this tab, there is a sub-panel titled 'Credentials'. This panel contains several input fields and buttons: 'SSH user name :' with a text box containing 'root'; 'SSH password (unsafe!) :' with an empty text box; 'SSH public key to use :' with an empty text box and a 'Select...' button; 'SSH private key to use :' with an empty text box and a 'Select...' button; 'Passphrase for SSH key :' with an empty text box; 'SMB account :' with an empty text box; 'SMB password :' with an empty text box; and 'SMB domain (optional) :' with an empty text box. At the bottom of the window, there are three buttons: 'Start the scan', 'Load report', and 'Quit'.

You can give extra information to some security checks so that the audit is more complete. For instance, if you give a SMB login and account to nessusd, then you will be given local information about the remote Windows host (such as the missing security patches). Many options can be set through this panel.

2.7 The scan options

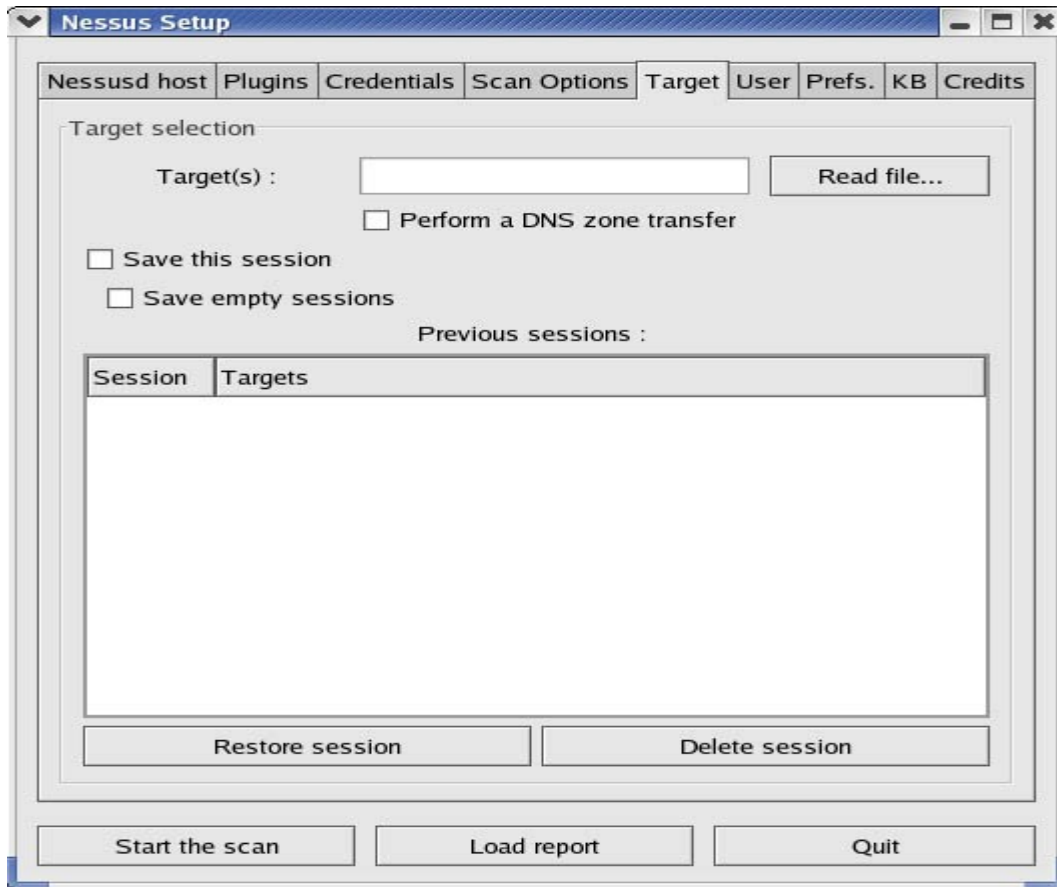
The screenshot shows the 'Nessus Setup' window with the 'Scan Options' tab selected. The window has a title bar with a minimize, maximize, and close button. Below the title bar is a tabbed interface with tabs for 'Nessusd host', 'Plugins', 'Credentials', 'Scan Options', 'Target', 'User', 'Prefs.', 'KB', and 'Credits'. The 'Scan Options' tab is active, showing a 'Scan options' section with various settings. The 'Port range' is set to 'default'. There are checkboxes for 'Consider unscanned ports as closed', 'Do a reverse lookup on the IP before testing it', 'Optimize the test' (checked), 'Safe checks' (checked), and 'Designate hosts by their MAC address'. The 'Number of hosts to test at the same time' is set to 20, and the 'Number of checks to perform at the same time' is set to 4. The 'Path to the CGIs' is set to '/cgi-bin:/scripts'. The 'Port scanner' section has a list of options: 'Nmap (NASL wrapper)', 'Exclude toplevel domain wildcard host', 'Ping the remote host', 'snmpwalk 'scanner'', and 'Netstat 'scanner'', each with a checkbox. The 'Start the scan', 'Load report', and 'Quit' buttons are at the bottom.

Tab	Value
Port range :	default
Consider unscanned ports as closed	<input type="checkbox"/>
Number of hosts to test at the same time :	20
Number of checks to perform at the same time :	4
Path to the CGIs :	/cgi-bin:/scripts
Do a reverse lookup on the IP before testing it	<input type="checkbox"/>
Optimize the test	<input checked="" type="checkbox"/>
Safe checks	<input checked="" type="checkbox"/>
Designate hosts by their MAC address	<input type="checkbox"/>
Port scanner :	
Nmap (NASL wrapper)	<input type="checkbox"/>
Exclude toplevel domain wildcard host	<input checked="" type="checkbox"/>
Ping the remote host	<input checked="" type="checkbox"/>
snmpwalk 'scanner'	<input type="checkbox"/>
Netstat 'scanner'	<input type="checkbox"/>

Start the scan Load report Quit

In this section, I choose which port scanner I want to use, how many hosts I want to have scanned at the same time, and how many plugins I want to run in parallel against each host. If I were to scan a firewalled web server, I could check the option "consider unscanned ports as closed" and only specify to scan port 80 - this would greatly speed up the scan.

2.8 Define the targets



The hosts of my local network are using private IP addresses, so entering '10.163.156.1-10.163.156.1.254' is fine. I do not check the 'Perform a DNS transfer zone' option, since it would make DNS transfer on *fr.nessus.org* and *nessus.org*, and it would be useless, since it would not gain any new hosts. I could use the following options to define my targets: 10.163.156.1

A single IP address.

10.163.156.1-254

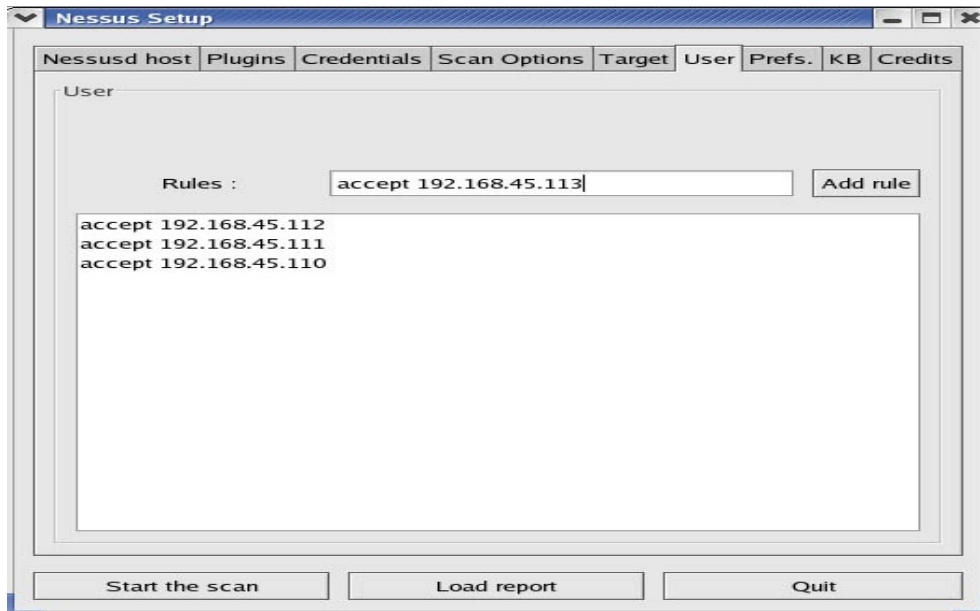
A range of IP addresses.

10.163.156.1-10.163.159.254

Another range of IP addresses.

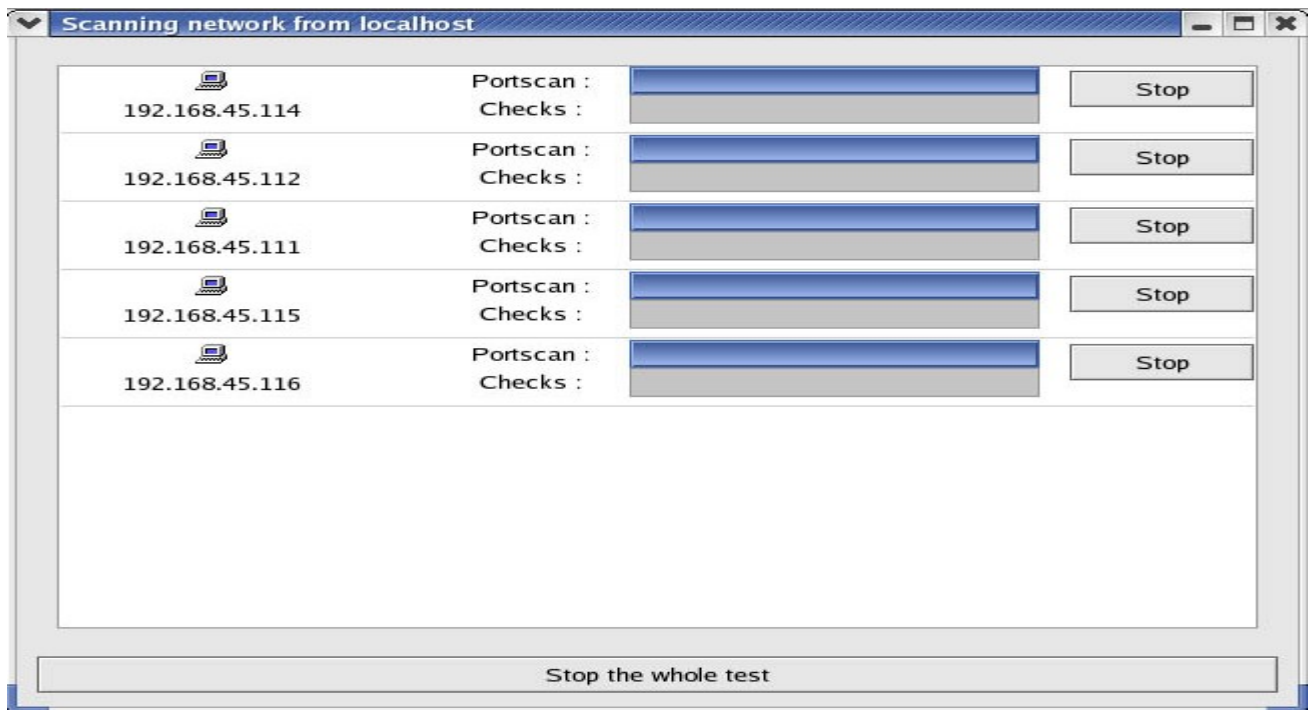
10.163.156.1/24

2.9 The rules section



The rules allow a user to restrict his test. For instance, I want to test 10.163.156.1/24, except 10.163.156.5. The rule set I Entered allows me to do that.

2.9 Start the test:



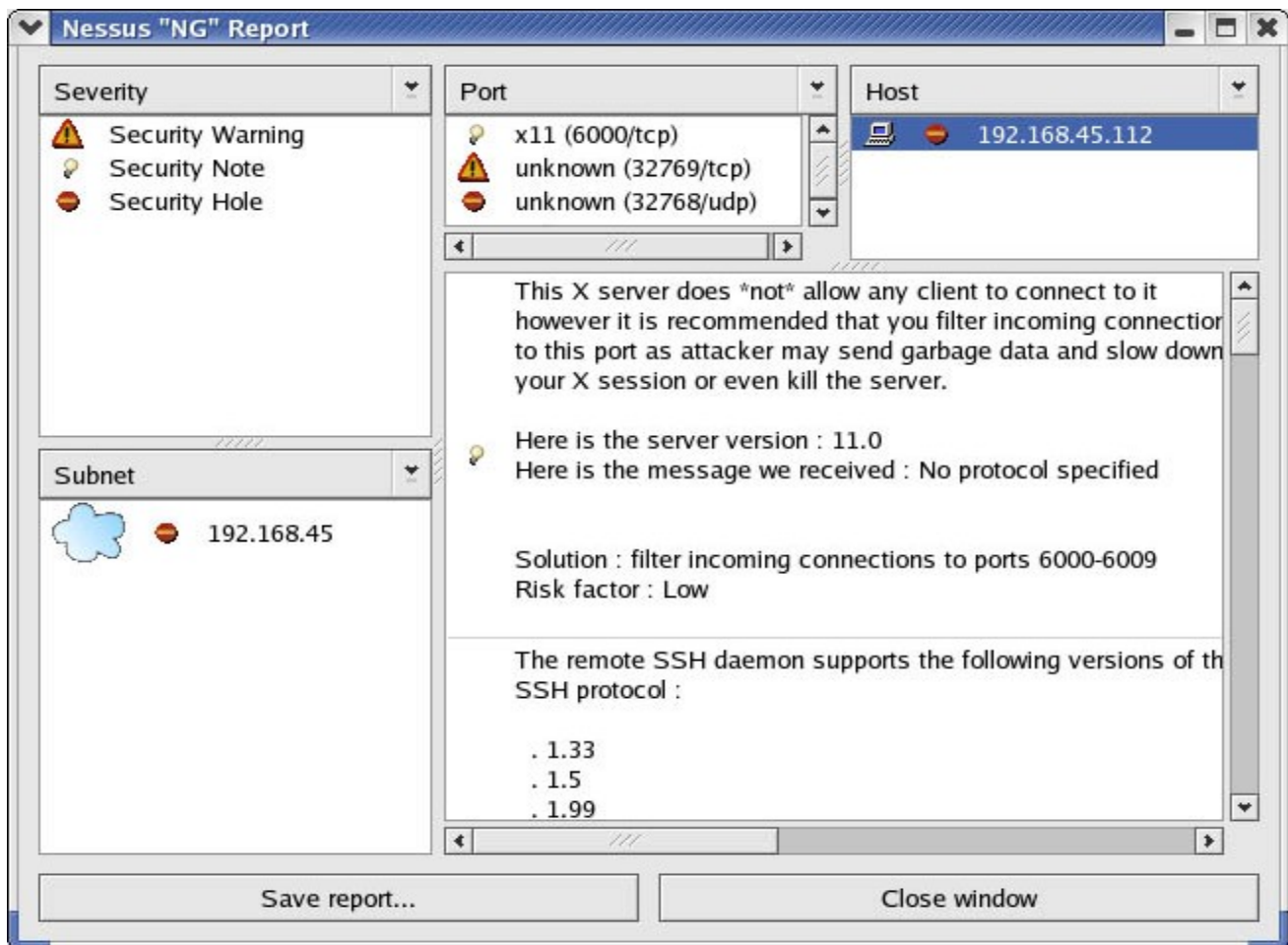


Fig. The results of the test.

Practical: 7

Digital Certificates

AIM: Implement web security with Open SSL tool kit

Theory:

1. Terminology

Digital Certificate: A Digital Certificate, or Digital ID, is the electronic counterpart to a driver's license, or passport. It can be presented electronically to prove your identity, or to validate your right to access private information or services online.

Certification Authority(CA): An entity that issues digital (X.509) certificates and vouches for the data contained in such certificates. A CA may be thought of as a trusted third party who "signs" certificates, making them valid. Eg. *Verisign, Thawte*

Terminology _ **CRL:** Certificate Revocation List

PEM: Privacy-Enhanced Mail format

DER: Distinguished encoding rules

X.509 Certificate: The standard format for digital certificates.

2. Setting up HTTPS server using Digital Certificate

To create a key for CA

```
/usr/bin/openssl genrsa -des3 1024 > ca.key
```

This will ask for a pass phrase.

To create certificate request

```
/usr/bin/openssl req -new -key ca.key -out ca.csr
```

This will ask for the pass phrase of the key. Enter the one you gave in the previous step.

Along with that all the information of the CA, like the country code, State, etc. need to be supplied.

To generate a self signed certificate

```
/usr/bin/openssl req -new -key ca.key -x509 -days 365 -out ca.crt
```

This will generate a self-signed CA certificate.

Thus now we have the self-signed CA certificate which can be used to sign other certificates.

To generate ca certificate in pem format

```
/usr/bin/openssl req -newkey rsa:1024 -keyout $PEM1 -nodes -x509 -days  
365 -out $PEM2 ; \  
cat $PEM1 > ca.pem ; \  
echo "" >> ca.pem ; \  
cat $PEM2 >> ca.pem ; \  

```

This certificate is required to sign the user certificate.

The CA maintains a text database of the certificates issued. A pre-defined directory structure is expected for the signing process which is defined in */usr/share/ssl/openssl.cnf*

file. You can change the required directory structure.

So create the following directory structure:

```
demoCA
```

```
/-private
```

```
/-newcerts
```

```
mkdir demoCA
```

```
cd demoCA
```

```
mkdir private
```

```
mkdir newcerts
```

Also, create files which are required for the database

```
vi serial
```

(put "00" in the file).

```
touch index.txt ( create empty file).
```

Please go through the various parameter in this file. Some of the parameters are

"optional" or "match". So for example a parameter Organization Unit (OU) is match then the CA can sign a certificate with only the same OU entry for the request certificate. An "optional" parameter has no restriction on the field.

Similarly create a key and csr for the server

(we will be using arian as our server name. Please replace the same with some other name or your server name).

```
/usr/bin/openssl genrsa -des3 1024 > arian.key
```

```
/usr/bin/openssl req -new -key arian.key -out arian.csr
```

This will ask all the information. In the common name field, give the server url or the IP address. This csr(certificate request) now needs to be signed by the CA. Hence we submit it to our CA for signature. The public key of the server and all the other information is provided with the csr. This is signed by the CA.

Sign the certificate

```
openssl ca -infiles arian.csr > arian.crt
```

The CA signs the certificate with his private key. Thus the certificate contains the public key and the general information of the server signed by the private key of CA. In no case the public key of the server is made available to any one, even the CA. The arian.crt file is our server certificate. We need to install this key on the web server. The server key, csr, and crt files are copied into the appropriate directories.

Copy the files files in appropriate directory of apache

(You need to be root(administrator) for this).

```
cp arian.key /etc/httpd/conf/ssl.key/server.key
```

```
cp arian.crt /etc/httpd/conf/ssl.crt/server.crt
```

```
cp arian.csr /etc/httpd/conf/ssl.csr/server.csr
```

Restart the apache server.

```
httpd -k stop
```

```
httpd -k start
```

You will be asked the passphrase for the server key.

(You need to be root(administrator) for this).

The **make** file provided also does the same thing. The steps required are

make ca.key --> Gen. CA key

make ca.csr --> Gen. CA csr

make ca.crt --> Gen. self signed CA certificate

make dirstruct --> Create the directory structure and files required.

make aryan.key --> Gen server key.

make aryan.csr --> Gen server csr.

make sign --> Sign the server certificate by CA. .

make install --> Copy the server keys at proper locations

make restart --> Restart the apache web server.

(Needs root perms for last two operations).

3. Testing in Browser-Mozilla

Open your browser(we will use mozilla here)

Accesss the site:

We will be using *10.12.14.10* for the web server address. Replace the same with your server url or .Common Name. used in to create the certificate.

Type in the url

http://10.12.14.10/

You can access this normally. Now try to access the same site with https protocol.

https://10.12.14.10/

And this time you are asked about the site being untrusted and some reasons give out. Try to analyze the results.

Accept the certificate only for the session. Go to some other site and then come back again. you will not be asked any thing.

Now close the browser and then again visit the same url, you are again asked for the certificate verification. This time accept the certificate permanently. Browse the site. Then close the browser.

Start the browser again. This time browser is not asking for any verification. This is because we have accepted the certificate permanently. So where is it stored? To find out go to Edit -> Preferences -> Privacy and Security -> Certificates -> Manage Certificates. Open the "web sites" tab. You will find one entry about the certificate that we accepted permanently. View the certificate. It is the same certificate that we saw before accepting.

Delete the certificate from the "web sites" tab. close the browser and open again to browse our site. As we have deleted the site, this time we are again asked about the site as in the first case.