414458: COMPUTER LABORATORY – VII

PART A : Information and Cyber Security

B.E. (Information Technology) 2015 Course
Semester I

Teaching Scheme		**Examination Scheme**	
Practical	**: 4 Hrs/Week**	**Term Work**	**: 50 Marks**
Theory	**: 3 Hr/Week**	**Practcal**	**: 50 Marks**

DEPARTMENT OF INFORMATION
TECHNOLOGY
2020-2021

INDEX

SCHEDULE

Sr. No.	Title	No. of Hrs.	Week
1	To implement RSA algorithm for key generation and cipher verification	2	1
2	To develop a program based on number theory such as Chinese remainder or Extended Euclidian algorithm.	2	2
3	Write program in C++ or Java to implement Diffie Hellman key exchange algorithm	2	3
4	To implement RSA algorithm using Libraries (API).	4	4 ,5
5	To implement SHA-1 algorithm using Libraries (API).	2	6
6	Security Tools (Minimum one)	2	11
	A. To configure and demonstrate use of vulnerability assessment tool such as NESSUS	2	12
	B. To implement web security with Open SSL tool kit.		

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

OBJECTIVE :

To study

- Concept of Public key and Private Key.
- Public key algorithm
- Working of RSA algorithm

THEORY:

Asymmetric/Public Key Algorithm:

Public key algorithms were evolved to solve the problem of key distribution in symmetric algorithms. This is achieved by using one key for encryption and a different but related key for decryption. These algorithms are designed such that it is computationally infeasible to determine the decryption key given only knowledge of the cryptographic algorithm and the encryption key. Also in some algorithms, such as RSA, either of the two related keys can be used for encryption, with the other used for decryption.

A public key encryption scheme has six ingredients:

- **Plaintext:** This is readable message or data that is fed into the algorithm as input.
- **Encryption algorithm:** The algorithm performs various transformations on the plaintext.
- **Public and private key:** This is a pair of keys that have been selected so that if one is used for encryption, the other is used for decryption.
- **Ciphertext:** This is the scrambled message produced as output. It depends on the plaintext and the key. For a given message, two different keys will produce two different ciphertexts.
- **Decryption algorithm:** This algorithm accepts the ciphertext and the matching key and produces the original plaintext.

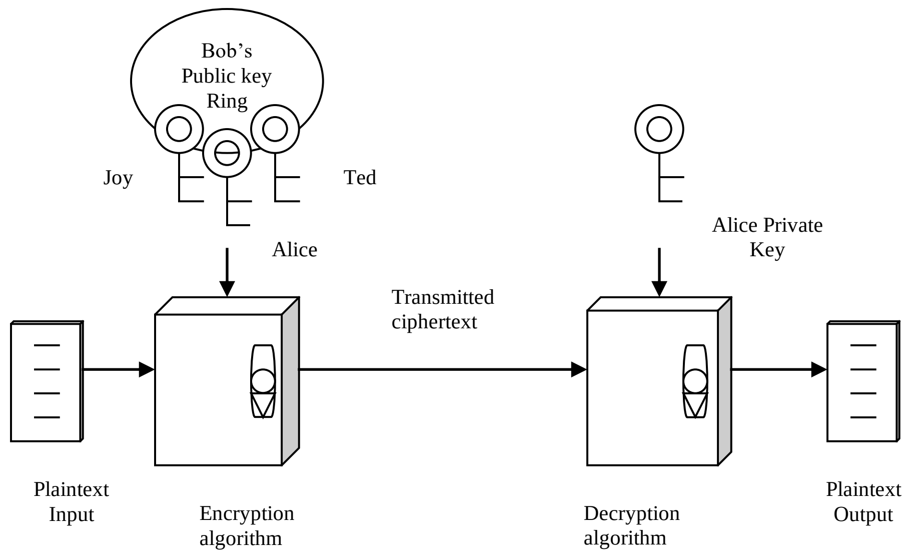


Fig. 1.1: Public key cryptography

The essential steps in public key algorithm are as follows:

1. Each user generates a pair of keys to be used for the encryption and decryption of messages.
2. Each user places one of the two keys in a public register or the other accessible file. This is the public key. The companion key is kept private. Each user maintains a collection of public keys obtained from other parties participating in communication.
3. If A wishes to send a confidential message to B, A encrypts the message using B's public key.
4. When B receives the message, B decrypts it using B's private key. No other recipient can decrypt the message because only B knows B's private key.

RSA Algorithm

RSA (which stands for Rivest, Shamir and Adleman who first publicly described it), an algorithm for public-key cryptographyinvolves three steps key generation, encryption and decryption.

RSA is a block cipher with each block having a binary value less than some number n. That is the block size must be less than or equal to log2 (n). Encryption

&decryption are of the following form, for some plaintext block M and ciphertext block C:

$C = M^{e} \bmod n$

$M = C^{d} \bmod n = (M^{e})^{d} \bmod n = M^{ed} \bmod n$

Both sender and receiver must know the value of n. The sender knows the value of e, and only the receiver knows the value of d. Thus, this is a public-key encryption algorithm with a public key of PU = {e, n} and a private key of PR = {d, n}. For this algorithm to be satisfactory for public key encryption, the following requirements must meet:

1. It is possible to find values of e, d, n such that $M^{ed} = M \bmod n$ for all $M<n$.
2. It is relatively easy to calculate M^{e} and C^{d} for all values of $M<n$.
3. It is infeasible to determine d given e and n.

Algorithm

1. Key generation

The keys (public key and private key) for the RSA algorithm are generated as:

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 $\varphi(n) = (p-1)(q-1)$, where φ is Euler's totient function
4. Choose an integer e such that $1 < e < \varphi(n)$ and $\gcd(e, \varphi(n)) = 1$, i.e. e and $\varphi(n)$ are co prime.
   - 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.

5. Determine d = e–1 mod φ(n); i.e. d is the multiplicative inverse of e mod φ(n).
    - This is more clearly stated as solve for d given (d*e)mod φ(n) = 1
    - This is often computed using the extended Euclidean algorithm.
    - d is kept as the private key exponent.

   Public key : PU = {e, n}
   Private Key : PR = {d, n}

2. **Encryption**

Alice transmits her public key (e, n) to Bob and keeps the private key secret. Bob then wishes to send message M to Alice.He computes the ciphertext c corresponding to

$$C = M^e \bmod n$$

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

3. **Decryption**

Alice can recover M from C by using her private key exponent d via computing

$$M = C^d \bmod n$$

Example

1. Select two prime numbers, p = 17 and q = 11.
2. Calculate n = pq = 17*11 = 187.
3. Calculate Ø(n) = (p-1)(q-1) = 16*10 = 160.
4. Select e such that relatively prime to Ø(n) = 160 and less than Ø(n); we choose e = 7.
5. Determine d such that de ≡ 1 (mod 160) and d < 160. The correct value is d = 23, because 23*7 = 161 = 10*160+1; d can be calculated using the extended Euclid's algorithm.

The resulting keys are public key PU = {7, 187} and private key PR = {23, 187}. The example shows the use of these keys for plaintext input of M=88.

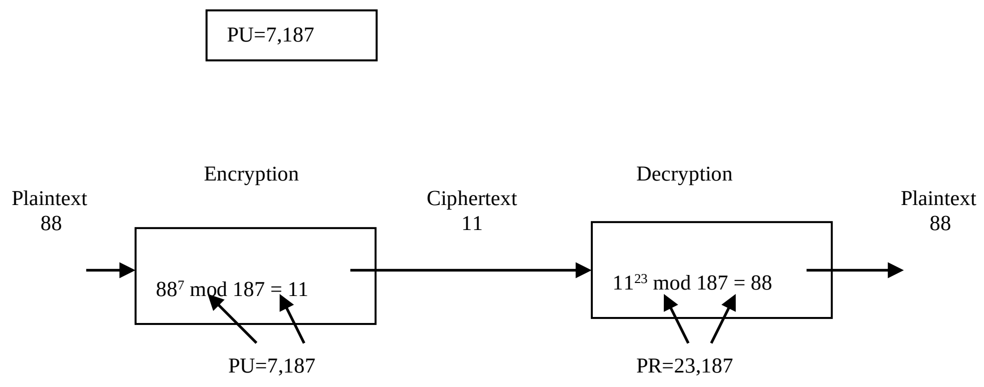


Fig. 1.2 : Example of RSA.

INPUT:

- Two prime numbers p = 17 and q = 11.
- Select e = 7.
- Plaintext = 88.

OUTPUT:

- PU = 7, 187.
- PR = 23, 187.
- Ciphertext = 11.

FAQs:

1. What are symmetric and asymmetric ciphers?
2. Why strong Primes are necessary in RSA?
3. Alice wants to generate a pair of RSA public and private keys. She starts by selecting two primes p = 5 and q = 7. Compute n, φ(n).
4. For long messages, RSA will be applied in blocks. If the block is very small, say it contains only one letter in each block, will the encryption be secure?
5. What are the possible attacks on RSA?

PRACTISE ASSIGNMENTS:

1. Implement RSA for text input.
2. Implement RSA for client server system.

AIM **:**Develop an program in C++ or Java based on number theory such as Chinese Remainder or Extended Euclidian algorithm.

OBJECTIVE :

To study

- Chinese Remainder Theorem
- Set of residues
- Relatively prime numbers
- What is modulo multiplicative inverse of a number.

THEORY:

Relative Prime Numbers

Two integers are termed relative prime if the only common factor between them is 1.

i.e. Greatest Common Divisor(m, n) = 1

Any integer can be broken down into certain multiples of prime numbers. This is called prime factorization. When two integers are prime factorized and the only common number is 1, then the two integers are relative prime.

Two distinct primes and are always relatively prime.

Relative primality is not transitive.

Example

18 = 2 x 3 x 3

35 = 7 x 5

so 18 and 35 are relative primes.

18 = 2 x 3 x 3

21 = 3 x 7

3 is common, so 18 and 21 are not relative prime.

Set of Residues

It is a set of nonnegative integers less than n.

Zn = {0,1,2,…….(n – 1) }

Chinese Remainder Theorem (CRT)

Let m1, m2, …, mk be pair wise relatively prime positive integers. That is, gcd(mi, mj) = 1 for 1£i<j £k.

Let aiÎZmi for 1£i £k and set M=m1m2…mk.

Then there exists a unique A ÎZm such that Aºai mod mi for i = 1…k.

A can be computed as:

[illegible]

where [illegible] for 1 ≤ i ≤ k.

Steps in Chinese Remainder Theorem

1. Find M = m1 × m2 × … × mk. This is the common modulus.
2. Find M1 = M/m1, M2 = M/m2… Mk = M/mk.
3. Find the multiplicative inverse of M1, M2, …, Mk using the corresponding moduli (m1, m2, …, mk). Call the inverses $M1^{-1}$, $M2^{-1}$, …, Mk^{-1}.
4. The solution to the simultaneous equations is

$$x = (a_1 \times M_1 \times M_1^{-1} + a_2 \times M_2 \times M_2^{-1} + \cdots + a_k \times M_k \times M_k^{-1}) \bmod M$$

Example1

Represent 973 in Z1813 as a k-tuple.

Answer:

- M = 1813 = 37 * 49 è m1 = 37 & m2 = 49
- A = 973
- A = (A mod m1, A mod m2) = (11, 42)

Example2

Find x for the following equations:

x ≡2 mod 3

$x \equiv 3 \bmod 5$

$x \equiv 2 \bmod 7$

Answer

1. $M = 3 \times 5 \times 7 = 105$
2. $M1 = 105 / 3 = 35$, $M2 = 105 / 5 = 21$, $M3 = 105 / 7 = 15$
3. The inverses are $M1-1 = 2$, $M2-1 = 1$, $M3 -1 = 1$
4. $x = (2 \times 35 \times 2 + 3 \times 21 \times 1 + 2 \times 15 \times 1) \bmod 105 = 23 \bmod 105$
5. $x = 23$

INPUT : Values of a_i and m_i

OUTPUT : Unique value of X

FAQs :

1. State Chinese Remainder Theorem.
2. List some applications of CRT.
3. What is set of residues?
4. What is multiplicative inverse?
5. How to find unique integer which is represented by the k tuple.
6. What is meant by relatively prime numbers?

PRACTISE ASSIGNMENTS

Write a program to illustrate Extended Euclid's Algorithm.

Assignment 3 Implement Diffie-Hellman Key Exchange algorithm

AIM **:** Write program in C++ or Java to implement Diffie Hellman key exchange algorithm.

OBJECTIVE :

To study

- Benefits of Symmetric key algorithm
- Discrete log problem

THEORY:

Motivation for Key Exchange Algorithm

The fastest encryption schemes are symmetric encryption schemes. In order to use these, the two communicators (Alice & Bob) would have to meet in a secure location. BUT, this sort of defeats the purpose, because their whole goal is to communicate when securely when they aren't in the same place. A modern day example of why it would be useful to exchange a key without meeting deals with an online purchase. You are making the purchase online so that you DON'T have to go to the store to meet with the vendor. So, you don't really want to go to the store to just exchange a secret key either. You want to be able to do that from the comfort of your own home.

Thus, that is the underlying problem: how do two users exchange a secret key without a secure communication channel?

The first solution to this problem was the Diffie-Hellman Key Exchange. What's interesting about this algorithm is that neither user actually gets to choose the key. But, at the end of the algorithm, both users have calculated the same key, which is not easy for an eavesdropper to calculate.

In order to understand why the Diffie-Hellman Key Exchange is difficult, it is important to understand the Discrete Log Problem.

Discrete Log Problem

The (discrete) exponentiation problem is as follows: Given a base a, an exponent b and a modulus p, calculate c such that $a^b \equiv c \pmod{p}$ and $0 \le c < p$.

It turns out that this problem is fairly easy and can be calculated "quickly" using fast-exponentiation.

The discrete log problem is the inverse problem:

Given a base a, a result c ($0 \le c < p$) and a modulus p, calculate the exponent b such that

$a^b \equiv c \pmod{p}$.

It turns out that no one has found a quick way to solve this problem. To get an intuition as to why this is the case, try picking different values of a and p, and listing out each successive power of a mod p. What you will find is that there is no discernable pattern for the list of numbers created. Thus, given a number on the list, it's very difficult to predict where it appears on the list.

Here's a concrete example:

Given $a = 2$, $b = 7$ and $p = 37$, calculate c: $2^7 = 128$, and $128 \equiv 17 \pmod{37}$

Given $a = 2$, $c = 17$, and $p = 37$, calculate b: Try each value of b until you find one that works! (For large prime numbers, this is too slow.)

IMPLEMENTATION METHOD :

Diffie Hellman Key Exchange

Alice and Bob agree on two values: a large prime number p, and a generator g, $1 < g < p$. (It's better if g is an actual generator, meaning that when you raise it to the 1st, 2nd, 3rd, …, p-1 powers, you get all different answers.)

These values are known to everyone.

In secret, Alice picks a value a, with $1 < a < p$.
In secret, Bob picks a value b, with $1 < b < p$.

Alice calculates $g^a \pmod{p}$, call this f(a) and sends it to Bob.
Bob calculates $g^b \pmod{p}$, call this f(b) and sends it to Alice.

Note that f(a) and f(b) are also known by everyone.

In secret, Alice computes $f(b)^a \pmod{p}$ – this is the exchanged key.
In secret, Bob computes $f(a)^b \pmod{p}$ – this is again, the exchanged key.

Why does this work?

$f(b)^a \equiv (g^b)^a \equiv g^{ab} \pmod{p}$. Similarly,
$f(a)^b \equiv (g^a)^b \equiv g^{ab} \pmod{p}$.

Here's a concrete example:

Let $p = 37$ and $g = 13$.

Let Alice pick a = 10. Alice calculates 13^{10} (mod 37) which is 4 and sends that to Bob.

Let Bob pick b = 7. Bob calculates 13^{7} (mod 27) which is 32 and sends that to Alice.

(Note: 6 and 7 are secret to Alice and Bob, respectively, but both 4 and 32 are known by all.)

Alice receives 32 and calculates 32^{10} (mod 37) which is 30, the secret key.
Bob receives 4 and calculates 4^{7} (mod 37) which is 30, the same secret key.

Note that neither Alice nor Bob chose 30, but that they ended up with that secret key anyway. Furthermore, note that even with knowing p = 37, g = 13, f(a) = 4 and f(b) = 32, it is difficult to ascertain the secret key, 30 without doing a brute force check.

In particular, if the discrete log problem were easy, this scheme could be broken. Consider the following:

If an adversary saw that f(a) = 4, p = 37 and g = 13 and could solve the discrete log problem, then they could calculate that $13^{10} \equiv 4$ (mod 37). Once they had this value, 10, then they could take f(b) = 32 and then calculate 32^{10} (mod 37) to arrive at 30.

Thus, the Diffie-Hellman Key Exchange is only as secure as the Discrete Log problem.

INPUT : Public Values – g and p
Private Values – for Alice a and Bob b

OUTPUT **:** Shared Secrete Key ($K = g^{ab} \bmod p$)

FAQs **:**

1. How Does the Diffie-Hellman Key Exchange Work?
2. How to Import a Diffie-Hellman Public Key and Calculate the Secret Session Key?
3. What are the Vulnerabilities in Diffie -Hellman?
4.

PRACTISE ASSIGNMENTS

Study of AES algorithm.

AIM : Write a program in C++, C# or Java to implement RSA algorithm using Libraries (API).

OBJECTIVE :

To study

- Working of RSA algorithm
- Use of Java APIs

THEORY:

Algorithm

Key Generation	
Select p, q	p and q both prime, $p \neq q$
Calculate n = p * q	
Calculate Ø(n) = (p-1)(q-1)	
Select integer e	gcd(Ø(n),e) = 1; 1<e<Ø(n)
Calculate d	d = e(-1) mod Ø(n)
Public key	PU = {e, n}
Private key	PR = {d, n}

Encryption	
Plaintext	M<n
Ciphertext	$C=M^e \bmod n$

Decryption	
Ciphertext	C
Plaintext	$M = C^d \bmod n$

Required Classes

- **KeyGeneration**

1. **Class KeyPairGenerator**

The KeyPairGenerator class is used to generate pairs of public and private keys. A Key pair generator for a particular algorithm creates a public/private key pair that can be used with this algorithm. It also associates algorithm-specific parameters with each of the generated keys.

Constructor Summary	
protected	**KeyPairGenerator**(String algorithm) Creates a KeyPairGenerator object for the specified algorithm.
Required Methods	
KeyPair	**generateKeyPair**() Generates a key pair.
static KeyPairGenerator	**getInstance**(String algorithm) Generates a KeyPairGenerator object that implements the specified digest algorithm.
void	**initialize**(AlgorithmParameterSpec params, SecureRandom random) Initializes the key pair generator with the given parameter set and source of randomness.

2. **Class SecureRandom**

This class provides a cryptographically strong pseudo-random number generator.

Constructor Summary
SecureRandom() By using this constructor, the caller obtains a SecureRandom object containing the implementation from the highest-priority installed provider that has a SecureRandom implementation.
SecureRandom(byte[] seed) By using this constructor, the caller obtains a SecureRandom object containing

the implementation from the highest-priority installed provider that has a SecureRandom implementation.

3. **Class KeyPair**

This class is a simple holder for a key pair (a public key and a private key). It does not enforce any security, and, when initialized, should be treated like a PrivateKey.

Constructor Summary	
KeyPair(PublicKey publicKey, PrivateKey privateKey) Constructs a key pair from the given public key and private key.	
Required Methods	
PrivateKey	**getPrivate**() Returns a reference to the private key component of this key pair.
PublicKey	**getPublic**() Returns a reference to the public key component of this key pair.

4. **Class ObjectOutputStream**

An ObjectOutputStream writes primitive data types and graphs of Java objects to an OutputStream. The objects can be read (reconstituted) using an ObjectInputStream.

Constructor Summary	
protected	**ObjectOutputStream**() Provide a way for subclasses that are completely reimplementing ObjectOutputStream to not have to allocate private data just used by this implementation of ObjectOutputStream.
	ObjectOutputStream(OutputStream out) Creates an ObjectOutputStream that writes to the specified OutputStream.
Required Methods	
void	**close**() Closes the stream.
void	**writeObject**(Object obj)

	Write the specified object to the ObjectOutputStream.

b. Encryption and Decryption

1. Class KeyGenerator

This class provides the functionality of a (symmetric) key generator.Key generators are constructed using one of the getInstance class methods of this class.

Constructor Summary	
protected	**KeyGenerator**(KeyGeneratorSpi keyGenSpi, Provider provider, String algorithm) Creates a KeyGenerator object.
Required Methods	
SecretKey	**generateKey**() Generates a secret key.
static KeyGenerator	**getInstance**(String algorithm) Generates a KeyGenerator object for the specified algorithm.
void	**init**(SecureRandom random) Initializes this key generator.

2. Class ObjectInputStream

An ObjectInputStream deserializes primitive data and objects previously written using an ObjectOutputStream.

Constructor Summary	
protected	**ObjectInputStream**() Provide a way for subclasses that are completely reimplementing ObjectInputStream to not have to allocate private data just used by this implementation of ObjectInputStream.
	ObjectInputStream(InputStream in)

	Creates an ObjectInputStream that reads from the specified InputStream.
Required Methods	
void	**close**() Closes the input stream.
Object	**readObject**() Read an object from the ObjectInputStream.

3. Interface Key

The Key interface is the top-level interface for all keys. It defines the functionality shared by all key objects.

4. Class Cipher

This class provides the functionality of a cryptographic cipher for encryption and decryption.

Constructor Summary	
protected	**Cipher**(CipherSpi cipherSpi, Provider provider, String transformation) Creates a Cipher object.
Required Methods	
byte[]	**doFinal**() Finishes a multiple-part encryption or decryption operation, depending on how this cipher was initialized.
byte[]	**doFinal**(byte[] input, int inputOffset, int inputLen) Encrypts or decrypts data in a single-part operation, or finishes a multiple-part operation.
static Cipher	**getInstance**(String transformation) Generates a Cipher object that implements the specified transformation.
void	**init**(int opmode, Key key) Initializes this cipher with a key.
Key	**unwrap**(byte[] wrappedKey, String wrappedKeyAlgorithm, int wrappedKeyType)

	Unwrap a previously wrapped key.
byte[]	**wrap**(Key key) Wrap a key.
int	**update**(byte[] input, int inputOffset, int inputLen, byte[] output) Continues a multiple-part encryption or decryption operation (depending on how this cipher was initialized), processing another data part.

5. Class DataOutputStream

A data output stream lets an application write primitive Java data types to an output stream in a portable way.

Constructor Summary	
DataOutputStream(OutputStream out) Creates a new data output stream to write data to the specified underlying output stream.	
Required Methods	
void	**write**(int b) Writes the specified byte (the low eight bits of the argument b) to the underlying output stream.
void	**writeInt**(int v) Writes an int to the underlying output stream as four bytes, high byte first.

6. Class InputStream

This abstract class is the superclass of all classes representing an input stream of bytes.

Required Methods	
int	**read**(byte[] b) Reads some number of bytes from the input stream and stores them into the buffer array b.

7. Class OutputStream

This abstract class is the superclass of all classes representing an output stream of bytes.

Required Methods	
void	**close**() Closes this output stream and releases any system resources associated with this stream.
void	**write**(byte[] b) Writes b.length bytes from the specified byte array to this output stream.

Algorithm

Key Generation

1. Geta key pair generator object for generating keys for RSA algorithm by using KeyPairGenerator class.
2. Initialize the Key Pair Generator.
3. Generate the pair of keys by using KeyPair class.
4. Write public and private keys to two different files.

Encryption

1. Initialize cipher in encrypt mode.
2. Read data from plaintext file into byte array.
3. Call doFinal method to carry out encryption operation.
4. Write output array to ciphertext file.

Decryption

1. Initialize cipher in decrypt mode.
2. Read data from ciphertext file into byte array.

3. Call doFinal method to carry out encryption operation.
4. Write output array to decrypted file.

INPUT : File containing plaintext.

OUTPUT :After encryption – ciphertext file
After decryption – original plaintext

FAQs :

1. What is the use of KeyPairGenerator class?
2. What are the modes in which cipher can be initialized?

PRACTISE ASSIGNMENTS:

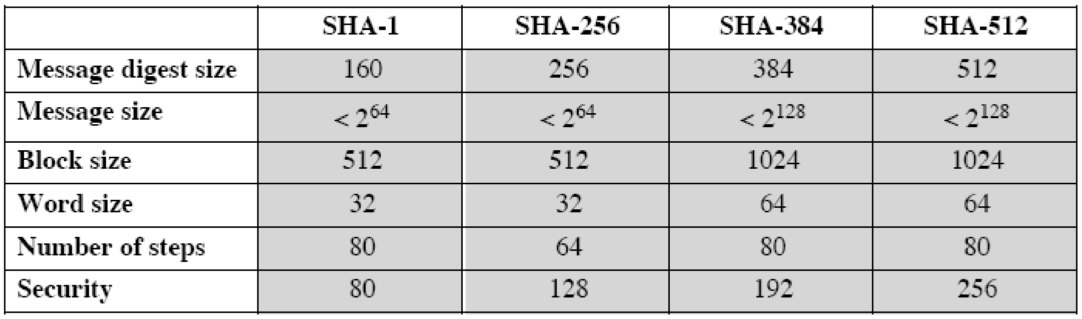
Implement RSA for client server system using libraries(API).

AIM **:**To study and implement SHA-1 (Secure Hash Algorithm)

OBJECTIVE: To implement and understand details of SHA-1 (Secure Hash Algorithm)

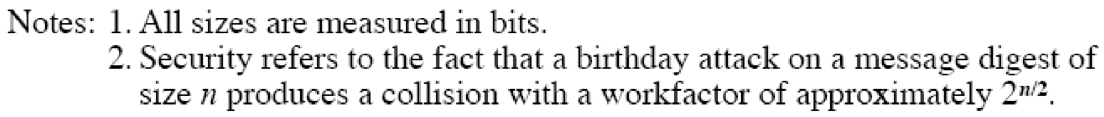
THEORY:

The National Institute of Standard and Technology(NIST) along with NSA developed the Secure Hash Algorithm(SHA).SHA works with any input message that is less than 2 ^{64}bits in length. The output of SHA-1 is a message digest which is 160bits in length. Following table shows SHA parameters.



	SHA-1	SHA-256	SHA-384	SHA-512
Message digest size	160	256	384	512
Message size	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Block size	512	512	1024	1024
Word size	32	32	64	64
Number of steps	80	64	80	80
Security	80	128	192	256

Fig. 5.1 Comparison of SHA parameters



Notes: 1. All sizes are measured in bits.
2. Security refers to the fact that a birthday attack on a message digest of size n produces a collision with a workfactor of approximately $2^{n/2}$.

The word secure is SHA was decided based on two features. SHA is designed to be computationally infeasible to:

1. Obtain the original message, given its message digest and
2. Find two messages producing the same message digest

Following are the important steps in execution of SHA

Step 1. Padding

The first step in SHA is to add padding to the end of the original message in such a way that the length of the message is 64 bits short of multiple of 512.Padding is always added even if message is already 64 bit short of multiple of 512.

Step 2. Append length

The length of the message excluding the length of the padding is now calculated and appended to the end of the padding as a 64 bit block.

Step 3. Divide the input into 512 bit block

The input message is now divided into blocks, each of length 512 bits. These blocks become the input to the message digest processing logic.

Step 4. Initialize chaining variables

Five chaining variables A through E are initializes. Each chaining variable is 32 bits in length. Following table shows their values

Variable Name	Value(in Hexadecimal)
A	01 23 45 67
B	89 AB CD EF
C	FE DC BA 98
D	76 54 32 10
E	C3 D2 E1 F0

Fig. 5.2 Values of chaining variable

Step 5. Process Block

This process is divided into following sub steps

Step 5.1

Copy the chaining variables –E into variables a-e. The combination of a-e called abcde will be considered as a single register for storing the temporary intermediate as well as the final results.

Step 5.2

Now divide the current 512 bit block into 16 sub blocks, each consisting of 32 bits.

Step 5.3

SHA has 4 rounds each consisting of 20 steps or iteration. Each round takes the current 512 bit block, the register abcde and a constant K[t] (where t=0 to 79) as the three inputs. It then updates the contents of the register abcde using SHA algorithm steps.We have only four constants defined for K[t],one used in each of four rounds .The values of K[t] are given in following table

Round	Value of t between	K[t](in Hexadecimal)
1	1 and 19	5A 92 79 99
2	20 and 39	6E D9 EB A1
3	40 and 59	9F 1B BC DC
4	60 and 79	CA 62 C1 D6

Fig. 5.3 Values of K[t]

Step 5.4

SHA consistof four rounds, each round containing 20 iterations. This makes a total of 80 iterations. The logical operation of SHA-1 is shown in following fig.

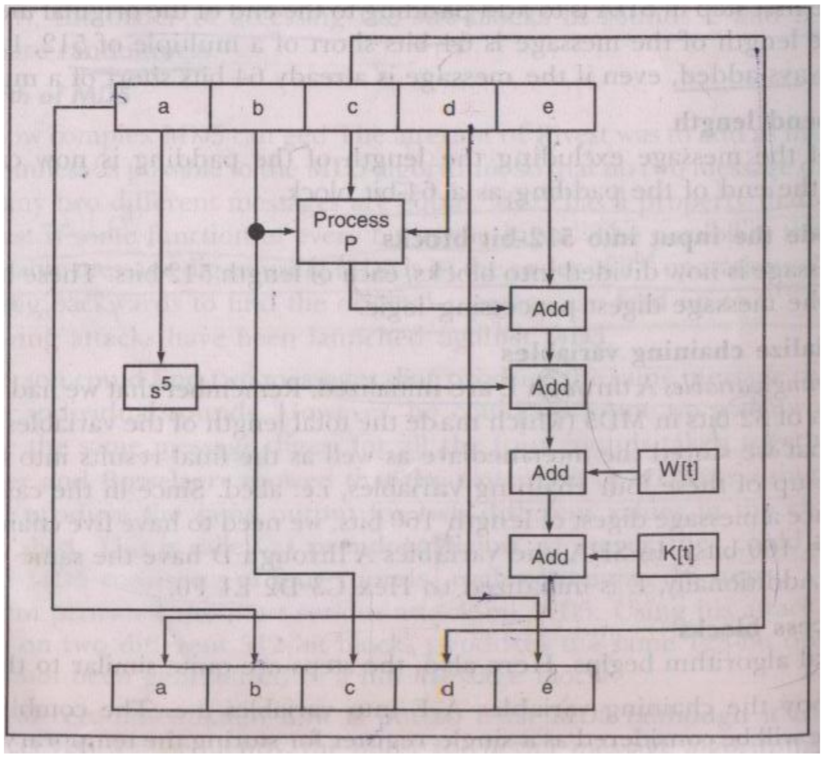


Fig. 5.4 Single SHA-1 iteration

Mathematically iteration consists of following operations

abcde =(e + Process P +s^5 (a) + W[t] + K[t]),a,s^{30} (b),c, d

where

abcde =The register made up of five variables a, b, c, d, e

process P=The logical operation ,which is given in following table

s^t =Circular left shift of 32 bit sub block by t bits.

W[t]=A 32 bit value derived from current 32 bit sub block

K[t]=one of the constant defined earlier

Process P is given as

Round	Process P
1	(b AND c) OR ((NOT b) AND (d))
2	b XOR c XOR d
3	(b AND c) OR (b AND d) OR (c AND d)
4	B XOR c XOR d

Fig 5.5 Process P in four rounds

Value of W[t] is calculated as follow

For the first 16 blocks of W(i.e t=0 to 15) the contents of the inpu message sub block M[t] become the content of W[t]. Remaining values are calculated as

For t=0 to 15	Value of W[t]
W[t]=	Same as M[t]
W[t]=	(W[t -16] **XOR** W[t-14] **XOR** W[t-8] **XOR** W[t-3])

Fig 5.6 Values of W

Required Classes

1. **Class MessageDigest (java.security.MessageDigest)**

This MessageDigest class provides applications the functionality of a message digest algorithm, such as MD5 or SHA. Message digests are secure one-way hash functions that take arbitrary-sized data and output a fixed-length hash value.

Constructor Summary

protected	**MessageDigest(**Stringalgorithm**)** Creates a message digest with the specified algorithm name.
Required Methods	
static MessageDigest	**getInstance**(String algorithm) Generates a MessageDigest object that implements the specified digest algorithm.
static MessageDigest	**getInstance**(String algorithm,Provider provider) Generates a MessageDigest object implementing the specified algorithm, as supplied from the specified provider, if such an algorithm is available from the provider.
static MessageDigest	**getInstance**(String algorithm, String provider) Generates a MessageDigest object implementing the specified algorithm, as supplied from the specified provider, if such an algorithm is available from the provider.
void	**update**(byte[] input) Updates the digest using the specified byte.
void	**update**(byte[] input) Updates the digest using the specified array of bytes.
void	**update**(byte[] input, int offset, int len) Updates the digest using the specified array of bytes, starting at the specified offset.

INPUT **:** Plaintext

OUTPUT **:**Message Digest

FAQs **:**

1. What is SHA?
2. Who has developed SHA?
3. What are the variants of SHA?
4. What is padding of bits ?Why it is necessary
5. Compare MD5 with SHA-1?
6. What are the registers used ?What is their purpose
7. SHA -1 uses how many rounds and iterations?
8. Where SHA is used?

PRACTISE ASSIGNMENTS

Design an application which will useSHA 1 for maintaining integrity.

Assignment 8 Study Of Protocol Analyzer Tool

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

OBJECTIVE: Study any protocol analyzer software and use its implementation features.

THEORY:

Introduction to Wireshark

Wireshark is a GUI network protocol analyzer. It lets user interactively browse packet data from a live network or from a previously saved capture file. Its open source license allows experts in the network community to add enhancement. It runs on all popular platforms.

Wireshark native capture file format is libpcap format, which is also the format used by tcpdump and various other tools. It can read many or all the file formats including compressed and uncompressed files and folders.

Platforms on which Wireshark runs

Wireshark runs on most UNIX and various windows. It requires GTK+, GTK6, libpcap and other libraries in order to run.

- **UNIX**
  - Applet, MAC, BEOS, JBM, AIX, BSD open etc.
- **LINUX**
  - Mandrake LINUX, Red Hat, SUSE LINUX etc.
- **WINDOWS**
  - Windows server 2003/XP/2000/NT

Building and Installing

Wireshark can be downloaded from the main Web site:

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

or from any of their mirrors worldwide. Installing Wireshark 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 Wireshark. You may acquire a binary installable of Wireshark at
   http://www.ethereal.com/download.html#binaries
   Download the installer and execute it.

WinCap

WinCap is a library file necessary function running Wireshark. If you don't have WinCap installed you will not be able to open saved captured files. During installing Wireshark it will ask to install the WinCap for proper functioning of Wireshark.

Uses of Wireshark

Here is some examples people use Wireshark 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

Using Wireshark

1. **Starting it up**

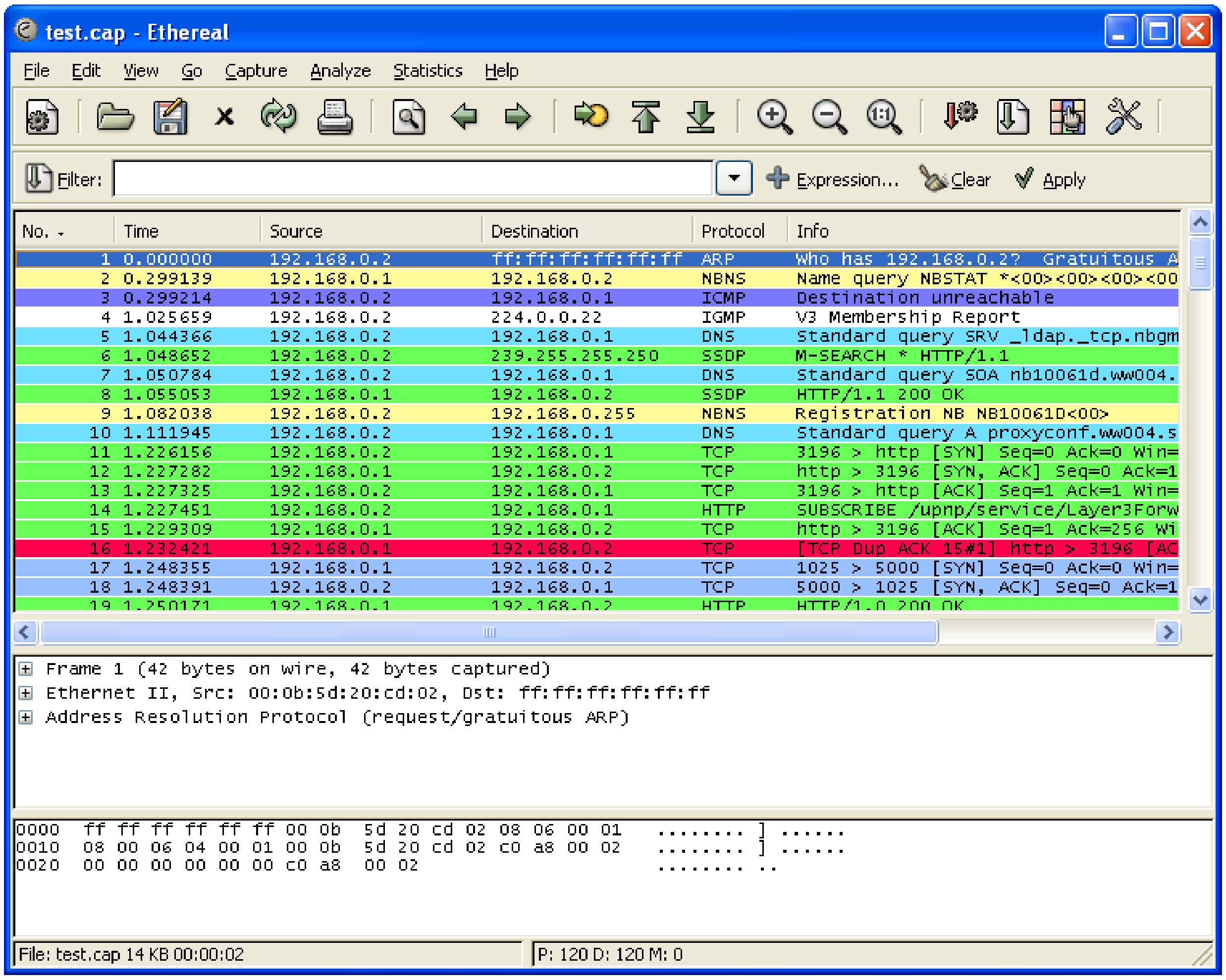
After compiling the software (or installing the binary packages available), the program can be started by double-clicking on the icon. If a user who is not the root is allowed to run the program with root privileges, he can not only capture data on the network, but should any security holes be found 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 pane 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 hex dump, 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 8.1).



```
No. -  Time       Source         Destination        Protocol  Info
  1 0.000000   192.168.0.2    ff:ff:ff:ff:ff:ff  ARP       Who has 192.168.0.2?  Gratuitous A
  2 0.299139   192.168.0.1    192.168.0.2        NBNS      Name query NBSTAT *<00><00><00><00
  3 0.299214   192.168.0.2    192.168.0.1        ICMP      Destination unreachable
  4 1.025659   192.168.0.2    224.0.0.22         IGMP      V3 Membership Report
  5 1.044366   192.168.0.2    192.168.0.1        DNS       Standard query SRV _ldap._tcp.nbgm
  6 1.048652   192.168.0.2    239.255.255.250    SSDP      M-SEARCH * HTTP/1.1
  7 1.050784   192.168.0.2    192.168.0.1        DNS       Standard query SOA nb10061d.ww004.
  8 1.055053   192.168.0.1    192.168.0.2        SSDP      HTTP/1.1 200 OK
  9 1.082038   192.168.0.2    192.168.0.255      NBNS      Registration NB NB10061D<00>
 10 1.111945   192.168.0.2    192.168.0.1        DNS       Standard query A proxyconf.ww004.s
 11 1.226156   192.168.0.2    192.168.0.1        TCP       3196 > http [SYN] Seq=0 Ack=0 Win=
 12 1.227282   192.168.0.1    192.168.0.2        TCP       http > 3196 [SYN, ACK] Seq=0 Ack=1
 13 1.227325   192.168.0.2    192.168.0.1        TCP       3196 > http [ACK] Seq=1 Ack=1 Win=
 14 1.227451   192.168.0.2    192.168.0.1        HTTP      SUBSCRIBE /upnp/service/Layer3Forw
 15 1.229309   192.168.0.1    192.168.0.2        TCP       http > 3196 [ACK] Seq=1 Ack=256 Wi
 16 1.232421   192.168.0.1    192.168.0.2        TCP       [TCP Dup ACK 15#1] http > 3196 [AC
 17 1.248355   192.168.0.1    192.168.0.2        TCP       1025 > 5000 [SYN] Seq=0 Ack=0 Win=
 18 1.248391   192.168.0.2    192.168.0.1        TCP       5000 > 1025 [SYN, ACK] Seq=0 Ack=1
 19 1.250171   192.168.0.1    192.168.0.2        HTTP      HTTP/1.0 200 OK

⊞ Frame 1 (42 bytes on wire, 42 bytes captured)
⊞ Ethernet II, Src: 00:0b:5d:20:cd:02, Dst: ff:ff:ff:ff:ff:ff
⊞ Address Resolution Protocol (request/gratuitous ARP)

0000  ff ff ff ff ff ff 00 0b  5d 20 cd 02 08 06 00 01   ........ ] ......
0010  08 00 06 04 00 01 00 0b  5d 20 cd 02 c0 a8 00 02   ........ ] ......
0020  00 00 00 00 00 00 c0 a8  00 02                     ........ ..

File: test.cap 14 KB 00:00:02          P: 120 D: 120 M: 0
```

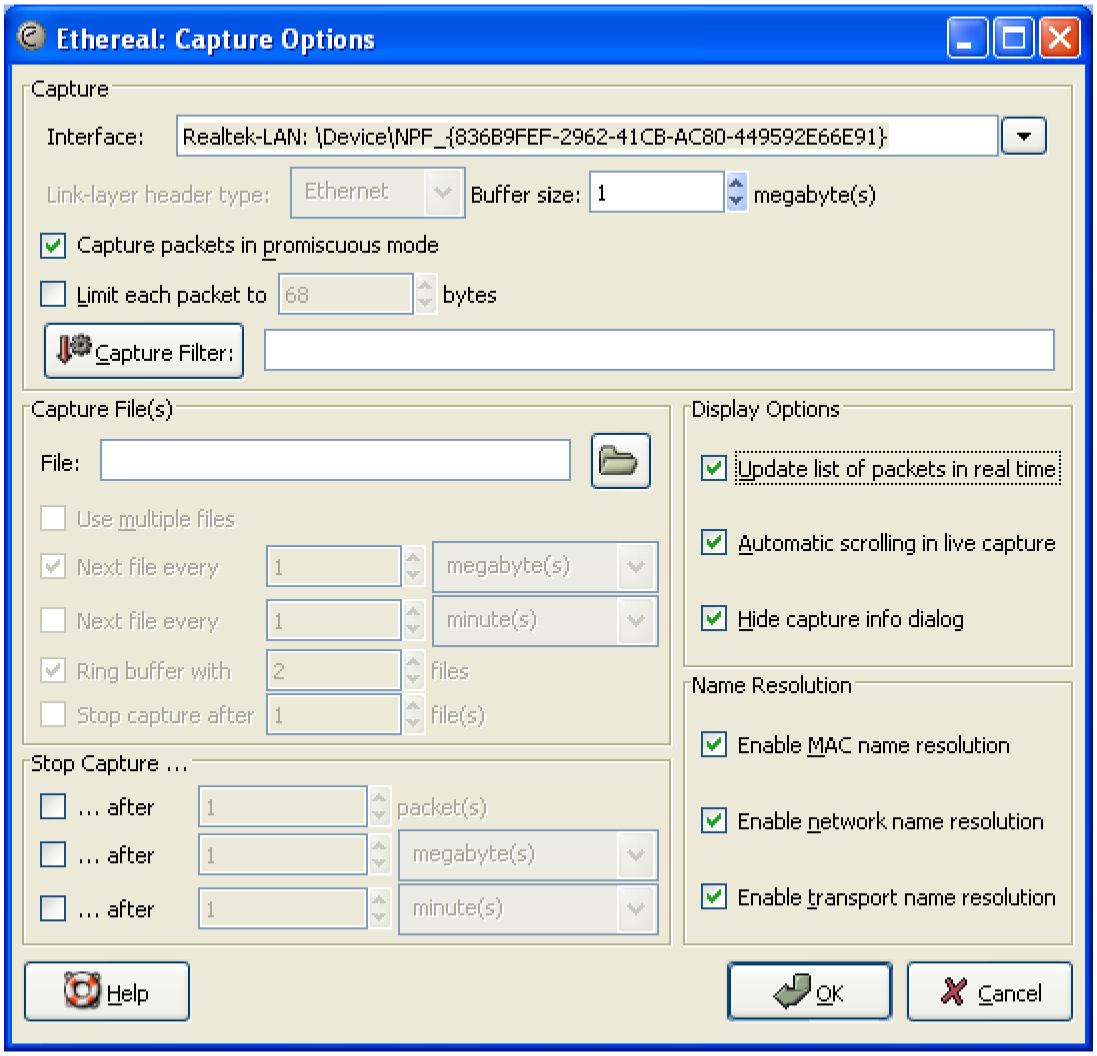
Fig. 8.1 Output Window

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 intrusions 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 8.2).


Ethereal: Capture Options

Capture
Interface: Realtek-LAN: \Device\NPF_{836B9FEF-2962-41CB-AC80-449592E66E91}
Link-layer header type: Ethernet
Buffer size: 1 megabyte(s)
[✓] Capture packets in promiscuous mode
[ ] Limit each packet to 68 bytes
Capture Filter:

Capture File(s)
File:
[ ] Use multiple files
[✓] Next file every 1 megabyte(s)
[ ] Next file every 1 minute(s)
[✓] Ring buffer with 2 files
[ ] Stop capture after 1 file(s)

Stop Capture ...
[ ] ... after 1 packet(s)
[ ] ... after 1 megabyte(s)
[ ] ... after 1 minute(s)

Display Options
[✓] Update list of packets in real time
[✓] Automatic scrolling in live capture
[✓] Hide capture info dialog

Name Resolution
[✓] Enable MAC name resolution
[✓] Enable network name resolution
[✓] Enable transport name resolution

Help OK Cancel

Fig.8.2 Capture Options

Capture Frame

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.

2. **Link-layer header type**

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

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.

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).

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.

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 labeled Capture Filter, and Ethereal will bring up the Capture Filters dialog box and allow you to create and/or select a filter.

Capture File(s) frame

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.

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. **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.

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.

5. **Ring buffer with n files**

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

6. **Stop capture after n file(s)**

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

Stop Capture... frame

1. **after n packet(s)**

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

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. **after n minute(s)**

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

Display Options frame

1. **Update list of packets in real time**

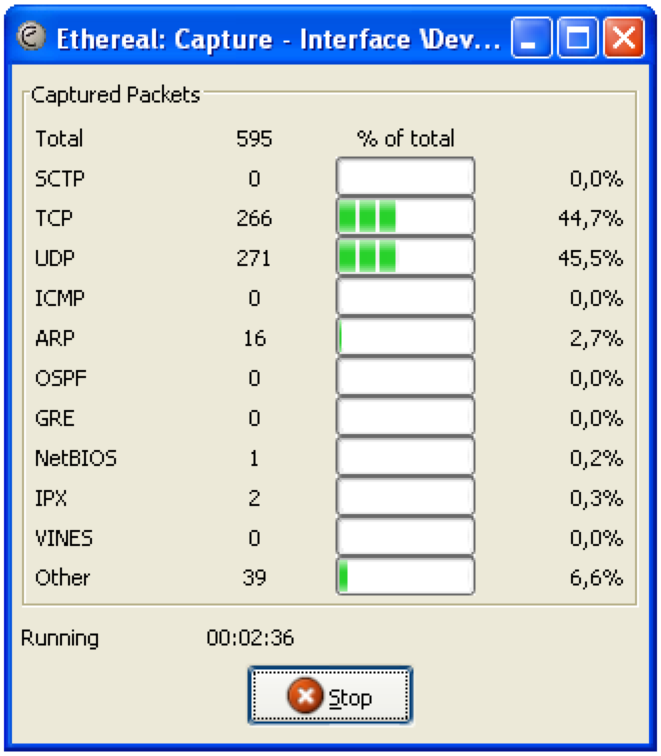
This option allows you to specify that Ethereal should update the packet list pane in real time. If you do not specify this, Ethereal does not display any packets until you stop the capture. When you check this, Ethereal captures in a separate process and feeds the captures to the display process.

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 grayed out if "Update list of packets in real time" is disabled.

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.



Captured Packets

Total	595	% of total	
SCTP	0		0,0%
TCP	266		44,7%
UDP	271		45,5%
ICMP	0		0,0%
ARP	16		2,7%
OSPF	0		0,0%
GRE	0		0,0%
NetBIOS	1		0,2%
IPX	2		0,3%
VINES	0		0,0%
Other	39		6,6%

Running 00:02:36

Fig.8.3 Capture Info

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.

1. **Enable MAC name resolution**

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

2. **Enable network name resolution**

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

3. **Enable transport name resolution**

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

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. he "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.
5. Use "Browse for other folders" to browse files and folders in your file system.
6. 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.
7. Click on the Cancel button to go back to Ethereal and not save the captured packets.

Opening captured files

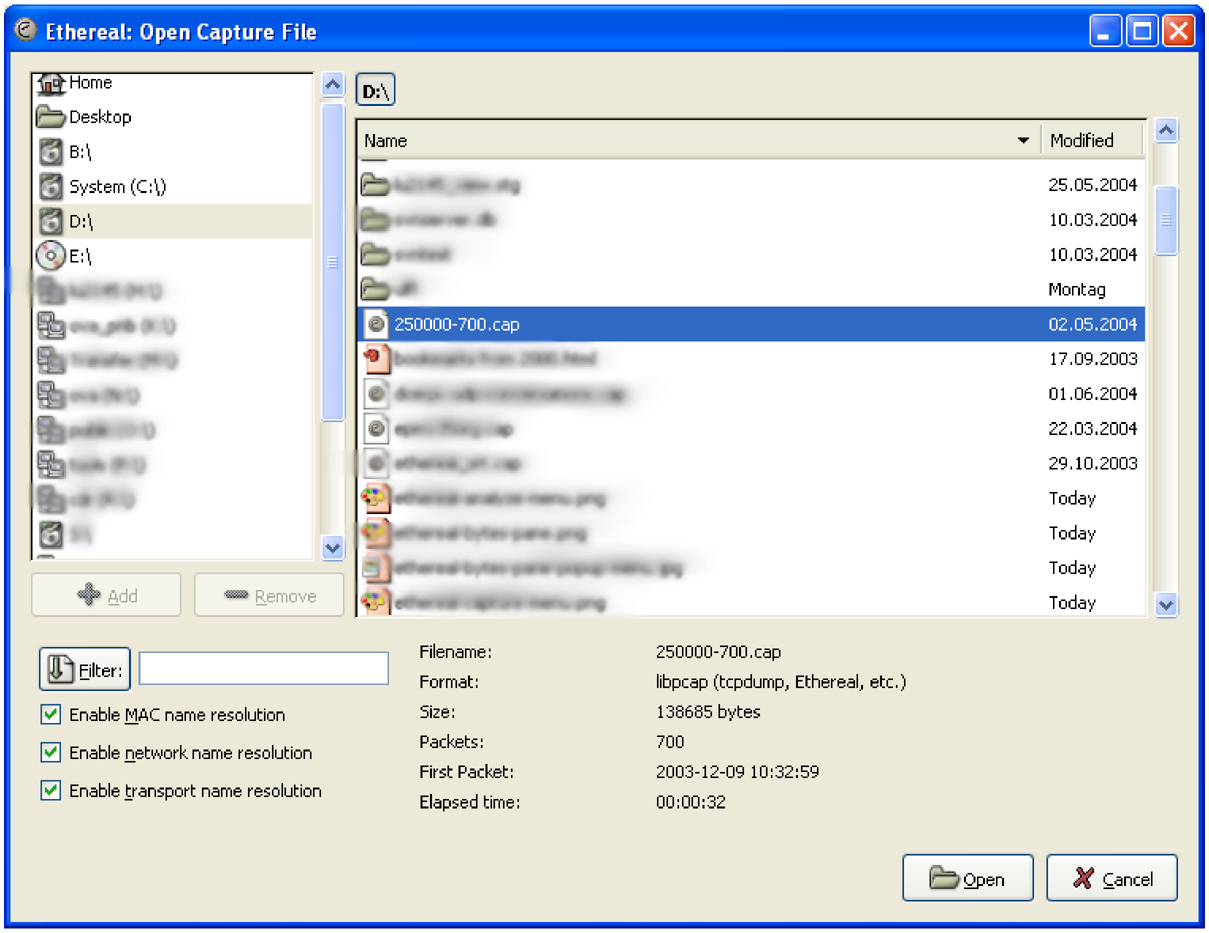
Ethereal can read in previously saved capture files. To read them, simply select the Open menu item from the File menu. Ethereal will then pop up the File Open dialog box (Figure 4).

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.

You can change the display filter and name resolution settings later while viewing the packets. However, for very large capture files it can take a significant amount of time changing these settings, so it might be a good idea to set them in advance here.



Name	Modified
[illegible]	25.05.2004
[illegible]	10.03.2004
[illegible]	10.03.2004
[illegible]	Montag
250000-700.cap	02.05.2004
[illegible]	17.09.2003
[illegible]	01.06.2004
[illegible]	22.03.2004
[illegible]	29.10.2003
[illegible]	Today
[illegible]	Today
[illegible]	Today
[illegible]	Today

✓ Enable MAC name resolution
✓ Enable network name resolution
✓ Enable transport name resolution

Filename:	250000-700.cap
Format:	libpcap (tcpdump, Ethereal, etc.)
Size:	138685 bytes
Packets:	700
First Packet:	2003-12-09 10:32:59
Elapsed time:	00:00:32

Fig. 8.4 Open 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.

FAQs:

1. List out uses of Wireshark.
2. What protocols are supported by Wireshark?
3. What is the use of capture filter?
4. How to save the output file.
5. How do I put an interface into promiscuous mode?

PRACTISE ASSIGNMENTS

Study any other protocol analyzer tool and compare it with Wireshark.

1. SniffPass
2. SoftPerfect Network Protocol Analyzer

Assignment 9 Study Of Port Scanner Tool

AIM **:**Configure and demonstrate use of vulnerability assessment tool such as NESSUS.

OBJECTIVE: Study any port scanner tool and use its implementation features.

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 configure 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.

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.

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.

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.

The use of the Unix client

The 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
----------
```

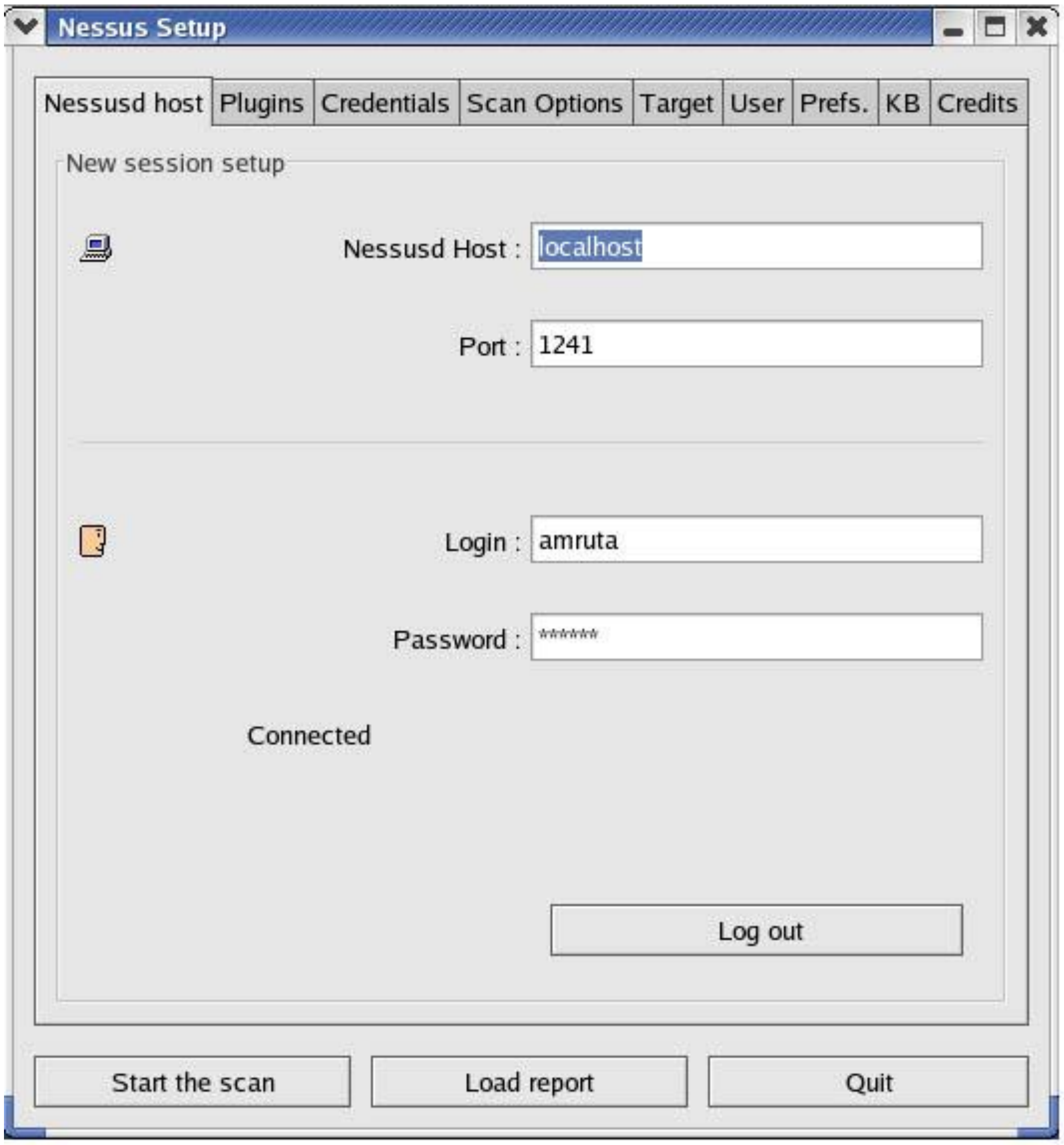
```
Enter the rules for this user, and hit ctrl-D once you are done :
(the user can have an empty rules set)

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.
```

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
The client configuration



Nessusd Host : localhost

Port : 1241

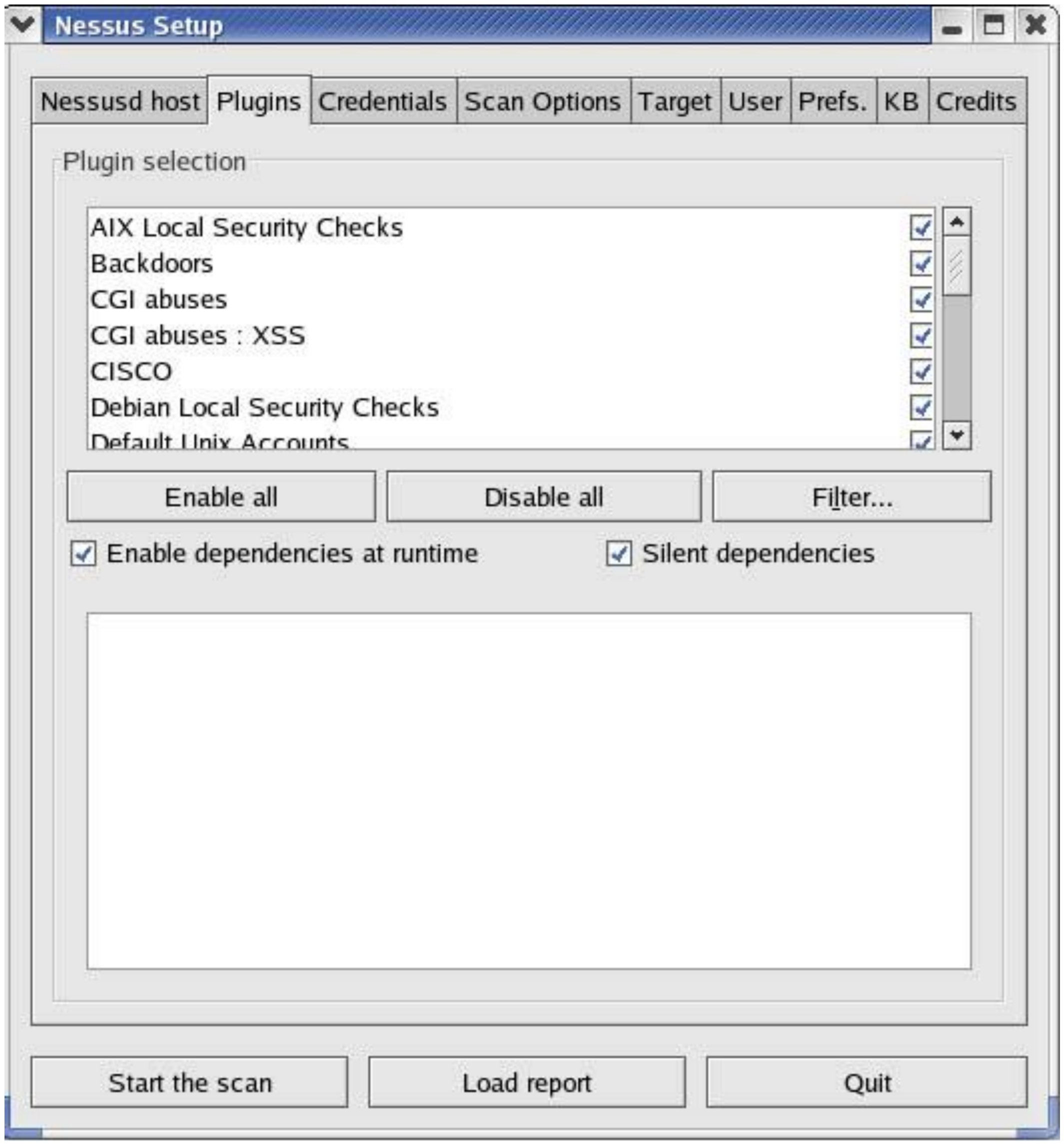
Login : amruta

Password : *******

Connected

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

The security checks configuration

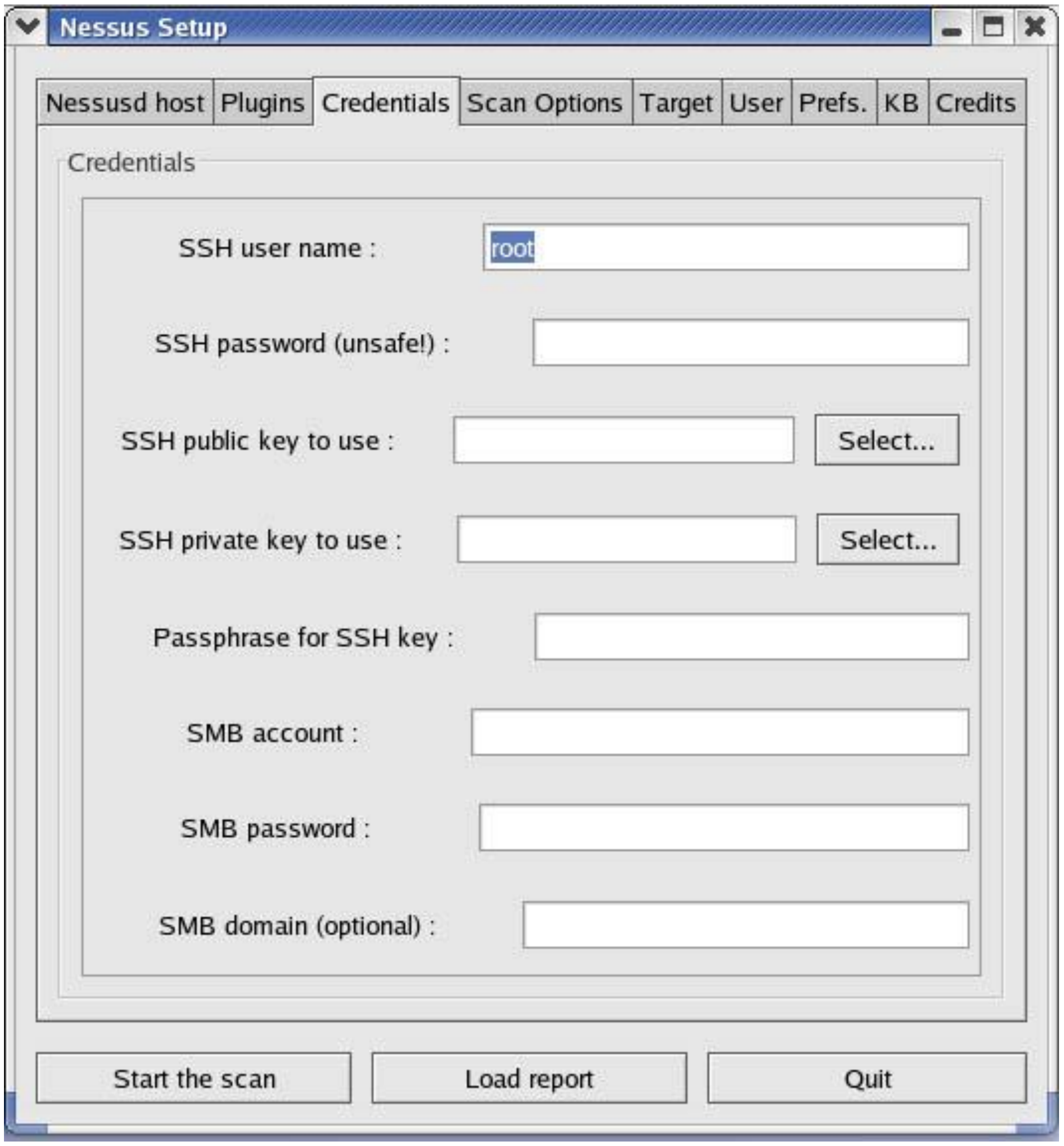


AIX Local Security Checks ✓
Backdoors ✓
CGI abuses ✓
CGI abuses : XSS ✓
CISCO ✓
Debian Local Security Checks ✓
Default Unix Accounts ✓

✓ Enable dependencies at runtime
✓ Silent dependencies

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

The plugins preferences



Credentials

SSH user name : root

SSH password (unsafe!) :

SSH public key to use :

SSH private key to use :

Passphrase for SSH key :

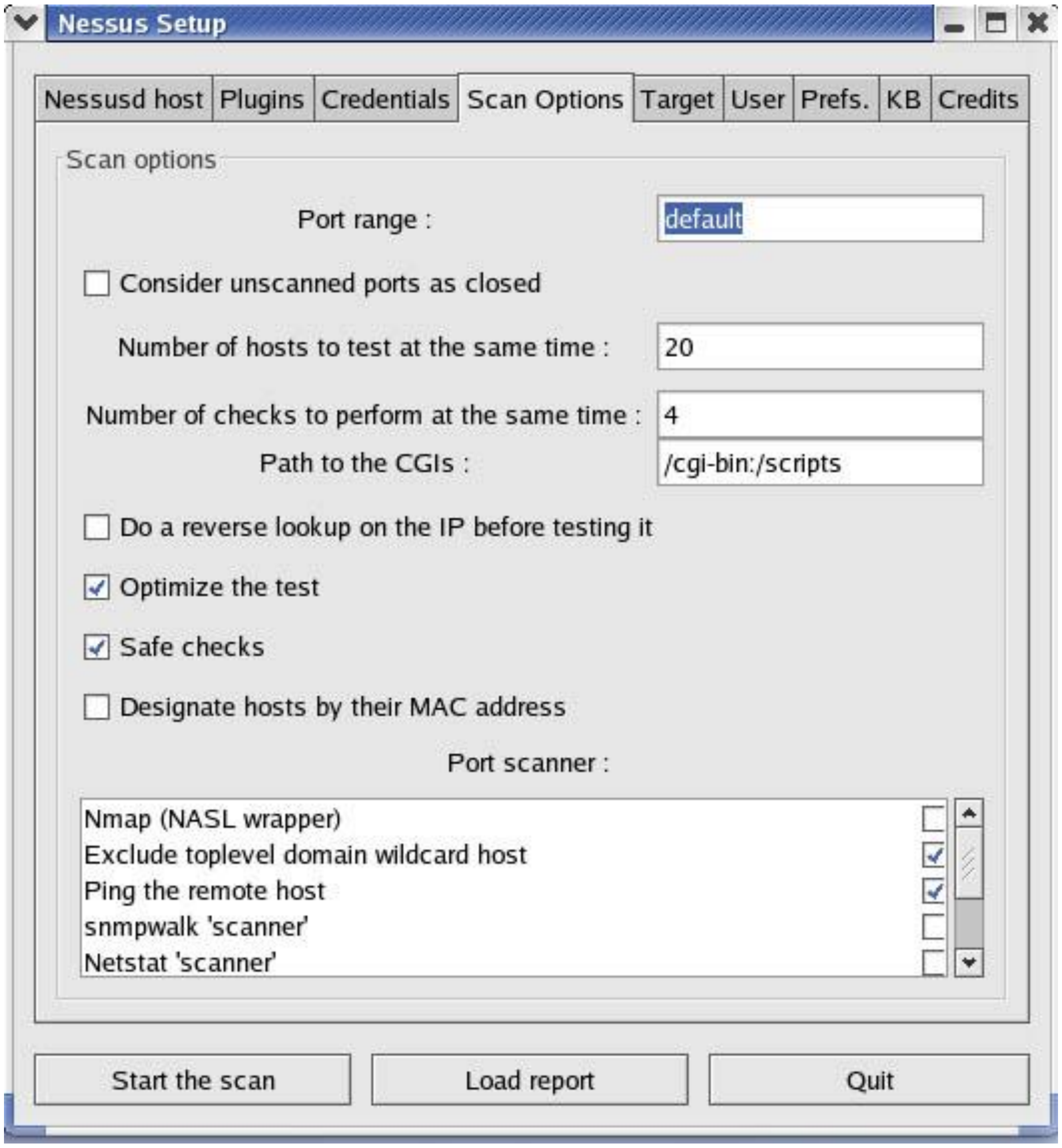
SMB account :

SMB password :

SMB domain (optional) :

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.

The scan options


Nessus Setup

Nessusd host | Plugins | Credentials | Scan Options | Target | User | Prefs. | KB | Credits

Scan options

Port range : default

☐ Consider unscanned ports as closed

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

☑ Optimize the test

☑ Safe checks

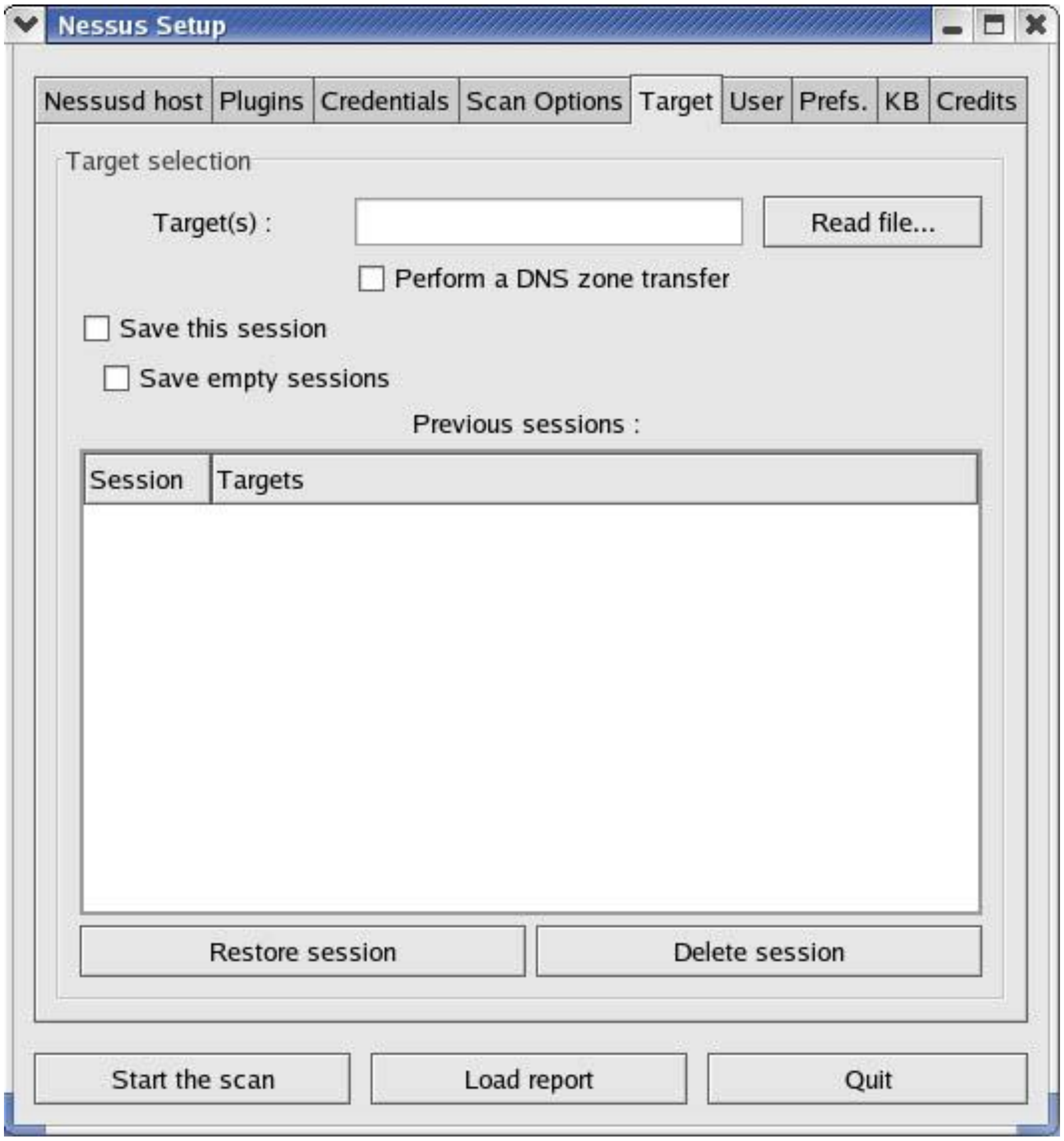
☐ Designate hosts by their MAC address

Port scanner :

Nmap (NASL wrapper)	☐
Exclude toplevel domain wildcard host	☑
Ping the remote host	☑
snmpwalk 'scanner'	☐
Netstat 'scanner'	☐

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.

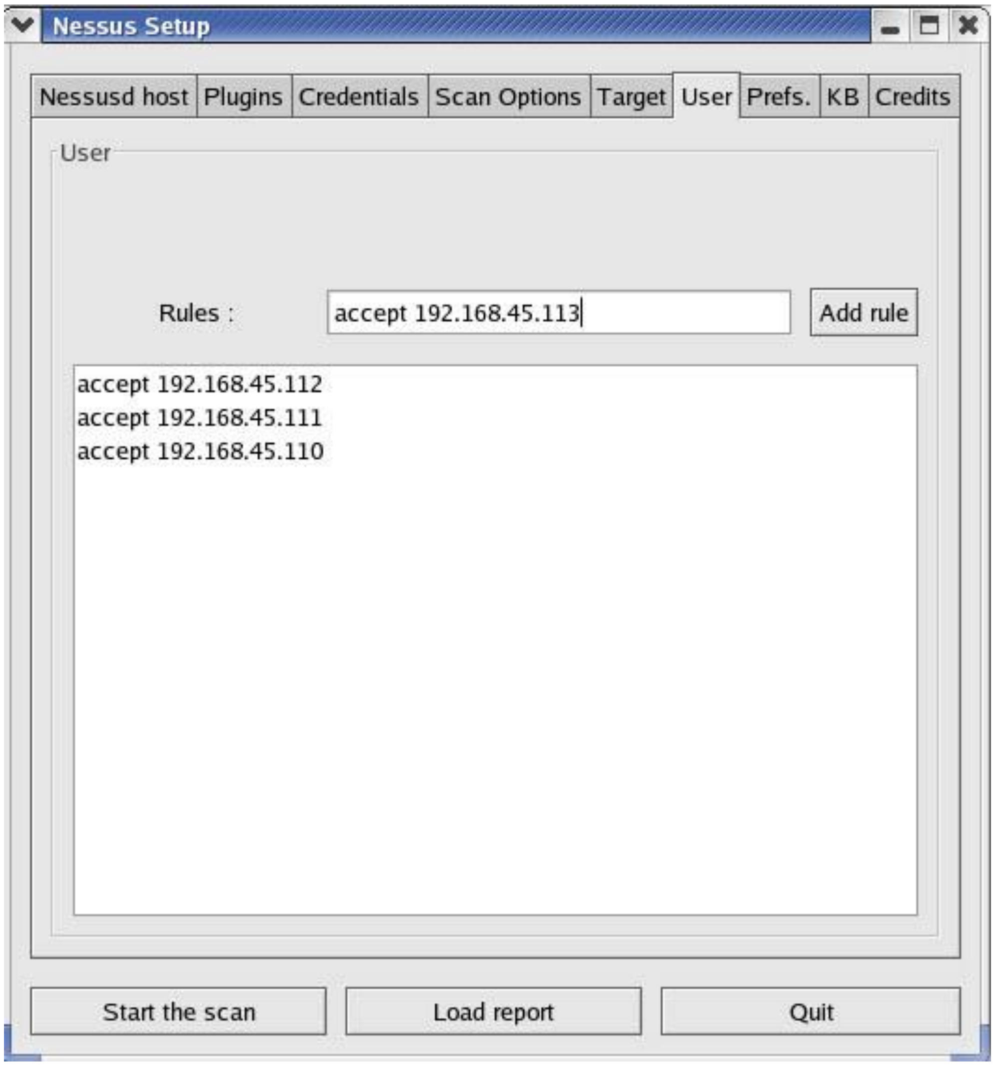
Define the targets



The hosts of my local network are using private IP adresses, 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
The rules section

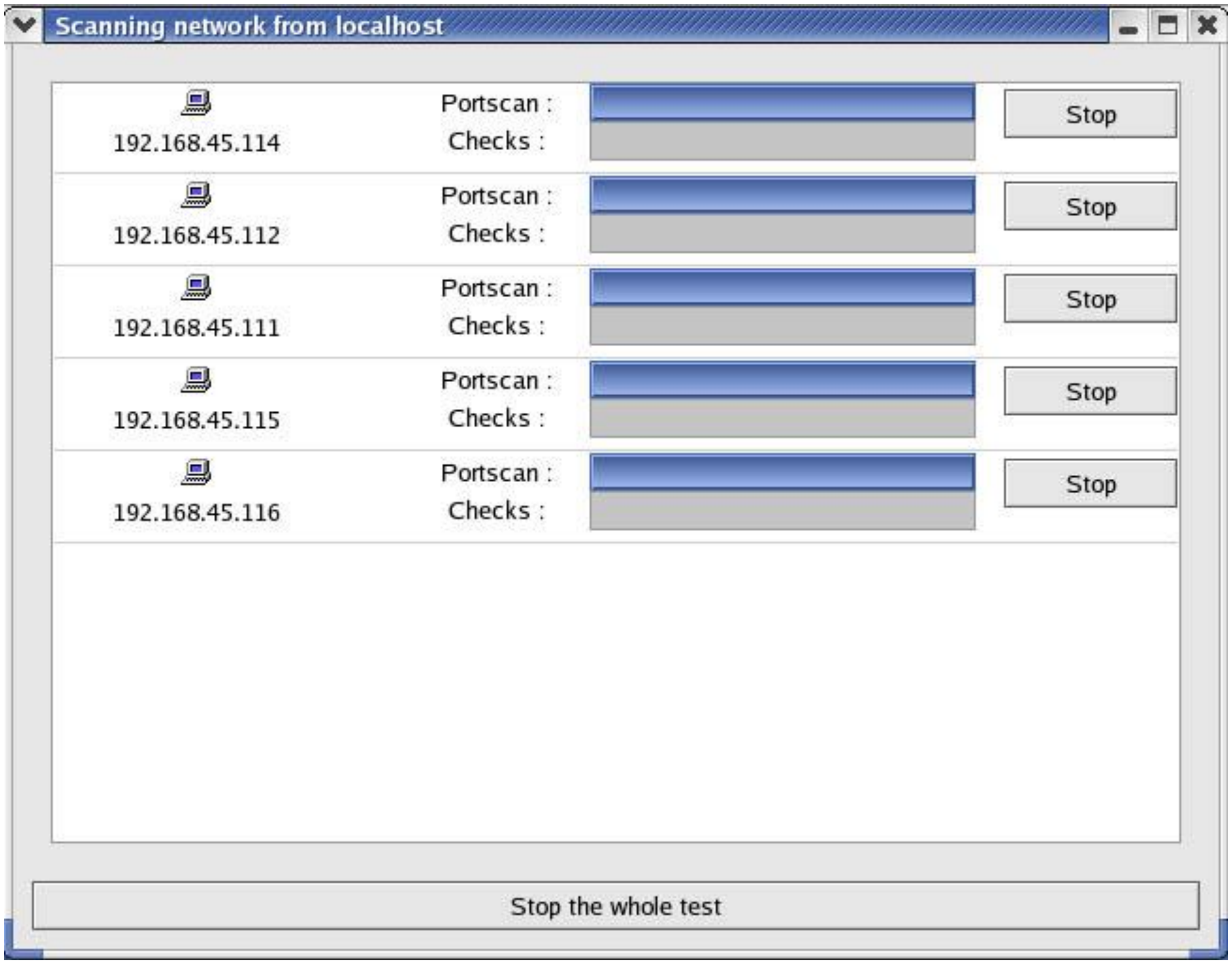


Rules : accept 192.168.45.113

```
accept 192.168.45.112
accept 192.168.45.111
accept 192.168.45.110
```

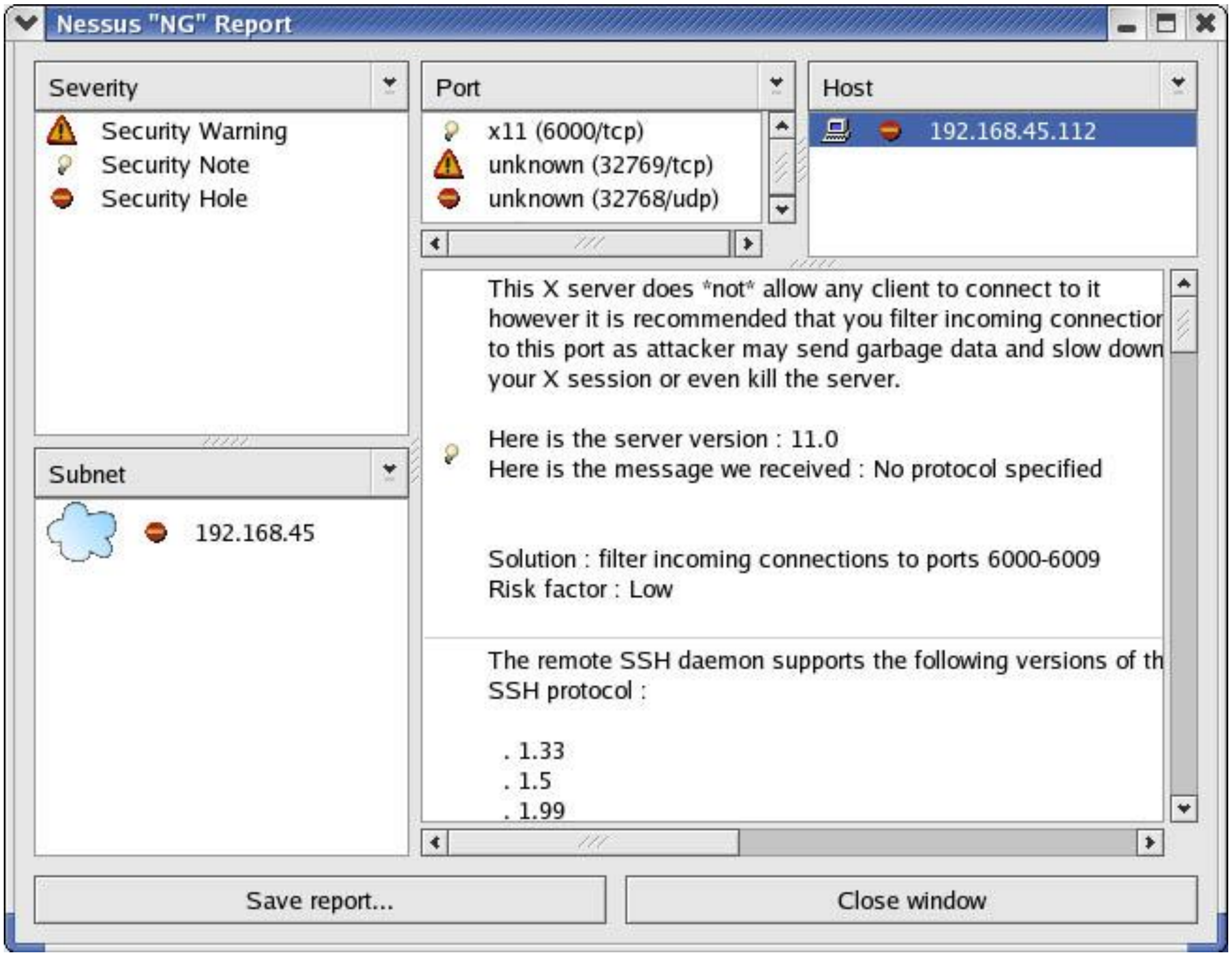
The rules allow a user to restrict his test. For instance, I want to test 10.163.156.1/24, except 10.163.156.5. The rule set I Entered allows me to do that.

Start the test



Scanning network from localhost

192.168.45.114	Portscan : Checks :		Stop
192.168.45.112	Portscan : Checks :		Stop
192.168.45.111	Portscan : Checks :		Stop
192.168.45.115	Portscan : Checks :		Stop
192.168.45.116	Portscan : Checks :		Stop



x11 (6000/tcp)
unknown (32769/tcp)
unknown (32768/udp)

192.168.45.112

192.168.45

```
This X server does *not* allow any client to connect to it
however it is recommended that you filter incoming connectior
to this port as attacker may send garbage data and slow down
your X session or even kill the server.

Here is the server version : 11.0
Here is the message we received : No protocol specified


Solution : filter incoming connections to ports 6000-6009
Risk factor : Low

The remote SSH daemon supports the following versions of th
SSH protocol :

 . 1.33
 . 1.5
 . 1.99
```

FAQs:

1. What is use of Nessus?
2. What is plugin architecture?
3. What is role of client and server in Nessus?

PRACTISE ASSIGNMENTS:

Study any other port scanner tool and compare its features with NESSUS.

1. NetScanTools