

*Department of Computer Engineering*

# **Lab Manual**

**Third Year Semester-VI**

**Subject:** Cryptography and Cryptography and  
System Security Lab (CSSL)

### **Vision**

To foster and permeate higher and quality education with value added engineering, technology programs, providing all facilities in terms of technology and platforms for all round development with societal awareness and nurture the youth with international competencies and exemplary level of employability even under highly competitive environment so that they are innovative adaptable and capable of handling problems faced by our country and world at large.

### **Mission**

The Institution is committed to mobilize the resources and equip itself with men and materials of excellence thereby ensuring that the Institution becomes pivotal center of service to Industry, academia, and society with the latest technology. RAIT engages different platforms such as technology enhancing Student Technical Societies, Cultural platforms, Sports excellence centers, Entrepreneurial Development Center and Societal Interaction Cell. To develop the college to become an autonomous Institution & deemed university at the earliest with facilities for advanced research and development programs on par with international standards. To invite international and reputed national Institutions and Universities to collaborate with our institution on the issues of common interest of teaching and learning sophistication.

## **Institute Vision, Mission & Quality Policy**

---

### **Quality Policy**

ज्ञानधीनं जगत् सर्वम ।  
**Knowledge is supreme.**

#### **Our Quality Policy**

**It is our earnest endeavor to produce high quality engineering professionals who are innovative and inspiring, thought and action leaders, competent to solve problems faced by society, nation and world at large by striving towards very high standards in learning, teaching**

## Department Vision & Mission

---

### **Vision**

To impart higher and quality education in computer science with value added engineering and technology programs to prepare technically sound, ethically strong engineers with social awareness. To extend the facilities, to meet the fast-changing requirements and nurture the youths with international competencies and exemplary level of employability and research under highly competitive environments.

### **Mission**

- To mobilize the resources and equip the institution with men and materials of excellence to provide knowledge and develop technologies in the thrust areas of computer science and Engineering.
- To provide the diverse platforms of sports, technical, co-curricular and extracurricular activities for the overall development of student with ethical attitude.
- To prepare the students to sustain the impact of computer education for social needs encompassing industry, educational institutions and public service.
- To collaborate with IITs, reputed universities and industries for the technical and overall upliftment of students for continuing learning and entrepreneurship.

# Departmental Program Educational Objectives (PEOs)

---

## **1. Learn and Integrate**

To provide Computer Engineering students with a strong foundation in the mathematical, scientific and engineering fundamentals necessary to formulate, solve and analyze engineering problems and to prepare them for graduate studies.

## **2. Think and Create**

To develop an ability to analyze the requirements of the software and hardware, understand the technical specifications, create a model, design, implement and verify a computing system to meet specified requirements while considering real-world constraints to solve real world problems.

## **3. Broad Base**

To provide broad education necessary to understand the science of computer engineering and the impact of it in a global and social context.

## **4. Techno-leader**

To provide exposure to emerging cutting-edge technologies, adequate training & opportunities to work as teams on multidisciplinary projects with effective communication skills and leadership qualities.

## **5. Practice citizenship**

To provide knowledge of professional and ethical responsibility and to contribute to society through active engagement with professional societies, schools, civic organizations or other community activities.

## **6. Clarify Purpose and Perspective**

To provide strong in-depth education through electives and to promote student awareness on the life-long learning to adapt to innovation and change, and to be successful in their professional work or graduate studies.

## Departmental Program Outcomes (POs)

---

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

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

**PO3: Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

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

**PO5: Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

**PO6: The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

**PO7: Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

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

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

**P10: Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

**PO11: Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

**PO12: Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

---

## Departmental Program Specific Outcomes (PSOs)

---

**PSO1:** To build competencies towards problem solving with an ability to understand, identify, analyze and design the problem, implement and validate the solution including both hardware and software.

**PSO2:** To build appreciation and knowledge acquiring of current computer techniques with an ability to use skills and tools necessary for computing practice.

**PSO3:** To be able to match the industry requirements in the area of computer science and engineering. To equip skills to adopt and imbibe new technologies.

# Index

Sr. No.	Contents	Page No.
1.	List of Experiments	1
2.	Course Objective, Course Outcome & Experiment Plan	2,3
3.	CO-PO, CO-PSO Mapping	4
4.	Study and Evaluation Scheme	5
5.	Experiment No. 1	6
6.	Experiment No. 2	11
7.	Experiment No. 3	18
8.	Experiment No. 4	22
9.	Experiment No. 5	26
10.	Experiment No. 6	36
11.	Experiment No. 7	42
12.	Experiment No. 8	46
13.	Experiment No. 9	52
14.	Experiment No. 10	56
15.	Experiment No. 11	72

# List of Experiments

Sr. No.	Experiments Name
1	Design and Implementation of a product cipher using Substitution and Transposition ciphers
2	Implementation and analysis of RSA public key cryptosystem and Digital signature scheme.
3	Implementation of Diffie Hellman Key exchange algorithm
4	For varying message sizes, test integrity of message using MD-5, SHA-1, and analyze the performance of the two protocols. Use crypt APIs
5	Study the use of network reconnaissance tools like WHOIS, dig, traceroute, ns lookup to gather information about networks and domain registrars.
6	Study of packet sniffer tools: Wireshark: 1. Download and install Wireshark and capture icmp, tcp, and http packets in promiscuous mode. 2. Explore how the packets can be traced based on different filters.
7	Download and install nmap. Use it with different options to scan open ports, perform OS fingerprinting, do a ping scan, tcp port scan, udp port scan, xmas scan etc.
8	Simulate DOS attack using Hping3 and Wireshark.
9	Simulate buffer overflow attack using Splint, Cppcheck etc.
10	Setting up personal Firewall using iptables.
11	Implementation of Virus and Antivirus.



# Course Objectives & Course Outcome

## Course Outcomes:

CO1	To be able to apply the knowledge of symmetric cryptography to implement simple ciphers.
CO2	To be able to analyze and implement public key cryptosystem and Digital signature scheme like RSA and El Gamal.
CO3	To analyze and evaluate performance of hashing algorithms.
CO4	To explore the different network reconnaissance tools like sniffers, port scanners and other related tools to gather network related information.
CO5	To be able to set up firewalls and transport layer security using open-source technologies.
CO6	To be able to explore various attacks like buffer-overflow, and Denial of Service attacks.

# Experiment Plan

Module No.	Week No.	Experiments Name	Course Outcome	Weightage
1	W1	Design and Implementation of a product cipher using Substitution and Transposition ciphers	CO1	10
2	W2	Implementation and analysis of RSA public key cryptosystem and Digital signature scheme.	CO2	5
3	W3	Implementation of Diffie Hellman Key exchange algorithm	CO2	5
4	W4	For varying message sizes, test integrity of message using MD-5, SHA-1, and analyze the performance of the two protocols. Use crypt APIs	CO3	10
5	W5	Study the use of network reconnaissance tools like WHOIS, dig, traceroute, ns lookup to gather information about networks and domain registrars.	CO4	2.5
6	W6	Study of packet sniffer tools: Wireshark: 1. Download and install Wireshark and capture icmp, tcp, and http packets in promiscuous mode. 2. Explore how the packets can be traced based on different filters.	CO4	2.5
7	W7	Download and install nmap. Use it with different options to scan open ports, perform OS fingerprinting, do a ping scan, tcp port scan, udp port scan, xmas scan etc.	CO4	2.5
8	W8	Simulate DOS attack using Hping3 and Wireshark.	CO4	2.5
9	W9	Simulate buffer overflow attack using Splint, Cppcheck etc.	CO5	5
10	W10	Setting up personal Firewall using iptables.	CO5	5
11	W11	Implementation of Virus and Antivirus.	CO6	10

## Mapping Course Outcomes (CO) - Program Outcomes (PO)

Subject Weight	Course Outcomes		Contribution to Program outcomes (PO)											
			1	2	3	4	5	6	7	8	9	10	11	12
<b>PRATICAL</b> 80%	CO1	To be able to apply the knowledge of symmetric cryptography to implement simple ciphers.	3	2	2	1								2
	CO2	To be able to analyze and implement public key cryptosystem and Digital signature scheme like RSA and El Gamal.		2	2	2		1	1	1	1			
	CO3	To analyze and evaluate performance of hashing algorithms.			3	2	2	1		1				1
	CO4	To explore the different network reconnaissance tools like sniffers, port scanners and other related tools to gather network related information.		2		3	3					1		1
	CO5	To be able to set up firewalls and transport layer security using open source technologies.		2		2	3					1	1	1
	CO6	To be able to explore various attacks like buffer-overflow, and web-application attacks.		2		2	2			1		1	1	1

### Mapping of Course outcomes with Program Specific outcomes:

Course Outcomes		Contribution to Program Specific outcomes		
		PSO1	PSO2	PSO3
CO1	To be able to apply the knowledge of symmetric cryptography to implement simple ciphers.	3		
CO2	To be able to analyze and implement public key cryptosystem and Digital signature scheme like RSA and El Gamal.	3		1
CO3	To analyze and evaluate performance of hashing algorithms.	3		
CO4	To explore the different network reconnaissance tools like sniffers, port scanners and other related tools to gather network related information.	2	3	1
CO5	To be able to set up firewalls and transport layer security using open-source technologies.		3	2
CO6	To be able to explore various attacks like buffer-overflow and Denial of Service attacks.		3	1

# Study and Evaluation Scheme

Course Code	Course Name	Teaching Scheme			Credits Assigned			
CSL602	Cryptography and Cryptography and System Security Lab	Theory	Practical	Tutorial	Theory	Practical	Tutorial	Total
		--	02	--	--	01	--	01

Course Code	Course Name	Examination Scheme		
CSL602	Cryptography and Cryptography and System Security Lab	Term Work	Oral & practical	Total
		25	-	25

**Term Work:** Laboratory work will be based on above syllabus with minimum 10 experiments to be incorporated.

Experiments----- (15) Marks

Assignment----- (05) Marks

Attendance (Theory + Practical) ----- (05) Marks

**Total**----- **(25) Marks**

# **Cryptography and System Security Lab**

## **Experiment No.: 1**

**Design and Implementation of a product cipher  
using Substitution and Transposition ciphers**

# Experiment No. 1

1. **Aim:** Alice wants to send message “We are discovered. Save yourself” to Bob using Product Cipher. Both are agreed for key as n for additive cipher and number of rows=2 for transposition cipher. Write a program which helps Alice to encrypt the message and Bob will be able to decrypt it.

2. **Objectives:**

- To understand the encryption and decryption fundamentals.
- To apply the concepts of the product cipher.
- To understand the confusion and diffusion properties of a block cipher.

3. **Outcomes:** The learner will be able to

- Understand the principles and practices of cryptographic techniques

4. **Hardware / Software Required:** C/C++/JAVA

5. **Theory:**

Substitution cipher is a method of encryption by which units of plaintext are replaced with ciphertext according to a regular system; the "units" may be single letters (the most common), pairs of letters, triplets of letters, mixtures of the above, and so forth. The receiver deciphers the text by performing an inverse substitution.

Transposition cipher is a method of encryption by which the positions held by units of plaintext (which are commonly characters or groups of characters) are shifted according to a regular system, so that the ciphertext constitutes a permutation of the plaintext. That is, the order of the units is changed.

Substitution ciphers can be compared with Transposition ciphers. In a transposition cipher, the units of the plaintext are rearranged in a different and usually quite complex order, but the units themselves are left unchanged. By contrast, in a substitution cipher, the units of the plaintext are retained in the same sequence in the ciphertext, but the units themselves are altered.

1. Caesar Cipher: In cryptography, a Caesar cipher, also known as a Caesar's cipher, the shift cipher, Caesar's code or Caesar shift, is one of the simplest and most widely known encryption techniques. It is a type of substitution cipher in which each letter in the plaintext is replaced by a letter some fixed number of positions down the alphabet. For example, with a

shift of 3, A would be replaced by D, B would become E, and so on. The method is named after Julius Caesar, who used it to communicate with his generals.

Example:

The transformation can be represented by aligning two alphabets; the cipher alphabet is the plain alphabet rotated left or right by some number of positions. For instance, here is a Caesar cipher using a left rotation of three places (the shift parameter, here 3, is used as the key):

Plain: ABCDEFGHIJKLMNOPQRSTUVWXYZ

Cipher: DEFGHIJKLMNOPQRSTUVWXYZABC

When encrypting, a person looks up each letter of the message in the "plain" line and writes down the corresponding letter in the "cipher" line. Deciphering is done in reverse.

Plaintext: the quick brown fox jumps over the lazy dog

Ciphertext: WKH TXLFN EURZQ IRA MXPSV RYHU WKH ODCB GRJ

2. Columnar Transposition: In a columnar transposition, the message is written out in rows of a fixed length, and then read out again column by column, and the columns are chosen in some scrambled order. Both the width of the rows and the permutation of the columns are usually defined by a keyword. For example, the word ZEBRAS is of length 6 (so the rows are of length 6), and the permutation is defined by the alphabetical order of the letters in the keyword. In this case, the order would be "6 3 2 4 1 5".

In a regular columnar transposition cipher, any spare spaces are filled with nulls; in an irregular columnar transposition cipher, the spaces are left blank. Finally, the message is read off in columns, in the order specified by the keyword. For example, suppose we use the keyword ZEBRAS and the message WE ARE DISCOVERED. FLEE AT ONCE. In a regular columnar transposition, we write this into the grid as:

6 3 2 4 1 5 W E A R E D I S C O V E R E D F L E E A T O N C E Q K J E U

The ciphertext is then read off as:

EVLNE ACDTK ESEAQ ROFOJ DEECU WIREE

## **6. Procedure/ Program:**

### **1. Substitution Encryption-**

- a. Accept plaintext, P from user
- b. Accept key, K from user.
- c. Generate ciphertext,  $C=(P+K)\text{mod } 26$
- d. Display plaintext, P and ciphertext, C.

### **2. Transposition Encryption:**

- a. Count how many letters are in your ciphertext, C (for example, 75) and factor that number ( $75 = 5 \times 5 \times 3$ ).
- b. Create all of the possible matrices to fit this plaintext (in our case,  $3 \times 25$ ,  $5 \times 15$ ,  $15 \times 5$ ,  $25 \times 3$ ).
- c. Write the ciphertext, C row-wise into these matrices.
- d. Permute the columns. (Shuffle the columns)
- e. Read the matrix in column-wise to get the new ciphertext, C1.
- f. Display plaintext C and ciphertext C1.

### **3. Transposition Decryption:**

- a. Write the ciphertext, C1 column-by-column.
- b. Permute the columns.
- c. Read the matrix row-wise to recover text C.
- d. Display plaintext C and ciphertext C1

### **4. Substitution decryption:**

- a. Generate plaintext,  $P=(C-K) \text{ mod } 26$
- b. Display plaintext, P and ciphertext, C.

## **7. Conclusion:**

A product cipher is a composite of two or more elementary ciphers with the goal of producing a cipher which is more secure than any of the individual components. In product cipher substitution and transposition are applied to create confusion and diffusion in the text message.



## **8. Viva Questions:**

- What is product cipher?
- What makes a product cipher secure?
- What is difference between the diffusion and confusion?

## **9. References:**

1. William Stallings, “*Cryptography and Network Security: Principles and Practice*”, Pearson education, Fifth edition.
2. Bernard Menezes, “*Network Security and Cryptography*”, Cengage Learning, Second Edition.
3. Behrouz A Forouzan, Debdeep Mukhopadhyay, “*Cryptography and Network Security*”, Tata McGraw Hill, Second edition
4. Behrouz A. Forouzan, “*Cryptography and Network Security*”, Tata McGraw Hill.
5. Charles P. Pfleeger, “*Security in Computing*”, Pearson Education.

# **Cryptography and System Security Lab**

## **Experiment No.: 2**

**Implementation and analysis of RSA public key cryptosystem and Digital signature scheme**

# Experiment No. 2

1. **Aim:** John wants to send encrypted message to Merry using RSA cryptography. John and Merry selects two random numbers to generate public and private key. Write a program to simulate secured communication and authentication between John and Merry.
2. **Objectives:**
  - To understand the concept of public key cryptosystem.
  - To implement and analyze the RSA cryptosystem.
  - To understand and implement RSA digital signature.
3. **Outcomes:** The learner will be able to  
To be able to analyze and implement public key cryptosystem and Digital signature scheme like RSA and El Gamal.
4. **Hardware / Software Required:** C/C++/JAVA/Python
5. **Theory:**

## RSA Cryptosystem

This cryptosystem is one the initial system. It remains most employed cryptosystem even today. The system was invented by three scholars Ron Rivest, Adi Shamir, and Len Adleman and hence, it is termed as RSA cryptosystem. We will see two aspects of the RSA cryptosystem, firstly generation of key pair and secondly encryption-decryption algorithms.

### Generation of RSA Key Pair

Each person or a party who desires to participate in communication using encryption needs to generate a pair of keys, namely public key and private key. The process followed in the generation of keys is described below –

- Generate the RSA modulus ( $n$ )
  - Select two large primes,  $p$  and  $q$ .
  - Calculate  $n=p*q$ . For strong unbreakable encryption, let  $n$  be a large number, typically a minimum of 512 bits.
- Find Derived Number ( $e$ )
  - Number  $e$  must be greater than 1 and less than  $(p-1)(q-1)$ .

- There must be no common factor for  $e$  and  $(p - 1)(q - 1)$  except for 1. In other words two numbers  $e$  and  $(p - 1)(q - 1)$  are coprime.
- Form the public key
  - The pair of numbers  $(n, e)$  form the RSA public key and is made public.
  - Interestingly, though  $n$  is part of the public key, difficulty in factorizing a large prime number ensures that attacker cannot find in finite time the two primes  $(p \& q)$  used to obtain  $n$ . This is strength of RSA.
- Generate the private key
  - Private Key  $d$  is calculated from  $p, q,$  and  $e$ . For given  $n$  and  $e$ , there is unique number  $d$ .
  - Number  $d$  is the inverse of  $e$  modulo  $(p - 1)(q - 1)$ . This means that  $d$  is the number less than  $(p - 1)(q - 1)$  such that when multiplied by  $e$ , it is equal to 1 modulo  $(p - 1)(q - 1)$ .
  - This relationship is written mathematically as follows –
$$ed = 1 \bmod (p - 1)(q - 1)$$

The Extended Euclidean Algorithm takes  $p, q,$  and  $e$  as input and gives  $d$  as output.

### Example

An example of generating RSA Key pair is given below. (For ease of understanding, the primes  $p \& q$  taken here are small values. Practically, these values are very high).

- Let two primes be  $p = 7$  and  $q = 13$ . Thus, modulus  $n = pq = 7 \times 13 = 91$ .
- Select  $e = 5$ , which is a valid choice since there is no number that is common factor of 5 and  $(p - 1)(q - 1) = 6 \times 12 = 72$ , except for 1.
- The pair of numbers  $(n, e) = (91, 5)$  forms the public key and can be made available to anyone whom we wish to be able to send us encrypted messages.
- Input  $p = 7, q = 13,$  and  $e = 5$  to the Extended Euclidean Algorithm. The output will be  $d = 29$ .
- Check that the  $d$  calculated is correct by computing –
$$de = 29 \times 5 = 145 = 1 \bmod 72$$
- Hence, public key is  $(91, 5)$  and private keys is  $(91, 29)$ .

## Encryption and Decryption

Once the key pair has been generated, the process of encryption and decryption are relatively straightforward and computationally easy.

### RSA Encryption

- Suppose the sender wish to send some text message to someone whose public key is  $(n, e)$ .
- The sender then represents the plaintext as a series of numbers less than  $n$ .
- To encrypt the first plaintext  $P$ , which is a number modulo  $n$ . The encryption process is simple mathematical step as –

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

- In other words, the ciphertext  $C$  is equal to the plaintext  $P$  multiplied by itself  $e$  times and then reduced modulo  $n$ . This means that  $C$  is also a number less than  $n$ .
- Returning to our Key Generation example with plaintext  $P = 10$ , we get ciphertext  $C$  –

$$C = 105 \bmod 91$$

### RSA Decryption

- The decryption process for RSA is also very straightforward. Suppose that the receiver of public-key pair  $(n, e)$  has received a ciphertext  $C$ .
- Receiver raises  $C$  to the power of his private key  $d$ . The result modulo  $n$  will be the plaintext  $P$ .

$$\text{Plaintext} = C^d \bmod n$$

- Returning again to our numerical example, the ciphertext  $C = 82$  would get decrypted to number 10 using private key 29 –

$$\text{Plaintext} = 82^{29} \bmod 91 = 10$$

<b>Key Pair</b> Public key: $n = 55, e = 3$ Private key: $n = 55, d = 7$			<b>Key Pair Generation</b> Primes: $p = 5, q = 11$ Modulus: $n = pq = 55$ Public exponent: $e = 3$ Private exponent: $d = 3^{-1} \bmod 20 = 7$			
<b>Message</b>	<b>Encryption</b> $c = m^3 \bmod n$		<b>Decryption</b> $m = c^7 \bmod n$			
$m$	$m^2 \bmod n$	$m^3 \bmod n$	$c^2 \bmod n$	$c^3 \bmod n$	$c^6 \bmod n$	$c^7 \bmod n$
0	0	0	0	0	0	0
1	1	1	1	1	1	1
2	4	8	9	17	14	2
3	9	27	14	48	49	3
4	16	9	26	14	31	4
5	25	15	5	20	15	5
6	36	51	16	46	26	6
7	49	13	4	52	9	7
8	9	17	14	18	49	8
9	26	14	31	49	36	9

### RSA Digital Signature:

A digital signature is a mathematical scheme for demonstrating the authenticity of a digital message or documents. A valid digital signature gives a recipient reason to believe that the message was created by a known sender, that the sender cannot deny having sent the message (authentication and non-repudiation), and that the message was not altered in transit.

**To sign:** use a private signing algorithm

**To verify:** use a public verification algorithm

Alice wants to sign message  $m$ . She computes the signature of  $m$  (let's call it  $y$ ) and sends the signed message  $(m, y)$  to Bob. Bob gets  $(m, y)$ , runs the verification algorithm on it. The algorithm returns "true" iff  $y$  is Alice's signature of  $m$ .

The basic protocol:

1. Alice encrypts the document with her private key.
2. Alice sends the signed document to Bob.
3. Bob decrypts the document with Alice's public key.

## 6. Procedure/ Program:

1. Take choice as an input
2. If choice=1
  - a. Choose two large prime numbers P and Q.
  - b. Calculate  $N = P \times Q$ .
  - c. Select the public key (i.e. the encryption key) E such that it is not a factor of (P-1) and (Q-1).
  - d. Select the private key (i.e. the decryption key) D such that the following equation is true:  
 $(D \times E) \bmod (P-1) \times (Q-1) = 1$
  - e. For encryption, calculate the ciphertext CT from the plain text PT as follows:  
 $CT = PT^E \bmod N$
  - f. Send CT as the cipher text to the receiver.
  - g. For decryption, calculate the plaintext PT from the ciphertext CT as follows:  
 $PT = CT^D \bmod N$ .
3. If choice=2
  - a. Alice chooses secret odd primes p,q and computes  $n=pq$ .
  - b. Alice chooses  $e_A$  with  $\gcd(e_A, \Phi(n))=1$ .
  - c. Alice computes  $d_A = e_A^{-1} \bmod \Phi(n)$ .
  - d. Alice's signature is  $y = m^{d_A} \bmod n$ .
  - e. The signed message is (m,y).
  - f. Bob can verify the signature by calculating  $z = y^{e_A} \bmod n$ . (The signature is valid iff  $m=z$ ).

## 7. Conclusion:

RSA is a strong encryption algorithm. RSA implements a public-key cryptosystem that allows secure communications and “digital signatures”, and its security rests in part on the difficulty of factoring large numbers.

## 8. Viva Questions:

- What is RSA cryptosystem?
- What are the different attacks possible on RSA cryptosystem?

## 9. References:

1. William Stallings, “*Cryptography and Network Security: Principles and Practice*”, Pearson education, Fifth edition.
2. Bernard Menezes, “*Network Security and Cryptography*”, Cengage Learning, Second Edition.
3. Behrouz A Forouzan, Debdeep Mukhopadhyay, “*Cryptography and Network Security*”, Tata McGraw Hill, Second edition
4. Behrouz A. Forouzan, “*Cryptography and Network Security*”, Tata McGraw Hill.
5. Charles P. Pfleeger, “*Security in Computing*”, Pearson Education.



# **Cryptography and System Security Lab**

## **Experiment No.: 3**

### **Implementation of Diffie Hellman Key Exchange Algorithm**

# Experiment No. 3

1. **Aim:** Jenny and Sam are studying in 9<sup>th</sup> standard who want to generate keys for secured communication between them. They select one large prime number as a modulo and both are using their birth dates to calculate key.
2. **Objectives:**
  - To understand the principles of asymmetric key cryptography.
  - To understand the Diffie-Hellman Key exchange algorithm.
  - To understand the possible attacks on Diffie-Hellman.
3. **Outcomes:** The learner will be able to  
**Apply the cryptosystem to ensure privacy and integrity of information.**
4. **Hardware / Software Required:** C/C++/JAVA/Python
5. **Theory:**

## The Diffie-Hellman Algorithm

Diffie–Hellman key exchange (D–H) is a specific method of securely exchanging cryptographic keys over a public channel and was one of the first public-key protocols as originally conceptualized by Ralph Merkle and named after Whitfield Diffie and Martin Hellman. D–H is one of the earliest practical examples of public key exchange implemented within the field of cryptography. Traditionally, secure encrypted communication between two parties required that they first exchange keys by some secure physical channel, such as paper key lists transported by a trusted courier. The Diffie–Hellman key exchange method allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure channel. This key can then be used to encrypt subsequent communications using a symmetric key cipher.

The Diffie–Hellman key exchange algorithm solves the following dilemma. Alice and Bob want to share a secret key for use in a symmetric cipher, but their only means of communication is insecure. Every piece of information that they exchange is observed by their adversary Eve. How is it possible for Alice and Bob to share a key without making it available to Eve? At first glance it appears that Alice and Bob face an impossible task. It was a brilliant insight of Diffie and Hellman that the difficulty of the discrete logarithm problem for  $F_p^*$  provides a possible solution.

The first step is for Alice and Bob to agree on a large prime  $p$  and a nonzero integer  $g$  modulo  $p$ . Alice and Bob make the values of  $p$  and  $g$  public knowledge; for example, they might post the values on their web sites, so Eve knows them, too. For various reasons to be discussed later, it is best if they choose  $g$  such that its order in  $F^*$  is a large prime.

The next step is for Alice to pick a secret integer  $a$  that she does not reveal to anyone, while at the same time Bob picks an integer  $b$  that he keeps secret. Bob and Alice use their secret integers to compute

$$\underbrace{A \equiv g^a \pmod{p}}_{\text{Alice computes this}} \quad \text{and} \quad \underbrace{B \equiv g^b \pmod{p}}_{\text{Bob computes this}}.$$

They next exchange these computed values, Alice sends  $A$  to Bob and Bob sends  $B$  to Alice. Note that Eve gets to see the values of  $A$  and  $B$ , since they are sent over the insecure communication channel.

Finally, Bob and Alice again use their secret integers to compute

$$\underbrace{A' \equiv B^a \pmod{p}}_{\text{Alice computes this}} \quad \text{and} \quad \underbrace{B' \equiv A^b \pmod{p}}_{\text{Bob computes this}}.$$

The values that they compute,  $A'$  and  $B'$  respectively, are actually the same, since

$$A' \equiv B^a \equiv (g^b)^a \equiv g^{ab} \equiv (g^a)^b \equiv A^b \equiv B' \pmod{p}.$$

This common value is their exchanged key. The Diffie-Hellman key exchange algorithm is summarized in Table

Public Parameter Creation	
A trusted party chooses and publishes a (large) prime $p$ and an integer $g$ having large prime order in $F^*$ .	
Private Computations	
Alice	Bob
Choose a secret integer. Compute $A \equiv g^a \pmod{p}$ .	Choose a secret integer. Compute $B \equiv g^b \pmod{p}$ .
Public Exchange of Values	
Alice sends $A$ to Bob $\xrightarrow{\hspace{1cm}}$ $AB \xleftarrow{\hspace{1cm}}$ Bob sends $B$ to Alice	
Further Private Computations	
Alice	Bob
Compute the number $B^a \pmod{p}$ . The shared secret value is $B^a \equiv (g^b)^a \equiv g^{ab} \equiv (g^a)^b \equiv A^b \pmod{p}$ .	Compute the number $A^b \pmod{p}$ .

*Table 1. Diffie-Hellman Key Exchange*

## 6. Procedure/ Program:

- i. Firstly, Alice and Bob agree on two large prim numbers,  $n$  and  $g$ . These two integers need not be kept secret. Alice and Bob can use an insecure channel to agree on them.
- ii. Alice chooses another large random number  $x$ , and calculates  $A$  such that:
$$A = g^x \bmod n$$
- iii. Alice sends the number  $A$  to Bob.
- iv. Bob independently chooses another large random integer  $y$  and calculates  $B$  such that:
$$B = g^y \bmod n$$
- v. Bob sends the number  $B$  to Alice.
- vi. A now computes the secret key  $K_1$  as follows:
$$K_1 = B^x \bmod n$$
- vii. B now computes the secret key  $K_2$  as follows:
$$K_2 = A^y \bmod n$$

## 7. Conclusion:

The Diffie-Hellman key exchange algorithm is used to make secure channel to share secret key between sender and receiver. But man in middle attack is possible on this algorithm as values of  $n$  and  $g$  are publically known.

## 8. Viva Questions:

- Is there any particular reason to use Diffie-Hellman over RSA for key exchange?
- Explain the Diffie-Hellman shared key exchange mechanism?

## 9. References:

1. William Stallings, “*Cryptography and Network Security: Principles and Practice*”, Pearson education, Fifth edition.
2. Bernard Menezes, “*Network Security and Cryptography*”, Cengage Learning, Second Edition.
3. Behrouz A Forouzan, Debdeep Mukhopadhyay, “*Cryptography and Network Security*”, Tata McGraw Hill, Second edition
4. Behrouz A. Forouzan, “*Cryptography and Network Security*”, Tata McGraw Hill.
5. Charles P. Pfleeger, “*Security in Computing*”, Pearson Education.

# **Cryptography and System Security Lab**

## **Experiment No.: 4**

**For varying message sizes, test integrity of message using MD-5, SHA-1, and analyze the performance of the two protocols. Use crypt APIs**

## Experiment No. 4

1. **Aim:** For varying message sizes, test integrity of message using MD-5, SHA-1, and analyse the performance of the two protocols. Use crypt APIs

2. **Objectives:**

- To understand the applications of cryptographic hash functions.
- To distinguish between MD5 & SHA-1.
- To differentiate between hashing and encryption.

3. **Outcomes:** The learner will be able to

Apply security techniques and technologies to solve real-life security problems in practical systems.

4. **Hardware / Software Required:** C/C++/JAVA.

5. **Theory:**

MD5 (Message Digest algorithm 5) is a widely used cryptographic hash function with a 128-bit hash value. An MD5 hash is typically expressed as a 32-digit hexadecimal number. MD5 processes a variable length message into a fixed length output of 128 bits. The input message is broken up into chunks of 512-bit blocks (sixteen 32bit little endian integers) ; The message is padded so that its length is divisible by 512. The padding works as follows: first a single bit, 1, is appended to the end of the message. This is followed by as many zeros as are required to bring the length of the message up to 64 bits less than a multiple of 512. The remaining bits are filled up with a 64bit integer representing the length of the original message, in bits.

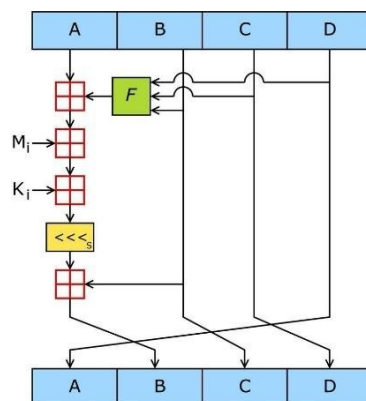


Figure 1: One MD5 operation. MD5 consists of 64 of these operations, grouped in four rounds of 16 operations.  $F$  is a nonlinear function; one function is used in each round.  $M_i$

denotes a 32bit block of the message input, and  $K_i$  denotes a 32bit constant, different for each operation.

The main MD5 algorithm operates on a 128bit state, divided into four 32bit words, denoted  $A, B, C$  and  $D$ . These are initialized to certain fixed constants. The main algorithm then operates on each 512bit message block in turn, each block modifying the state. The processing of a message block consists of four similar stages, termed *rounds*; each round is composed of 16 similar operations based on a nonlinear function  $F$ , modular addition, and leftrotation.

Figure 1 illustrates one operation within a round. There are four possible functions  $F$ ; a different one is used in each round:

$$F(X, Y, Z) = (X \wedge Y) \vee (\neg X \wedge Z)$$

$$G(X, Y, Z) = (X \wedge Z) \vee (Y \wedge \neg Z)$$

$$I(X, Y, Z) = Y \oplus (X \vee \neg Z)$$

$\oplus, \wedge, \vee, \neg$  denote the XOR, AND, OR and NOT operations respectively.

## 6. Procedure/Algorithm:

### 1. Append Padding Bits

The message is "padded" (extended) so that its length (in bits) is congruent to 448, modulo 512. That is, the message is extended so that it is just 64 bits shy of being a multiple of 512 bits long. Padding is always performed, even if the length of the message is already congruent to 448, modulo 512. Padding is performed as follows: a single "1" bit is appended to the message, and then "0" bits are appended so that the length in bits of the padded message becomes congruent to 448, modulo 512. In all, at least one bit and at most 512 bits are appended.

### 2. Append Length

A 64-bit representation of  $b$  (the length of the message before the padding bits were added) is appended to the result of the previous step. In the unlikely event that  $b$  is greater than  $2^{64}$ , then only the low order 64 bits of  $b$  are used. (These bits are appended as two 32bit words and appended low order word first in accordance with the previous conventions.) At this point the resulting message (after padding with bits and with  $b$ ) has a length that is an exact multiple of 512 bits. Equivalently, this message has a length that is an exact multiple of 16 (32 bit) words. Let  $M[0 \dots N-1]$  denote the words of the resulting message, where  $N$  is a multiple of 16.

### **3. Initialize MD Buffer**

A fourword buffer (A,B,C,D) is used to compute the message digest. Here each of A, B, C, D is a 32bit register. These registers are initialized to the following values in hexadecimal, loworder bytes first):

### **4. Process Message in 16Word Blocks**

We first define four auxiliary functions that each take as input three 32bit words and produce as output one 32bit word.

### **7. Conclusion:**

The main aim of message digest algorithm is to ensure integrity of message. The strength of MD5 algorithm lies in the chaining function, because of which integrity of message cannot be compromised

### **8. Viva Questions:**

- What is the MD5 algorithm? How it is works?
- What is SHA1 algorithm?

### **9. References:**

1. William Stallings, “*Cryptography and Network Security: Principles and Practice*”, Pearson education, Fifth edition.
2. Bernard Menezes, “*Network Security and Cryptography*”, Cengage Learning, Second Edition.
3. Behrouz A Forouzan, Debdeep Mukhopadhyay, “*Cryptography and Network Security*”, Tata McGraw Hill, Second edition
4. Behrouz A. Forouzan, “*Cryptography and Network Security*”, Tata McGraw Hill.
5. Charles P. Pfleeger, “*Security in Computing*”, Pearson Education.



# **Cryptography and System Security Lab**

## **Experiment No.: 5**

**Study and use network reconnaissance tools like WHOIS, dig, traceroute, nslookup, nmap to gather information about networks and domain registrars.**

# Experiment No. 5

1. **Aim:** Eve has just started learning security concepts. She is very curious to get basic information about networks. Show how WHOIS, dig, traceroute, nslookup utilities will help her to gather required information about networks.

2. **Objectives:**

- To understand network information discovery.
- To study various basic network commands to gather network information.
- To understand passive attack technique.

3. **Outcomes:**

The learner will be able to Apply basic network command to gather basic network information

4. **Hardware / Software Required:** Unix/Linux

5. **Theory:**

1. **WHOIS :** WHOIS is the Linux utility for searching an object in a WHOIS database. The WHOIS database of a domain is the publicly displayed information about a domains ownership, billing, technical, administrative, and nameserver information. Running a WHOIS on your domain will look the domain up at the registrar for the domain information. All domains have WHOIS information. WHOIS database can be queried to obtain the following information via WHOIS:

- Administrative contact details, including names, email addresses, and telephone numbers
- Mailing addresses for office locations relating to the target organization
- Details of authoritative name servers for each given domain

**Example: Querying Facebook.com**

**ssc@ssc-OptiPlex-380:~\$ whois facebook.com**

Whois Server Version 2.0

Domain names in the .com and .net domains can now be registered with many different competing registrars. Go to <http://www.internic.net>

for detailed information.

Server Name: FACEBOOK.COM.BRETLANDTRUSTMERCHANDISINGDEPART.COM

IP Address: 69.63.176.11

Registrar: GOOGLE INC.

Whois Server: whois.rrpproxy.net

Referral URL: <http://domains.google.com>

Server Name:

FACEBOOK.COM.DISABLE.YOUR.TIMELINE.NOW.WITH.THE.ORIGINAL.TIMELIN  
E-REMOVE.NET

IP Address: 8.8.8.8

Registrar: ENOM, INC.

Whois Server: whois.enom.com

Referral URL: <http://www.enom.com>

Server Name:

FACEBOOK.COM.GET.ONE.MILLION.DOLLARS.AT.WWW.UNIMUNDI.COM

IP Address: 209.126.190.70

Registrar: PDR LTD. D/B/A PUBLICDOMAINREGISTRY.COM

Whois Server: whois.PublicDomainRegistry.com

Referral URL: <http://www.PublicDomainRegistry.com>

Server Name: FACEBOOK.COM.LOVED.BY.WWW.SHQIPHOST.COM

IP Address: 46.4.210.254

Registrar: ONLINENIC, INC.

Whois Server: whois.onlinenic.com

Referral URL: <http://www.OnlineNIC.com>

Server Name: FACEBOOK.COM.MORE.INFO.AT.WWW.BEYONDWHOIS.COM

IP Address: 203.36.226.2

Registrar: INSTRA CORPORATION PTY, LTD.

Whois Server: whois.instra.net

Referral URL: <http://www.instra.com>

Server Name:

FACEBOOK.COM.ZZZZZ.GET.LAID.AT.WWW.SWINGINGCOMMUNITY.COM

IP Address: 69.41.185.229

Registrar: TUCOWS DOMAINS INC.

Whois Server: whois.tucows.com

Referral URL: <http://www.tucowsdomains.com>

Domain Name: FACEBOOK.COM

Registrar: MARKMONITOR INC.

Sponsoring Registrar IANA ID: 292

Whois Server: whois.markmonitor.com

Referral URL: <http://www.markmonitor.com>

Name Server: A.NS.FACEBOOK.COM

Name Server: B.NS.FACEBOOK.COM

Status: clientDeleteProhibited <http://www.icann.org/epp#clientDeleteProhibited>

Status: clientTransferProhibited <http://www.icann.org/epp#clientTransferProhibited>

Status: clientUpdateProhibited <http://www.icann.org/epp#clientUpdateProhibited>

Status: serverDeleteProhibited

<http://www.icann.org/epp#serverDeleteProhibited>  
Status:serverTransferProhibited <http://www.icann.org/epp#serverTransferProhibited>  
Status:serverUpdateProhibited <http://www.icann.org/epp#serverUpdateProhibited>  
Updated Date: 28-sep-2012  
Creation Date: 29-mar-1997  
Expiration Date: 30-mar-2020  
>>> Last update of whois database: Fri, 17 Jul 2015 04:12:12 GMT <<<  
The Registry database contains ONLY .COM, .NET, .EDU domains and Registrars.  
For more information on Whois status codes, please visit  
<https://www.icann.org/resources/pages/epp-status-codes-2014-06-16-en>.  
Domain Name: facebook.com  
Registry Domain ID: 2320948 DOMAIN\_COM-VRSN  
Registrar WHOIS Server: whois.markmonitor.com  
Registrar URL: <http://www.markmonitor.com>  
Updated Date: 2014-10-28T12:38:28-0700  
Creation Date: 1997-03-28T21:00:00-0800  
Registrar Registration Expiration Date: 2020-03-29T21:00:00-0700  
Registrar: MarkMonitor, Inc.  
Registrar IANA ID: 292  
Registrar Abuse Contact Email: [abusecomplaints@markmonitor.com](mailto:abusecomplaints@markmonitor.com)  
Registrar Abuse Contact Phone: +1.2083895740

Domain Status: clientUpdateProhibited (<https://www.icann.org/epp#clientUpdateProhibited>)  
Domain Status: clientTransferProhibited (<https://www.icann.org/epp#clientTransferProhibited>)  
Domain Status: clientDeleteProhibited (<https://www.icann.org/epp#clientDeleteProhibited>)  
Registry Registrant ID:  
Registrant Name: Domain Administrator  
Registrant Organization: Facebook, Inc.  
Registrant Street: 1601 Willow Road,  
Registrant City: Menlo Park  
Registrant State/Province: CA  
Registrant Postal Code: 94025  
Registrant Country: US  
Registrant Phone: +1.6505434800  
Registrant Phone Ext:  
Registrant Fax: +1.6505434800  
Registrant Fax Ext:  
Registrant Email: [domain@fb.com](mailto:domain@fb.com)  
Registry Admin ID:  
Admin Name: Domain Administrator  
Admin Organization: Facebook, Inc.  
Admin Street: 1601 Willow Road,  
Admin City: Menlo Park  
Admin State/Province: CA  
Admin Postal Code: 94025

Admin Country: US  
Admin Phone: +1.6505434800  
Admin Phone Ext:  
Admin Fax: +1.6505434800  
Admin Fax Ext:  
Admin Email: domain@fb.com  
Registry Tech ID:  
Tech Name: Domain Administrator  
Tech Organization: Facebook, Inc.  
Tech Street: 1601 Willow Road,  
Tech City: Menlo Park  
Tech State/Province: CA  
Tech Postal Code: 94025  
Tech Country: US  
Tech Phone: +1.6505434800  
Tech Phone Ext:  
Tech Fax: +1.6505434800  
Tech Fax Ext:  
Tech Email: domain@fb.com  
Name Server: b.ns.facebook.com  
Name Server: a.ns.facebook.com  
DNSSEC: unsigned  
URL of the ICANN WHOIS Data Problem Reporting System: <http://wdprs.internic.net/>  
>>> Last update of WHOIS database: 2015-07-16T21:08:30-0700 <<<

The Data in MarkMonitor.com's WHOIS database is provided by MarkMonitor.com for information purposes, and to assist persons in obtaining information about or related to a domain name registration record. MarkMonitor.com does not guarantee its accuracy. By submitting a WHOIS query, you agree that you will use this Data only for lawful purposes and that, under no circumstances will you use this Data to:

- (1) allow, enable, or otherwise support the transmission of mass unsolicited, commercial advertising or solicitations via e-mail (spam); or
- (2) enable high volume, automated, electronic processes that apply to MarkMonitor.com (or its systems).

MarkMonitor.com reserves the right to modify these terms at any time.

By submitting this query, you agree to abide by this policy.

MarkMonitor is the Global Leader in Online Brand Protection.

MarkMonitor Domain Management(TM)

MarkMonitor Brand Protection(TM)

MarkMonitor AntiPiracy(TM)

MarkMonitor AntiFraud(TM)

Professional and Managed Services

Visit MarkMonitor at <http://www.markmonitor.com>

Contact us at +1.8007459229

In Europe, at +44.02032062220

ssc@ssc-OptiPlex-380:~\$

2. **Dig** - Dig is a networking tool that can query DNS servers for information. It can be very helpful for diagnosing problems with domain pointing and is a good way to verify that your configuration is working. The most basic way to use dig is to specify the domain we wish to query:

**Example:**

**\$ dig duckduckgo.com**

```
; <<>> DiG 9.8.1-P1 <<>> duckduckgo.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 64399
;; flags: qr rd ra; QUERY: 1, ANSWER: 4, AUTHORITY: 0, ADDITIONAL: 0
;; QUESTION SECTION:
;duckduckgo.com. IN A
;; ANSWER SECTION:
duckduckgo.com. 99 IN A 107.21.1.61
duckduckgo.com. 99 IN A 184.72.106.253
duckduckgo.com. 99 IN A 184.72.106.52
duckduckgo.com. 99 IN A 184.72.115.86
;; Query time: 33 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Fri Aug 23 14:26:17 2013
;; MSG SIZE rcvd: 96
```

The lines above act as a header for the query performed. It is possible to run dig in batch mode,

so proper labeling of the output is essential to allow for correct analysis.

```
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 64399
```

```
:: flags: qr rd ra; QUERY: 1, ANSWER: 4, AUTHORITY: 0, ADDITIONAL: 0
```

The next section gives us a technical summary of our query results. We can see that the query was successful, certain flags were used, and that 4 "answers" were received.

```
:: QUESTION SECTION:
```

```
;duckduckgo.com. IN A
```

```
:: ANSWER SECTION:
```

```
duckduckgo.com. 99 IN A 107.21.1.61
```

```
duckduckgo.com. 99 IN A 184.72.106.253
```

```
duckduckgo.com. 99 IN A 184.72.106.52
```

```
duckduckgo.com. 99 IN A 184.72.115.86
```

The above section of the output contains the actual results we were looking for. It restates the query and then returns the matching DNS records for that domain name.

Here, we can see that there are four "A" records for "duckduckgo.com". By default, "A" records

are returned. This gives us the IP addresses that the domain name resolves to.

The "99" is the TTL (time to live) before the DNS server rechecks the association between the

domain name and the IP address. The "IN" means the class of the record is a standard internet class.

```
:: Query time: 33 msec
```

```
:: SERVER: 8.8.8.8#53(8.8.8.8)
```

```
:: WHEN: Fri Aug 23 14:26:17 2013
```

```
:: MSG SIZE rcvd: 96
```

These lines simply provide some statistics about the actual query results. The query time can be

indicative of problems with the DNS servers.

- 3. Traceroute** - traceroute prints the route that packets take to a network host. Traceroute utility uses the TTL field in the IP header to achieve its operation. For users who are new to TTL field, this field describes how much hops a particular packet will take while traveling on network. So, this effectively outlines the lifetime of the packet on network. This field is usually set to 32 or 64. Each time the packet is held on an intermediate router, it decreases the TTL value by 1. When a router finds the TTL value of 1 in a received packet then that packet is not forwarded but instead discarded. After discarding the packet, router sends an ICMP error message of —Time exceeded back to the source from where packet generated. The ICMP packet that is sent back contains the IP address of the router. So now it can be easily understood that traceroute operates by sending packets with TTL value starting from 1 and then incrementing by one each time. Each time a router receives the packet, it checks the TTL field, if TTL field is 1 then it discards the packet and sends the ICMP error packet containing its IP address and this is what traceroute requires. So traceroute incrementally fetches the IP of all the routers between the source and the destination.

**Example:**

**Traceroute example.com**

traceroute to example.com (64.13.192.208), 64 hops max, 40 byte packets

1 72.10.62.1 (72.10.62.1) 1.000 ms 0.739 ms 0.702 ms

2 10.101.248.1 (10.101.248.1) 0.683 ms 0.385 ms 0.315 ms

3 10.104.65.161 (10.104.65.161) 0.791 ms 0.703 ms 0.686 ms

4 10.104.65.161 (10.104.65.161) 0.791 ms 0.703 ms 0.686 ms

5 10.0.10.33 (10.0.10.33) 2.652 ms 2.260 ms 5.353 ms

6 acmkokeaig.gs01.gridserver.com (64.13.192.208) 3.384 ms 8.001 ms 2.439 ms

- 4. Nslookup** - The nslookup command is used to query internet name servers interactively for information. nslookup, which stands for "name server lookup", is a useful tool for finding out information about a named domain. By default, nslookup will translate a domain name to an IP address (or vice versa). For instance, to find out what the IP address of microsoft.com is, you could run the command:



**Example:****\$nslookup microsoft.com**

Server: 8.8.8.8

Address: 8.8.8.8#53

Non-authoritative answer:

Name: microsoft.com

Address: 134.170.185.46

Name: microsoft.com

Address: 134.170.188.221

Here, 8.8.8.8 is the address of our system's Domain Name Server. This is the server our system is configured to use to translate domain names into IP addresses. "#53" indicates that we are communicating with it on port 53, which is the standard port number domain name servers use to accept queries. Below this, we have our lookup information for microsoft.com. Our name server returned two entries, 134.170.185.46 and 134.170.188.221. This indicates that microsoft.com uses a round robin setup to distribute server load. When you access microsoft.com, you may be directed to either of these servers and your packets will be routed to the correct destination. You can see that we have received a "Non-authoritative answer" to our query. An answer is "authoritative" only if our DNS has the complete zone file information for the domain in question. More often, our DNS will have a cache of information representing the last authoritative answer it received when it made a similar query, this information is passed on to you, but the server qualifies it as "non-authoritative": the information was recently received from an authoritative source, but the DNS server is not itself that authority.

**5. nmap**

Nmap (Network Mapper) is a security scanner originally written by Gordon Lyon (also known by his pseudonym Fyodor Vaskovich) used to discover hosts and services on a computer network, thus creating a "map" of the network. To accomplish its goal, Nmap sends specially crafted packets to the target host and then analyzes the responses. Unlike many simple port scanners that just send packets at some predefined constant rate, Nmap accounts for the network conditions (latency fluctuations, network congestion, the target interference with the scan) during the run. Also, owing to the large and active user community providing feedback and contributing to its features, Nmap has been able to extend its discovery

capabilities beyond simply figuring out whether a host is up or down and which ports are open and closed; it can determine the operating system of the target, names and versions of the listening services, estimated uptime, type of device, and presence of a firewall.

**Nmap features include:**

- Host Discovery – Identifying hosts on a network. For example, listing the hosts which respond to pings or have a particular port open.
- Port Scanning – Enumerating the open ports on one or more target hosts.
- Version Detection – Interrogating listening network services listening on remote devices to determine the application name and version number.
- OS Detection – Remotely determining the operating system and some hardware characteristics of network devices.

**Basic commands working in Nmap:**

- For target specifications: `nmap<target's URL or IP with spaces between them>`
- For OS detection: `nmap -O <target-host's URL or IP>`
- For version detection: `nmap -sV<target-host's URL or IP>`

SYN scan is the default and most popular scan option for good reasons. It can be performed quickly, scanning thousands of ports per second on a fast network not hampered by restrictive firewalls. It is also relatively unobtrusive and stealthy since it never completes TCP connections

**Algorithm\Implementation Steps\Installation Steps:**

- Installing Nmap from the link.  
`sudo apt-get install nmap`
- Obtaining Your IP addresses.  
Use the `ifconfig` command in Linux.
- Performing a Scan of the Local Network.
  1. For the following steps, please use the nmap command line tool installed on Ubuntu
  2. Scan your subnet to determine how many hosts can be found. For example, if you are on the 192.168.1.0 subnet, you would enter the following command: `nmap -sP 192.168.1.*`
    - i. What is your subnet? \_\_\_\_\_
    - ii. How many hosts were found? \_\_\_\_\_

3. Next perform a stealth scan (Please use the IP for your subnet): `nmap -sS -P0 -p 192.169.1.*`
4. Now, you'll perform an OS identification. Use the Linux O/S to scan your Windows machine:
  - i. `nmap -O Windows_IP_ADDRESS`
  - ii. OS Type 1: \_\_\_\_\_
  - iii. Now we want to use the Windows machine to scan the Linux O/S. Go to a Windows DOS prompt and enter the following command:
  - iv. `nmap -O Linux_IP_ADDRESS`
  - v. Now we will perform a service selection scan. Let's scan for all computers with FTP running. We would do that as follows:

`nmap -p21 192.168.1.*`

5. List the IP addresses with that has the FTP open: \_\_\_\_\_

### **Input and Output:**

- Installation of nmap:

`>sudo apt-get install nmap`

- `nmap -sP 10.0.0.0/24`

Ping scans the network, listing machines that respond to ping.

- FIN scan (`-sF`)

Sets just the TCP FIN bit.

- `-sV` (Version detection) .

Enables version detection, as discussed above. Alternatively, can use `-A`, which enables

version detection among other things.

- `-sO` (IP protocol scan) .

IP protocol scan allows you to determine which IP protocols (TCP, ICMP, IGMP, etc.) are supported by target machines. This isn't technically a port scan, since it cycles through IP

protocol numbers rather than TCP or UDP port numbers.

- -O (Enable OS detection) .

Enables OS detection, as discussed above. Alternatively, you can use -A to enable OS detection along with other things.

- -p port ranges (Only scan specified ports) .

This option specifies which ports you want to scan and overrides the default.

#### Individual port

numbers are OK, as are ranges separated by a hyphen (e.g. 1-1023). The beginning and/or

end values of a range may be omitted, causing Nmap to use 1 and 65535, respectively.

- --top-ports <integer of 1 or greater>

Scans the N highest-ratio ports found in nmap-services file.

- nmap -iflist

host interface and route information with nmap by using --iflist option.

#### 6. Conclusion:

Various reconnaissance tools are studied and used to gather primary network information.

#### 7. Viva Questions:

- What is the use of whois command?
- What kind of information is gathered using traceroute command?
- What is the use of nslookup command?

#### 8. References:

1. William Stallings, “*Cryptography and Network Security: Principles and Practice*”, Pearson education, Fifth edition.
2. Bernard Menezes, “*Network Security and Cryptography*”, Cengage Learning, Second Edition.
3. Behrouz A Forouzan, Debdeep Mukhopadhyay, “*Cryptography and Network Security*”, Tata McGraw Hill, Second edition
4. Behrouz A. Forouzan, “*Cryptography and Network Security*”, Tata McGraw Hill.
5. Charles P. Pfleeger, “*Security in Computing*”, Pearson Education.

# **Cryptography and System Security Lab**

## **Experiment No.: 6**

**Study and install packet sniffer tool (Wireshark) to capture ICMP, TCP and HTTP packets in promiscuous mode based on different filters.**

# Experiment No. 6

1. **Aim:** Tom wants to launch an attack on one of the hosts in his network. He wants to monitor in-bound and out-bound traffic of that host to find some vulnerabilities. How Wireshark will help him for the same.

2. **Objectives:**

- Understand the need for traffic analysis.
- Understand the how packet sniffing is done using wireshark.
- Trace and understand various packets from dynamic traffic.

3. **Outcomes:**

The learner will be able to

- Sniff network packets and study insights of packets to get detail network information.

4. **Hardware / Software Required:** Unix/Linux/Windows, wireshark

5. **Theory:**

Wireshark, a network analysis tool formerly known as Ethereal, captures packets in real time and display them in human-readable format. Wireshark includes filters, color-coding and other features that let you dig deep into network traffic and inspect individual packets.

Features of Wireshark:

- Available for UNIX and Windows.
- Capture live packet data from a network interface.
- Open files containing packet data captured with tcpdump/WinDump, Wireshark, and a number of other packet capture programs.
- Import packets from text files containing hex dumps of packet data.
- Display packets with very detailed protocol information.
- Export some or all packets in a number of capture file formats.
- Filter packets on many criteria.
- Search for packets on many criteria.
- Colorize packet display based on filters.
- Create various statistics.

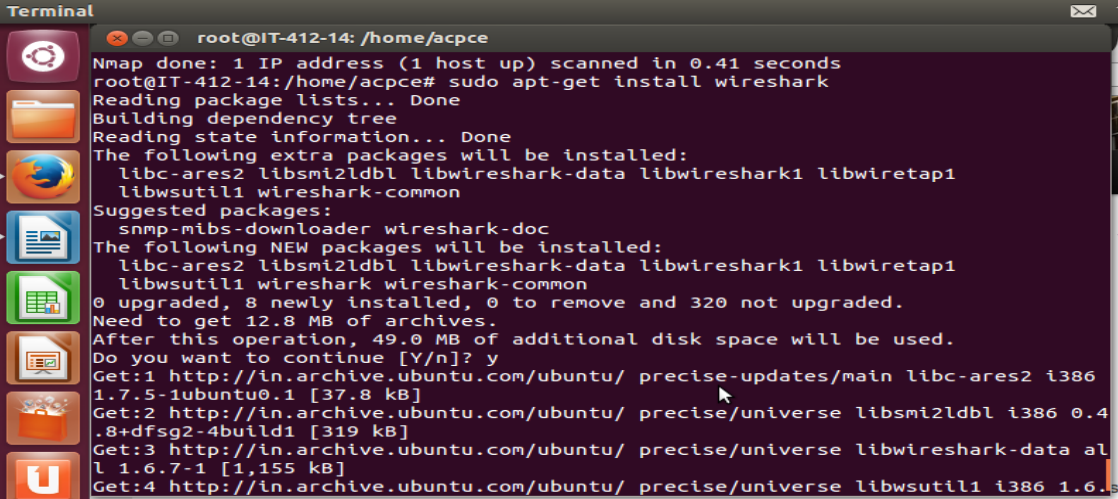
## Capturing Packets

After downloading and installing wireshark, you can launch it and click the name of an interface under Interface List to start capturing packets on that interface. For example, if you

want to capture traffic on the wireless network, click your wireless interface. You can configure advanced features by clicking Capture Options.

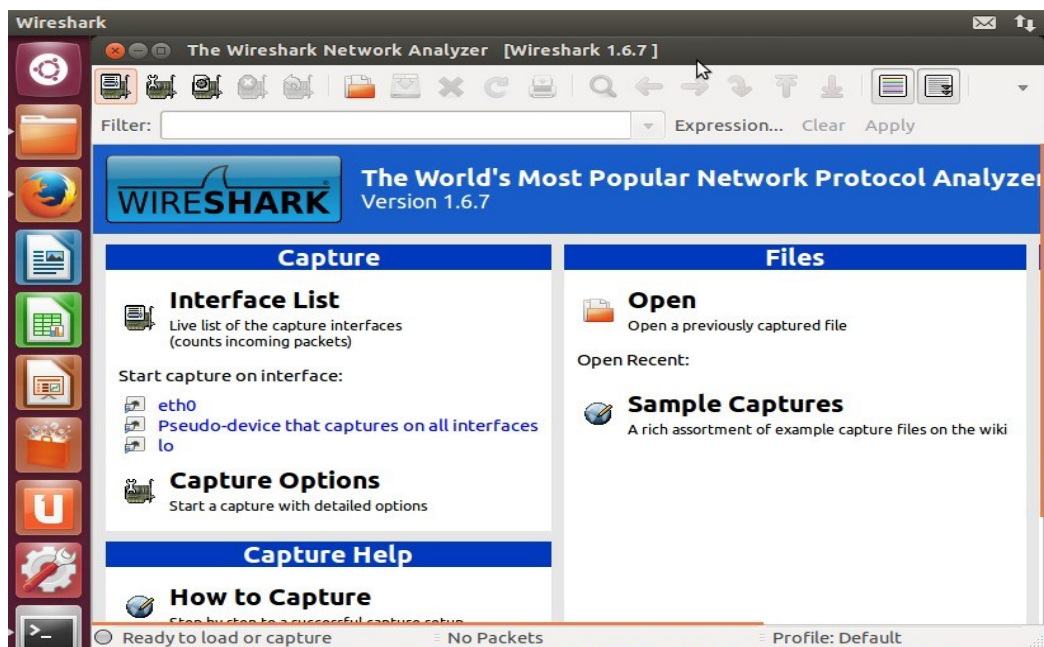
### Installation of Wireshark:

sudo apt-get install wireshark



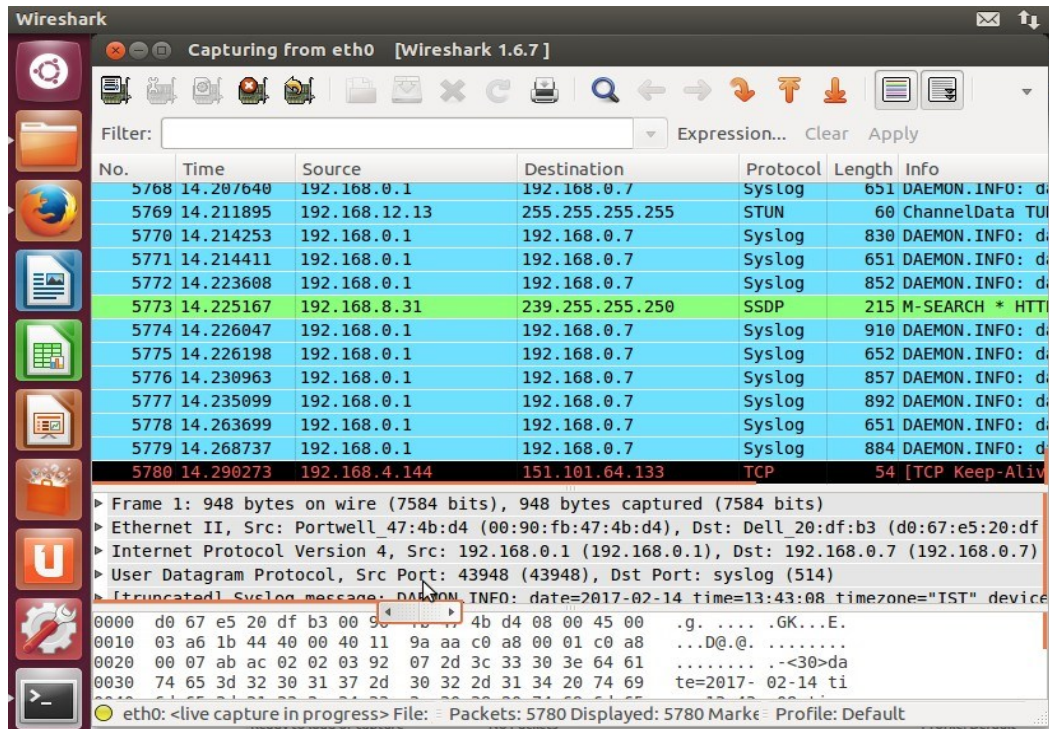
```
root@IT-412-14: /home/acpce
Nmap done: 1 IP address (1 host up) scanned in 0.41 seconds
root@IT-412-14:/home/acpce# sudo apt-get install wireshark
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  libc-ares2 libsmi2ldbl libwireshark-data libwireshark1 libwiretap1
  libwsutil1 wireshark-common
Suggested packages:
  snmp-mibs-downloader wireshark-doc
The following NEW packages will be installed:
  libc-ares2 libsmi2ldbl libwireshark-data libwireshark1 libwiretap1
  libwsutil1 wireshark wireshark-common
0 upgraded, 8 newly installed, 0 to remove and 320 not upgraded.
Need to get 12.8 MB of archives.
After this operation, 49.0 MB of additional disk space will be used.
Do you want to continue [Y/n]? y
Get:1 http://in.archive.ubuntu.com/ubuntu/ precise-updates/main libc-ares2 i386
1.7.5-1ubuntu0.1 [37.8 kB]
Get:2 http://in.archive.ubuntu.com/ubuntu/ precise/universe libsmi2ldbl i386 0.4
.8+dfsg2-4build1 [319 kB]
Get:3 http://in.archive.ubuntu.com/ubuntu/ precise/universe libwireshark-data al
l 1.6.7-1 [1,155 kB]
Get:4 http://in.archive.ubuntu.com/ubuntu/ precise/universe libwsutil1 i386 1.6.7-1 [1,155 kB]
```

After downloading and installing wireshark, you can launch it and click the name of an interface under Interface List to start capturing packets on that interface. For example, if you want to capture traffic on the wireless network, click your wireless interface. You can configure advanced features by clicking Capture Options.

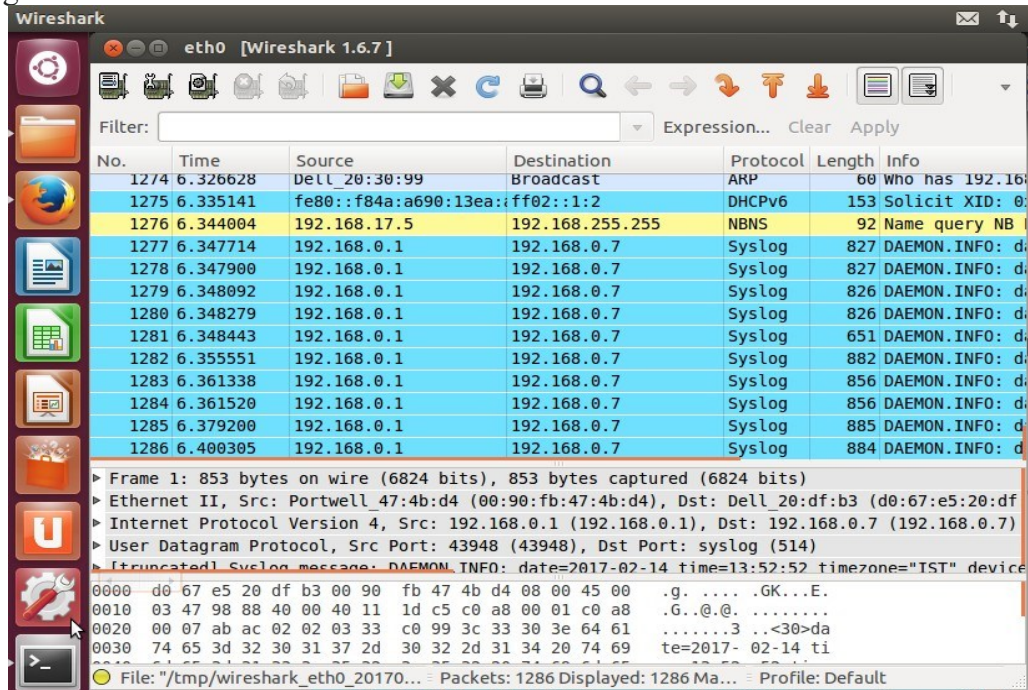




As soon as you click the interface's name, you'll see the packets start to appear in real time. Wireshark captures each packet sent to or from your system. If you're capturing on a wireless interface and have promiscuous mode enabled in your capture options, you'll also see other the other packets on the network

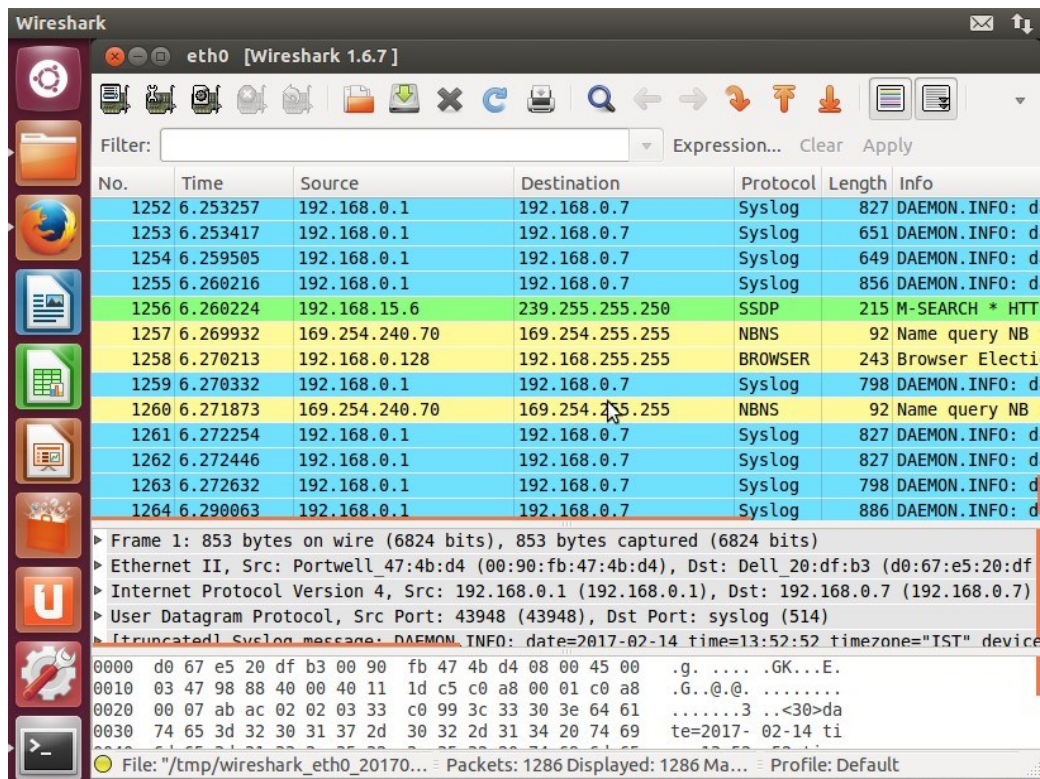


Click the stop capture button near the top left corner of the window when you want to stop capturing traffic.





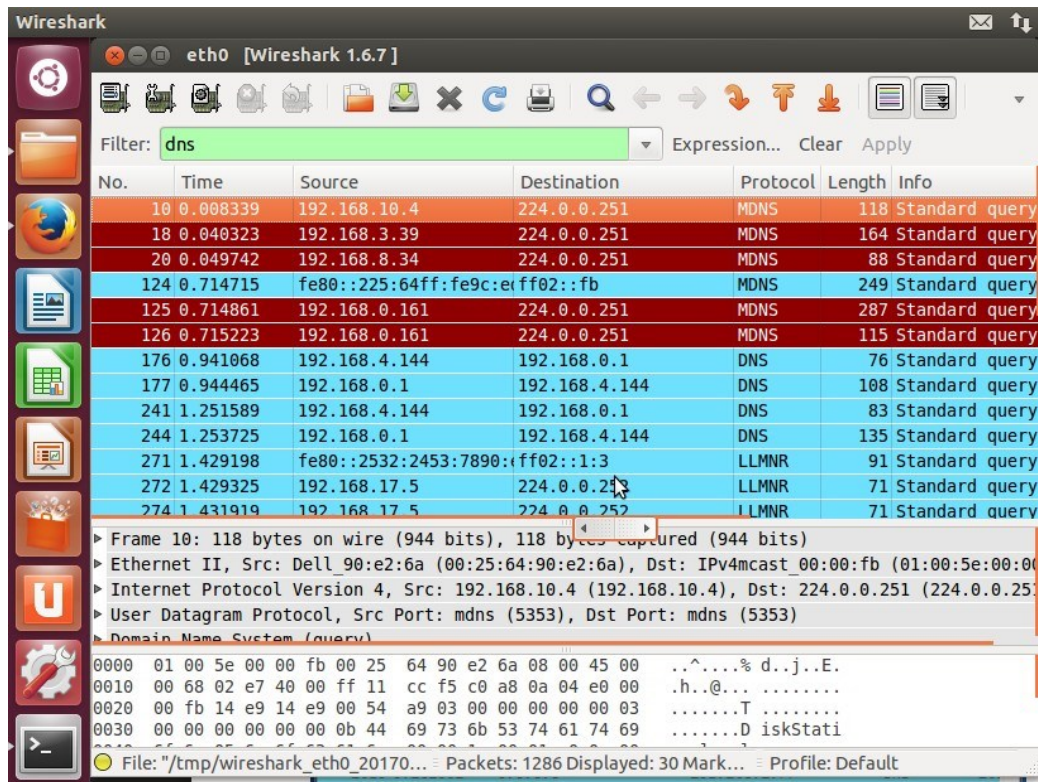
Wireshark uses colors to help you identify the types of traffic at a glance. By default, green is TCP traffic, dark blue is DNS traffic, light blue is UDP traffic, and black identifies TCP packets with problems — for example, they could have been delivered out-of-order.



## Filtering Packets

If you're trying to inspect something specific, such as the traffic a program sends when phoning home, it helps to close down all other applications using the network so you can narrow down the traffic. Still, you'll likely have a large amount of packets to sift through. That's where Wireshark's filters come in.

The most basic way to apply a filter is by typing it into the filter box at the top of the window and clicking Apply (or pressing Enter). For example, type `—dns` and you'll see only DNS packets. When you start typing, Wireshark will help you autocomplete your filter.



## 6. Conclusion:

Wireshark installation and network traffic analysis using packet sniffing is done. Detailed information about packets is explored by applying filters.

## 7. Viva Questions:

- What kind of information is gathered from packets?
- How to see List of all IP and MAC addresses of all machines active on network?

## 8. References:

1. William Stallings, “*Cryptography and Network Security: Principles and Practice*”, Pearson education, Fifth edition.
2. Bernard Menezes, “*Network Security and Cryptography*”, Cengage Learning, Second Edition.
3. Behrouz A Forouzan, Debdeep Mukhopadhyay, “*Cryptography and Network Security*”, Tata McGraw Hill, Second edition
4. Behrouz A. Forouzan, “*Cryptography and Network Security*”, Tata McGraw Hill.
5. Charles P. Pfleeger, “*Security in Computing*”, Pearson Education.

# **Cryptography and System Security Lab**

## **Experiment No.: 7**

**Download and install nmap. Use it with different options to scan open ports, perform OS fingerprinting, do a ping scan, tcp port scan, udp port scan, xmas scan etc.**

## Experiment No. 7

**6. Aim:** John have recently used some reconnaissance Linux commands and obtained IP address of one of the servers. He wants to gather more information about that server. Show how Nmap will help him to gather comprehensive network related and system related information.

**7. Objectives:**

- To understand network information discovery.
- To study network mapper to gather system and network information.
- To understand passive attack technique.

**8. Outcomes:**

The learner will be able to apply basic Linux utility to gather comprehensive system and network information

**9. Hardware / Software Required:** Unix/Linux

**10. Theory:**

### **Nmap**

Nmap (Network Mapper) is a security scanner originally written by Gordon Lyon (also known by his pseudonym Fyodor Vaskovich) used to discover hosts and services on a computer network, thus creating a "map" of the network. To accomplish its goal, Nmap sends specially crafted packets to the target host and then analyzes the responses. Unlike many simple port scanners that just send packets at some predefined constant rate, Nmap accounts for the network conditions (latency fluctuations, network congestion, the target interference with the scan) during the run. Also, owing to the large and active user community providing feedback and contributing to its features, Nmap has been able to extend its discovery

capabilities beyond simply figuring out whether a host is up or down and which ports are open and closed; it can determine the operating system of the target, names and versions of the listening services, estimated uptime, type of device, and presence of a firewall.

**Nmap features include:**

- Host Discovery – Identifying hosts on a network. For example, listing the hosts which respond to pings or have a particular port open.
- Port Scanning – Enumerating the open ports on one or more target hosts.
- Version Detection – Interrogating listening network services listening on remote devices to determine the application name and version number.
- OS Detection – Remotely determining the operating system and some hardware characteristics of network devices.

**Basic commands working in Nmap:**

- For target specifications: `nmap<target's URL or IP with spaces between them>`
- For OS detection: `nmap -O <target-host's URL or IP>`
- For version detection: `nmap -sV<target-host's URL or IP>`

SYN scan is the default and most popular scan option for good reasons. It can be performed quickly, scanning thousands of ports per second on a fast network not hampered by restrictive firewalls. It is also relatively unobtrusive and stealthy since it never completes TCP connections

**Algorithm\Implementation Steps\Installation Steps:**

- Installing Nmap from the link.  
`sudo apt-get install nmap`
- Obtaining Your IP addresses.  
Use the `ifconfig` command in Linux.
- Performing a Scan of the Local Network.

9. For the following steps, please use the nmap command line tool installed on Ubuntu

10. Scan your subnet to determine how many hosts can be found. For example, if you are on the 192.168.1.0 subnet, you would enter the following command: `nmap -sP 192.168.1.*`

- i. What is your subnet? \_\_\_\_\_
- ii. How many hosts were found? \_\_\_\_\_

11. Next perform a stealth scan (Please use the IP for your subnet): `nmap -sS -P0 -p 192.169.1.*`

12. Now, you'll perform an OS identification. Use the Linux O/S to scan your Windows machine:

- i. `nmap -O Windows_IP_ADDRESS`
- ii. OS Type 1: \_\_\_\_\_
- iii. Now we want to use the Windows machine to scan the Linux O/S. Go to a Windows DOS prompt and enter the following command:
- iv. `nmap -O Linux_IP_ADDRESS`
- v. Now we will perform a service selection scan. Let's scan for all computers with FTP running. We would do that as follows:

`nmap -p21 192.168.1.*`

13. List the IP addresses with that has the FTP open: \_\_\_\_\_

### **Input and Output:**

- Installation of nmap:  
`>sudo apt-get install nmap`
- `nmap -sP 10.0.0.0/24`  
Ping scans the network, listing machines that respond to ping.
- FIN scan (`-sF`)  
Sets just the TCP FIN bit.
- `-sV` (Version detection).  
Enables version detection, as discussed above. Alternatively, can use `-A`, which enables version detection among other things.
- `-sO` (IP protocol scan).  
IP protocol scan allows you to determine which IP protocols (TCP, ICMP, IGMP, etc.) are supported by target machines. This isn't technically a port scan, since it cycles through IP



protocol numbers rather than TCP or UDP port numbers.

- -O (Enable OS detection) .

Enables OS detection, as discussed above. Alternatively, you can use -A to enable OS detection along with other things.

- -p port ranges (Only scan specified ports) .

This option specifies which ports you want to scan and overrides the default Individual port numbers are OK, as are ranges separated by a hyphen (e.g. 1-1023). The beginning and/or end values of a range may be omitted, causing Nmap to use 1 and 65535, respectively.

- --top-ports <integer of 1 or greater>

Scans the N highest-ratio ports found in nmap-services file.

- nmap -iflist

host interface and route information with nmap by using --iflist option.

#### **14. Conclusion:**

Network mapper tool is studied and used to gather comprehensive system and network primary network information.

#### **15. Viva Questions:**

- What is the use of Nmap command?
- What kind of information is gathered using Nmap command?
- How to check operating system details on remote server?

#### **16. References:**

1. William Stallings, “*Cryptography and Network Security: Principles and Practice*”, Pearson education, Fifth edition.
2. Bernard Menezes, “*Network Security and Cryptography*”, Cengage Learning, Second Edition.
3. Behrouz A Forouzan, Debdeep Mukhopadhyay, “*Cryptography and Network Security*”, Tata McGraw Hill, Second edition
4. Behrouz A. Forouzan, “*Cryptography and Network Security*”, Tata McGraw Hill.
5. Charles P. Pfleeger, “*Security in Computing*”, Pearson Education

# **Cryptography and System Security Lab**

## **Experiment No.: 8**

### **Simulate DOS attack using Hping3 and Wireshark.**



# Experiment No. 8

1. **Aim:** Eve, an attacker wishes to make some services unavailable of a particular host on the network. How he will do this using Hping3 and Wireshark.
2. **Objectives:**
  - Understand the concept of DOS attacks.
  - Launch DOS attack using Hping3 and observe it using wireshark.
3. **Outcomes:** The learner will be able to  
Analyze DOS attack and its effect on the network using Hping3 and wireshark.  
practical systems.
4. **Hardware / Software Required:** Unix/Linux, Hping3, wireshark
5. **Theory:**

Denial-of-service (DoS) attack is an attempt to make a machine or network resource unavailable to its intended users, such as to temporarily or indefinitely interrupt or suspend services. A distributed denial-of-service (DDoS) is where the attack source is more than one, often thousands of, unique IP addresses. It is analogous to a group of people crowding the entry door or gate to a shop or business, and not letting legitimate parties enter into the shop or business, disrupting normal operations.

A DoS attack tries to make a web resource unavailable to its users by flooding the target URL with more requests than the server can handle. That means that during the attack period, regular traffic on the website will be either slowed down or completely interrupted.

A Distributed Denial of Service (DDoS) attack is a DoS attack that comes from more than one source at the same time. A DDoS attack is typically generated using thousands (potentially hundreds of thousands) of unsuspecting zombie machines. The machines used in such attacks are collectively known as “botnets” and will have previously been infected with malicious software, so they can be remotely controlled by the attacker. According to research, tens of millions of computers are likely to be infected with botnet programs worldwide.

Cybercriminals use DoS attacks to extort money from companies that rely on their websites being accessible. But there have also been examples of legitimate businesses having paid underground elements of the Internet to help them cripple rival websites. In addition, cybercriminals combine DoS attacks and phishing to target online bank customers. They use a DoS attack to take down the bank's website and then send out phishing e-mails to direct customers to a fake emergency site instead.

### Installation Steps:

- Install Hping3 and wireshark
- Flood the victim with TCP/ICMP/UDP packet using Hping3 (-- flood option)
- Observe the Dos attack and DDos attack using Wireshark

### Output

```
student@project-OptiPlex-360: ~
len=46 ip=192.168.0.220 ttl=64 DF id=59844 sport=0 flags=RA seq=259 win=0 rtt=4.
4 ms
^C
--- 192.168.0.220 hping statistic ---
260 packets transmitted, 260 packets received, 0% packet loss
round-trip min/avg/max = 0.2/2.6/4.4 ms
student@project-OptiPlex-360:~$ ifconfig
eth1      Link encap:Ethernet  HWaddr 00:25:64:92:fb:82
          inet addr:192.168.0.211  Bcast:192.168.255.255  Mask:255.255.0.0
          inet6 addr: fe80::225:64ff:fe92:fb82/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:58352 errors:0 dropped:66 overruns:0 frame:0
          TX packets:1612 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:6002472 (6.0 MB)  TX bytes:115578 (115.5 KB)
          Interrupt:16

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:245 errors:0 dropped:0 overruns:0 frame:0
          TX packets:245 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
```

```
student@project-OptiPlex-360: ~
ms
len=46 ip=192.168.0.220 ttl=64 DF id=59858 sport=0 flags=RA seq=13 win=0 rtt=2.1
ms
len=46 ip=192.168.0.220 ttl=64 DF id=59859 sport=0 flags=RA seq=14 win=0 rtt=2.1
ms
len=46 ip=192.168.0.220 ttl=64 DF id=59860 sport=0 flags=RA seq=15 win=0 rtt=2.2
ms
len=46 ip=192.168.0.220 ttl=64 DF id=59861 sport=0 flags=RA seq=16 win=0 rtt=2.1
ms
len=46 ip=192.168.0.220 ttl=64 DF id=59862 sport=0 flags=RA seq=17 win=0 rtt=2.1
ms
len=46 ip=192.168.0.220 ttl=64 DF id=59863 sport=0 flags=RA seq=18 win=0 rtt=2.1
ms
len=46 ip=192.168.0.220 ttl=64 DF id=59864 sport=0 flags=RA seq=19 win=0 rtt=2.1
ms
len=46 ip=192.168.0.220 ttl=64 DF id=59865 sport=0 flags=RA seq=20 win=0 rtt=2.1
ms
len=46 ip=192.168.0.220 ttl=64 DF id=59866 sport=0 flags=RA seq=21 win=0 rtt=2.1
ms
len=46 ip=192.168.0.220 ttl=64 DF id=59867 sport=0 flags=RA seq=22 win=0 rtt=2.1
ms
len=46 ip=192.168.0.220 ttl=64 DF id=59868 sport=0 flags=RA seq=23 win=0 rtt=2.1
ms
```

```
student@acpce-OptiPlex-380: ~
1 ms
len=46 ip=192.168.0.220 ttl=64 DF id=56651 sport=0 flags=RA seq=664 win=0 rtt=1.
1 ms
len=46 ip=192.168.0.220 ttl=64 DF id=56652 sport=0 flags=RA seq=665 win=0 rtt=1.
1 ms
len=46 ip=192.168.0.220 ttl=64 DF id=56653 sport=0 flags=RA seq=666 win=0 rtt=1.
1 ms
^[[2;3~len=46 ip=192.168.0.220 ttl=64 DF id=56654 sport=0 flags=RA seq=667 win=0
rtt=4.4 ms
len=46 ip=192.168.0.220 ttl=64 DF id=56655 sport=0 flags=RA seq=668 win=0 rtt=4.
4 ms
len=46 ip=192.168.0.220 ttl=64 DF id=56656 sport=0 flags=RA seq=669 win=0 rtt=4.
4 ms
len=46 ip=192.168.0.220 ttl=64 DF id=56657 sport=0 flags=RA seq=670 win=0 rtt=4.
5 ms
len=46 ip=192.168.0.220 ttl=64 DF id=56658 sport=0 flags=RA seq=671 win=0 rtt=4.
3 ms
len=46 ip=192.168.0.220 ttl=64 DF id=56659 sport=0 flags=RA seq=672 win=0 rtt=4.
3 ms
len=46 ip=192.168.0.220 ttl=64 DF id=56660 sport=0 flags=RA seq=673 win=0 rtt=4.
3 ms
len=46 ip=192.168.0.220 ttl=64 DF id=56661 sport=0 flags=RA seq=674 win=0 rtt=3.
9 ms
```

```
student@acpce-OptiPlex-380: ~
student@acpce-OptiPlex-380:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr b8:ac:6f:1f:b9:6a
          inet addr:192.168.0.219  Bcast:192.168.255.255  Mask:255.255.0.0
          inet6 addr: fe80::baac:6fff:fe1f:b96a/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3528 errors:0 dropped:0 overruns:0 frame:0
          TX packets:218 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:419904 (419.9 KB)  TX bytes:23592 (23.5 KB)
          Interrupt:16

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:201 errors:0 dropped:0 overruns:0 frame:0
          TX packets:201 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:14634 (14.6 KB)  TX bytes:14634 (14.6 KB)

student@acpce-OptiPlex-380:~$
```

No.	Time	Source	Destination	Protocol	Length	Info
868	8.674683	192.168.0.219	192.168.0.220	TCP	60	event-port > 0 [<None>] Seq=1 Win=512 Len=0
869	8.674704	192.168.0.220	192.168.0.219	TCP	54	0 > event-port [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
956	9.674744	192.168.0.219	192.168.0.220	TCP	60	ah-esp-encap > 0 [<None>] Seq=1 Win=512 Len=0
957	9.674762	192.168.0.220	192.168.0.219	TCP	54	0 > ah-esp-encap [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
1140	10.674810	192.168.0.219	192.168.0.220	TCP	60	acp-port > 0 [<None>] Seq=1 Win=512 Len=0
1141	10.674831	192.168.0.220	192.168.0.219	TCP	54	0 > acp-port [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
1236	11.674870	192.168.0.219	192.168.0.220	TCP	60	msync > 0 [<None>] Seq=1 Win=512 Len=0
1237	11.674888	192.168.0.220	192.168.0.219	TCP	54	0 > msync [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
1339	12.674930	192.168.0.219	192.168.0.220	TCP	60	gxs-data-port > 0 [<None>] Seq=1 Win=512 Len=0
1340	12.674948	192.168.0.220	192.168.0.219	TCP	54	0 > gxs-data-port [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
1436	13.674992	192.168.0.219	192.168.0.220	TCP	60	virtl-vmf-sa > 0 [<None>] Seq=1 Win=512 Len=0
1437	13.675010	192.168.0.220	192.168.0.219	TCP	54	0 > virtl-vmf-sa [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
1540	14.675052	192.168.0.219	192.168.0.220	TCP	60	newlixengine > 0 [<None>] Seq=1 Win=512 Len=0
1541	14.675064	192.168.0.220	192.168.0.219	TCP	54	0 > newlixengine [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
1622	15.675115	192.168.0.219	192.168.0.220	TCP	60	newlixconfig > 0 [<None>] Seq=1 Win=512 Len=0
1623	15.675137	192.168.0.220	192.168.0.219	TCP	54	0 > newlixconfig [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
1691	16.675176	192.168.0.219	192.168.0.220	TCP	60	tsrmagt > 0 [<None>] Seq=1 Win=512 Len=0
1692	16.675193	192.168.0.220	192.168.0.219	TCP	54	0 > tsrmagt [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
1764	17.675240	192.168.0.219	192.168.0.220	TCP	60	tpcsrvr > 0 [<None>] Seq=1 Win=512 Len=0
1765	17.675253	192.168.0.220	192.168.0.219	TCP	54	0 > tpsrvr [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
1849	18.675300	192.168.0.219	192.168.0.220	TCP	60	idware-router > 0 [<None>] Seq=1 Win=512 Len=0
1850	18.675318	192.168.0.220	192.168.0.219	TCP	54	0 > idware-router [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
1931	19.675365	192.168.0.219	192.168.0.220	TCP	60	autodesk-nlm > 0 [<None>] Seq=1 Win=512 Len=0
1932	19.675387	192.168.0.220	192.168.0.219	TCP	54	0 > autodesk-nlm [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
2036	20.675433	192.168.0.219	192.168.0.220	TCP	60	kme-trap-port > 0 [<None>] Seq=1 Win=512 Len=0
2037	20.675459	192.168.0.220	192.168.0.219	TCP	54	0 > kme-trap-port [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
2133	21.675485	192.168.0.219	192.168.0.220	TCP	60	infowave > 0 [<None>] Seq=1 Win=512 Len=0
2134	21.675504	192.168.0.220	192.168.0.219	TCP	54	0 > infowave [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
2208	22.675545	192.168.0.219	192.168.0.220	TCP	60	radsec > 0 [<None>] Seq=1 Win=512 Len=0
2209	22.675562	192.168.0.220	192.168.0.219	TCP	54	0 > radsec [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
2297	23.675611	192.168.0.219	192.168.0.220	TCP	60	sunclustergeo > 0 [<None>] Seq=1 Win=512 Len=0
2298	23.675630	192.168.0.220	192.168.0.219	TCP	54	0 > sunclustergeo [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
2408	24.675673	192.168.0.219	192.168.0.220	TCP	60	ada-cip > 0 [<None>] Seq=1 Win=512 Len=0
2409	24.675685	192.168.0.220	192.168.0.219	TCP	54	0 > ada-cip [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

▶ Frame 101: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)  
 ▶ Ethernet II, Src: Dell\_1f:ba:bd (b8:ac:6f:1f:ba:bd), Dst: Dell\_23:ee:4c (b8:ac:6f:23:ee:4c)  
 ▶ Internet Protocol Version 4, Src: 192.168.0.219 (192.168.0.219), Dst: 192.168.0.220 (192.168.0.220)  
 ▶ Transmission Control Protocol, Src Port: netmount (2061), Dst Port: 0 (0), Seq: 1, Len: 0

```

0000  b8 ac 6f 23 ee 4c b8 ac 6f 1f ba bd 08 00 45 00  ..o#...o....E.
0010  00 28 2d fe 00 00 40 06 c9 ca c0 a8 00 db c0 a8  .(-...f. ....
0020  00 dc 08 0d 00 00 66 38 77 74 5f 9e 93 d7 50 00  ....f8 wt....P.
0030  02 00 51 ad 00 00 00 00 00 00 00 00 00 00 00  ..Q.....
  
```

## **6. Conclusion:**

DoS attacks affects availability of resources. DoS attack is simulated using Hping3 and resources are monitored using Wireshark.

## **7. Viva Questions:**

- What is DoS Attack?
- How can you prevent DoS?

## **8. References:**

1. William Stallings, “*Cryptography and Network Security: Principles and Practice*”, Pearson Education, Fifth edition.
2. Bernard Menezes, “*Network Security and Cryptography*”, Cengage Learning, Second Edition.
3. Behrouz A Forouzan, Debdeep Mukhopadhyay, “*Cryptography and Network Security*”, Tata McGraw Hill, Second edition
4. Behrouz A. Forouzan, “*Cryptography and Network Security*”, Tata McGraw Hill.
5. Charles P. Pfleeger, “*Security in Computing*”, Pearson Education.

# **Cryptography and System Security Lab**

## **Experiment No.: 9**

**Simulate buffer overflow attack using  
Splint, Cppcheck etc.**

# Experiment No. 9

1. **Aim:** Merry received a mail with C++ program as an attachment. She suspects some software vulnerability in the received attachment. Help her to inspect the program using Splint, Cppcheck etc

2. **Objectives:**

- Understand the concept of buffer overflow.
- Understand and use Splint and cppcheck to detect buffer overflow.

3. **Outcomes:** The learner will be able to  
Install and use Splint and cppcheck tools to check source code to detect various vulnerabilities causing buffer overflow attack.

4. **Hardware / Software Required:** Unix/Linux/Windows, Splint, cppcheck

5. **Theory:**

1. **Cppcheck :** Cppcheck is a tool for static C/C++ code analysis (CLI). Cppcheck is a command-line tool that tries to detect bugs that your C/C++ compiler doesn't see. It is versatile, and can check non-standard code including various compiler extensions, inline assembly code, etc. Its internal preprocessor can handle includes, macros, and several pre-processor commands. While Cppcheck is highly configurable, you can start using it just by giving it a path to the source code.

It includes checks for:

- \* pointers to out-of-scope auto variables;
- \* assignment of auto variables to an effective parameter of a function;
- \* out-of-bounds errors in arrays and STL;
- \* missing class constructors;
- \* variables not initialized by a constructor;
- \* use of memset, memcpy, etcetera on a class;
- \* non-virtual destructors for base classes;
- \* operator= not returning a constant reference to itself;
- \* use of deprecated functions (mktemp, gets, scanf);

- \* exceptions thrown in destructors;
- \* memory leaks in class or function variables;
- \* C-style pointer cast in C++ code;
- \* redundant if;
- \* misuse of the strtol or sprintf functions;
- \* unsigned division or division by zero;
- \* unused functions and struct members;
- \* passing parameters by value;
- \* misuse of signed char variables;
- \* unusual pointer arithmetic (such as "abc" + 'd');
- \* dereferenced null pointers;
- \* incomplete statements;
- \* misuse of iterators when iterating through a container;
- \* dereferencing of erased iterators;
- \* use of invalidated vector iterators/pointers;

### **Step 1: Installation of cppcheck**

**\$sudo apt-get install cppcheck**

### **Step 2: Checking Vulnerability**

#### **Sample2.c**

```
char firstChar1 (/*@null@*/ char *s)
{
    return *s;
}

char firstChar2 (/*@null@*/ char *s)
{
    if (s == NULL) return '\0';
    return *s;
}
```





**\$ cppcheck sample2.c**

```
student@lab522linuxprinter:~$ cppcheck sample2.c
Checking sample2.c...
[sample2.c:10]: (error) The code contains unhandled characters (character code = 0xe2). Checking continues, but do not expect valid results.
[sample2.c:10]: (error) The code contains unhandled characters (character code = 0x80). Checking continues, but do not expect valid results.
[sample2.c:10]: (error) The code contains unhandled characters (character code = 0x98). Checking continues, but do not expect valid results.
[sample2.c:10]: (error) The code contains unhandled characters (character code = 0x99). Checking continues, but do not expect valid results.
student@lab522linuxprinter:~$
```

2. **Splint** : Splint is a tool for statically checking C programs for security vulnerabilities and programming mistakes. Splint does many of the traditional lint checks including unused declarations, type inconsistencies, use before definition, unreachable code, ignored return values, execution paths with no return, likely infinite loops, and fall through cases. More powerful checks are made possible by additional information given in source code annotations. Annotations are stylized comments that document assumptions about functions, variables, parameters and types. In addition to the checks specifically enabled by annotations, many of the traditional lint checks are improved by exploiting this additional information.

Splint is designed to be flexible and allow programmers to select appropriate points on the effort-benefit curve for particular projects. As different checks are turned on and more information is given in code annotations the number of bugs that can be detected increases dramatically. Problems detected by Splint include:

- Dereferencing a possibly null pointer
- Using possibly undefined storage or returning storage that is not properly defined
- Type mismatches, with greater precision and flexibility than provided by C compilers
- Violations of information hiding
- Memory management errors including uses of dangling references and memory leaks
- Dangerous aliasing
- Modifications and global variable uses that are inconsistent with specified interfaces
- Problematic control flow such as likely infinite loops, fall through cases or incomplete switches and suspicious statements
- Buffer overflow vulnerabilities
- Dangerous macro implementations or invocations
- Violations of customized naming conventions

**Examples1 :**

**\$ splint sample2.c**

```
student@lab522linuxprinter:~$ splint sample2.c
splint 3.1.2 --- 03 May 2009

sample2.c: (in function firstChar1)
sample2.c:11: Dereference of possibly null pointer s: *s
  A possibly null pointer is dereferenced. Value is either the result of a
  function which may return null (in which case, code should check it is not
  null), or a global, parameter or structure field declared with the null
  qualifier. (Use -nullderef to inhibit warning)
sample2.c:13:35: Storage s may become null
sample2.c: (in function firstChar2)
sample2.c:16:27: Invalid character (ascii: -30), skipping character
  Code cannot be parsed. For help on parse errors, see splint -help
  parseerrors. (Use -syntax to inhibit warning)
sample2.c:16:28: Invalid character (ascii: -128), skipping character
sample2.c:16:29: Invalid character (ascii: -104), skipping character
sample2.c:16:32: Invalid character (ascii: -30), skipping character
sample2.c:16:33: Invalid character (ascii: -128), skipping character
sample2.c:16:34: Invalid character (ascii: -103), skipping character
sample2.c:16:30: Return value type int does not match declared type char: 0
  Types are incompatible. (Use -type to inhibit warning)

Finished checking --- 8 code warnings
student@lab522linuxprinter:~$
```

## Example 2:

### Sample3.c

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

int main(void)
{
    char buff[15];
    int pass = 0;

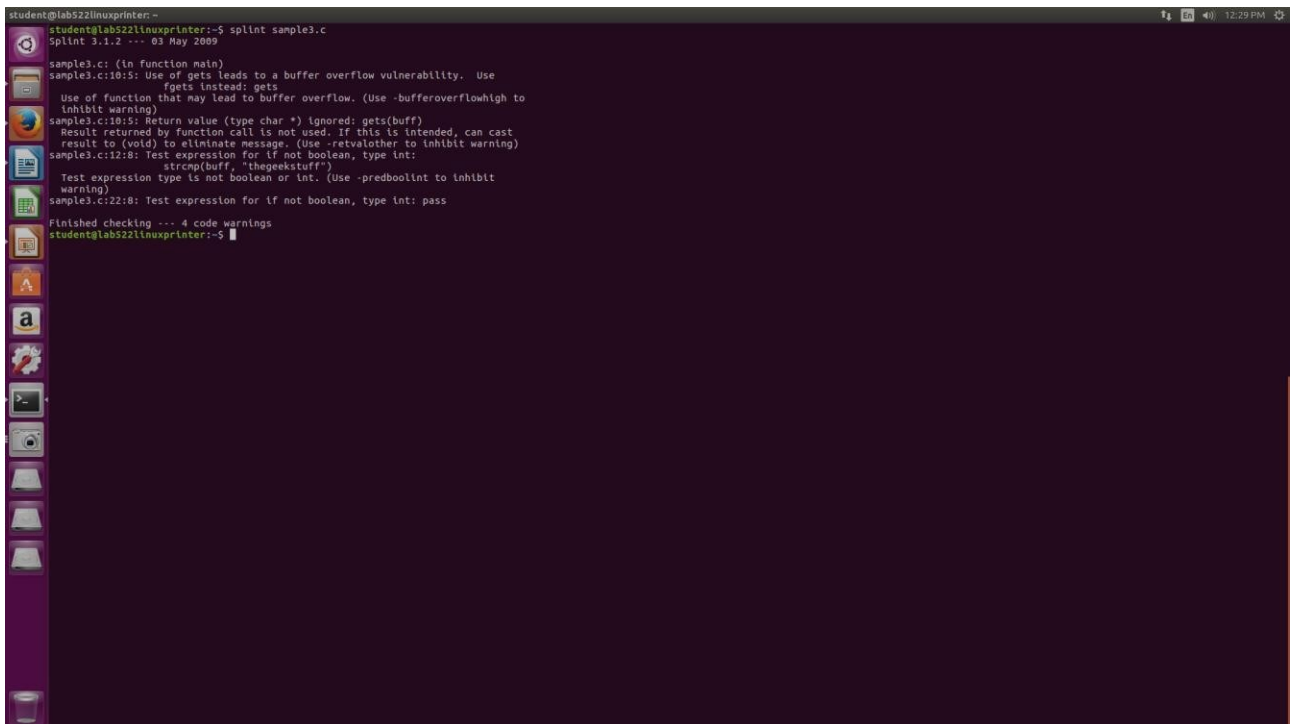
    printf("\n Enter the password : \n");
    gets(buff);

    if(strcmp(buff, "thegeekstuff"))
    {
        printf ("\n Wrong Password \n");
    }
    else
    {
        printf ("\n Correct Password \n");
        pass = 1;
    }
}
```

```
if(pass)
{
    /* Now Give root or admin rights to user*/
    printf ("\n Root privileges given to the user \n");
}

return 0;
}
```

**\$splint sample3.c**



```
student@lab522linuxprinter:~$ splint sample3.c
Splint 3.1.2 --- 03 May 2009

sample3.c: (in function main)
sample3.c:10:5: Use of gets leads to a buffer overflow vulnerability. Use
      fgets instead: gets
      Use of function that may lead to buffer overflow. (Use -bufferoverflowhigh to
      inhibit warning)
sample3.c:10:5: Return value (type char *) ignored: gets(buff)
      Result returned by function call is not used. If this is intended, can cast
      result to (void) to eliminate message. (Use -retvalother to inhibit warning)
sample3.c:12:8: Test expression for if not boolean, type int:
      strcmp(buff, "thegeekstuff")
      Test expression type is not boolean or int. (Use -predboolint to inhibit
      warning)
sample3.c:22:8: Test expression for if not boolean, type int: pass
Finished checking --- 4 code warnings
student@lab522linuxprinter:~$
```

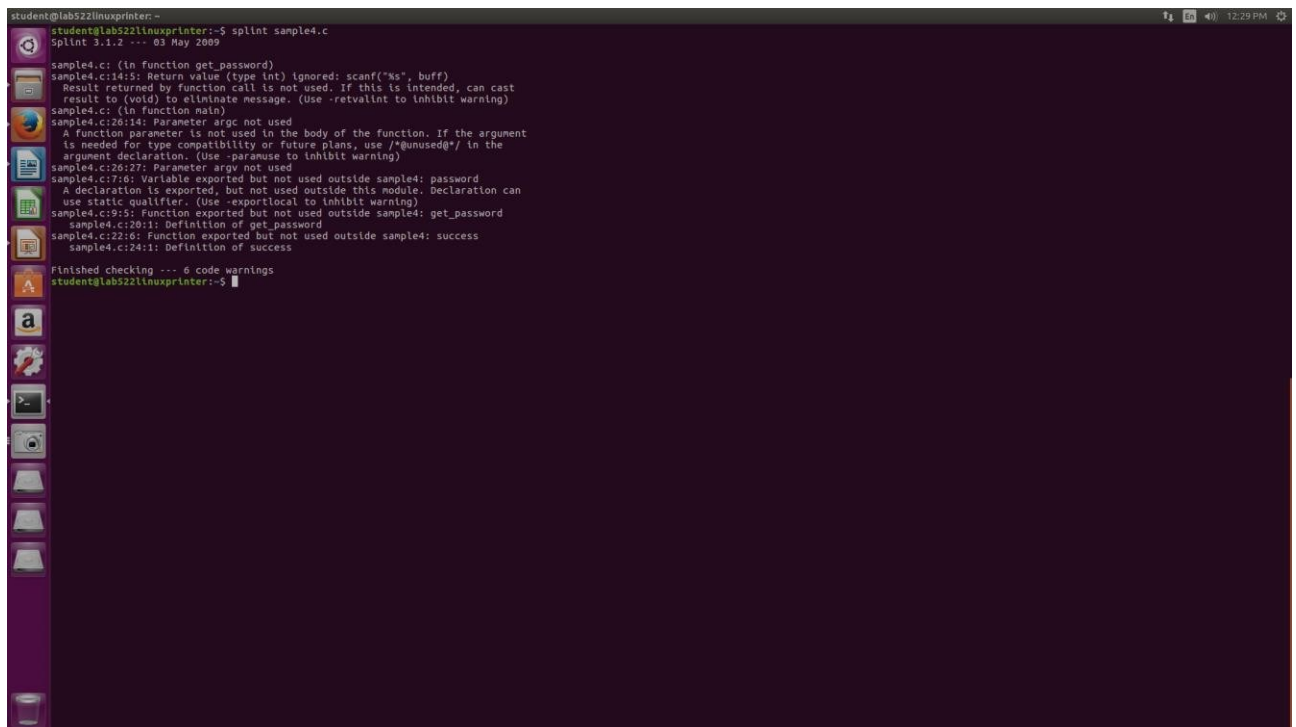
### Example 3:

#### Sample4.c

```
#include <stdio.h>
#include <string.h>
char password[] = "password";
int get_password() {
    int auth_ok = 0;
    char buff[16];
    printf("Enter password: ");
    scanf("%s", buff);
    if(strncmp(buff, password, sizeof(password)) == 0)
        auth_ok = 1;
    return auth_ok;
}
```

```
void success() {  
    printf("Success!\n");  
}  
int main(int argc, char** argv) {  
    int res = get_password();  
    if (res == 0) {  
        printf("Failure\n");  
        return 0;  
    }  
    success();  
    return 0;  
}
```

### \$splint sample4.c



```
student@lab522linuxprinter:~$ splint sample4.c  
splint 3.1.2 --- 03 May 2009  
  
sample4.c: (in function get_password)  
sample4.c:14:5: Return value (type int) ignored: scanf("%s", buff)  
    Result returned by function call is not used. If this is intended, can cast  
    result to (void) to eliminate message. (Use -retvalint to inhibit warning)  
sample4.c: (in function main)  
sample4.c:20:14: Parameter argc not used  
    A function parameter is not used in the body of the function. If the argument  
    is needed for type compatibility or future plans, use /*@unused@*/ in the  
    argument declaration. (Use -paramuse to inhibit warning)  
sample4.c:20:27: Parameter argv not used  
sample4.c:7:6: Variable exported but not used outside sample4: password  
    A declaration is exported, but not used outside this module. Declaration can  
    use static qualifier. (Use -exportlocal to inhibit warning)  
sample4.c:9:5: Function exported but not used outside sample4: get_password  
sample4.c:20:1: Definition of get_password  
sample4.c:22:5: Function exported but not used outside sample4: success  
sample4.c:24:1: Definition of success  
  
Finished checking --- 6 code warnings  
student@lab522linuxprinter:~$
```

## 6. Conclusion:

We have simulated and studied different tools like Ollydbg, splint and cppcheck to detect software vulnerability causing buffer overflow attack.

## 7. Viva Questions:

- What do you mean by buffer overflow?

## 8. References:

1. William Stallings, “*Cryptography and Network Security: Principles and Practice*”, Pearson education, Fifth edition.
2. Bernard Menezes, “*Network Security and Cryptography*”, Cengage Learning, Second Edition.
3. Behrouz A Forouzan, Debdeep Mukhopadhyay, “*Cryptography and Network Security*”, Tata McGraw Hill, Second edition
4. Behrouz A. Forouzan, “*Cryptography and Network Security*”, Tata McGraw Hill.
5. Charles P. Pfleeger, “*Security in Computing*”, Pearson Education.

# **Cryptography and System Security Lab**

## **Experiment No.: 10**

### **Setting up personal Firewall using iptables**

# Experiment No. 10

1. **Aim:** John wants to restrict usage of web resources on his personal computer. Help him to set the rules using iptables
2. **Objectives:**
  - Understand the need for personal firewall.
  - Learn how to use iptables to set firewall rules.
3. **Outcomes:** The learner will be able to  
Design their own rule and set up personal firewall at desktop using iptables.  
security problems with current research issues.
4. **Hardware / Software Required:** Unix/Linux, iptables
5. **Theory:**

All packets inspected by iptables pass through a sequence of built-in tables (queues) for processing. Each of these queues is dedicated to a particular type of packet activity and is controlled by an associated packet transformation/filtering chain.

There are three tables in total. The first is the mangle table which is responsible for the alteration of quality of service bits in the TCP header. This is hardly used in a home or SOHO environment.

The second table is the filter queue which is responsible for packet filtering. It has three built-in chains in which you can place your firewall policy rules. These are the:

- **Forward chain:** Filters packets to servers protected by the firewall.
- **Input chain:** Filters packets destined for the firewall.
- **Output chain:** Filters packets originating from the firewall.

**ACCEPT :** iptables stops further processing.

The packet is handed over to the end application or the operating system for processing.

**DROP :** iptables stops further processing.

The packet is blocked

**REJECT :** Works like the DROP target, but will also return an error message to the host sending the packet that the packet was blocked



## How To Start iptables

You can start, stop, and restart iptables after booting by using the commands:

**\$service iptables start**

**\$service iptables stop**

**\$service iptables restart**

## Determining The Status of iptables

You can determine whether iptables is running or not via the service iptables status command. Fedora Core will give a simple status message. For example

**\$service iptables status**

Firewall is stopped.

## Iptables commands

iptables command Switch	Description
-L	Listing of rules present in the chain
-n	Numeric output of addresses and ports
-v	Displays the rules in verbose mode
-t <-table->	If you don't specify a table, then the filter table is assumed. As discussed before, the possible built-in tables include: filter, nat, mangle
-j <target>	Jump to the specified target chain when the packet matches the current rule.
-A	Append rule to end of a chain
-F	Flush. Deletes all the rules in the selected table
-p <protocol-type>	Match protocol. Types include, icmp, tcp, udp, and all
-s <ip-address>	Match source IP address
-d <ip-address>	Match destination IP address
-i <interface-name>	Match "input" interface on which the packet enters.
-o <interface-name>	Match "output" interface on which the packet exits

## Setting Firewall Rules using iptables

### 1. Enabling traffic on localhost

```
$iptables -A INPUT -i eth0 -j ACCEPT
```

### 2. Drop the ICMP Packet coming from source

```
$iptables -I INPUT -p icmp -icmp -type 8 -j DROP -s (10.0.204.15) -d (10.0.0.16)
```

### 3. ICMP packet going from source are not allowed

```
$iptables -I OUTPUT -p icmp -j DROP -s (10.0.0.16) -d (10.0.204.15)
```

### 4. To drop all kind of packets

```
$iptables -I INPUT -p all
```

### 6. Conclusion:

Firewall plays an important role on security architecture of a medium size or large organizations. Personal firewall is implemented using ruleset in iptables.

### 7. Viva Questions:

- What is the role of firewall?
- How to implement personal firewall using iptables?

### 8. References:

1. William Stallings, “*Cryptography and Network Security: Principles and Practice*”, Pearson education, Fifth edition.
2. Bernard Menezes, “*Network Security and Cryptography*”, Cengage Learning, Second Edition.
3. Behrouz A Forouzan, Debdeep Mukhopadhyay, “*Cryptography and Network Security*”, Tata McGraw Hill, Second edition
4. Behrouz A. Forouzan, “*Cryptography and Network Security*”, Tata McGraw Hill.
5. Charles P. Pfleeger, “*Security in Computing*”, Pearson Education.

# **Cryptography and System Security Lab**

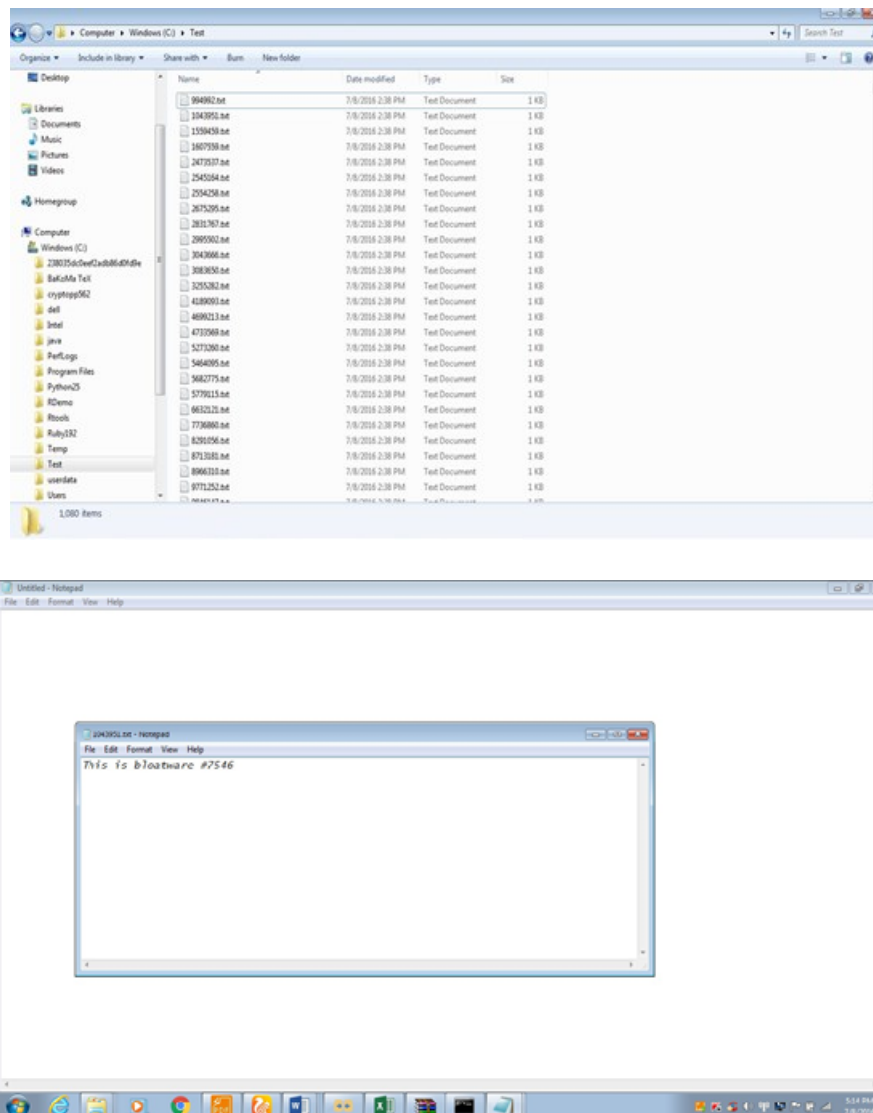
## **Experiment No.: 11**

### **Implementation of Virus and Antivirus.**

# Experiment No. 11

1. **Aim:** John was curious to learn the basic working of Virus and how it affects system. Also, he wants to neutralize the actions taken by virus. Help him to write basic script to do this task.
2. **Objective:** Study of virus and antivirus script writing.
3. **Outcome:** Students will be able to create simple script for virus and antivirus and study impact of it on system.
4. **Theory:** A virus is a malicious code that is loaded on to your device with an intent to cause damage and steal information. Computer viruses replicate multiple self-copies and occupy all the available memory and result in system damage. Some viruses can replicate and pass on its copies across various networks and bypass security systems as well. To enable protection to your computer system – an antivirus program should be installed. The antivirus software scans, identifies and removes viruses, computer worms, Trojans, etc. Most of the antivirus programs are equipped with auto-update feature to stay up-to-date with new virus definitions that are released in the wild recently. They offer on-demand and on-access scanning options and choice varies from user to user.
5. **Algorithm\Implementation Steps\Installation Steps:**
  - Step 1: Create your target folder - this is the folder you want your virus to attack. For demonstration purposes, I have created a folder called Test under the C Drive. C:\Test
  - Step 2: Create batch file to create virus (bloatware.bat).  
This virus would create an endless number of text files which contains a different number in the text document each time.

```
@echo off
color 0a
msg *You have just launched BloatWarez %random%
:Reckon
echo This is bloatware #%random% >C:\Test\%random%%random%.txtgoto Reckon
```



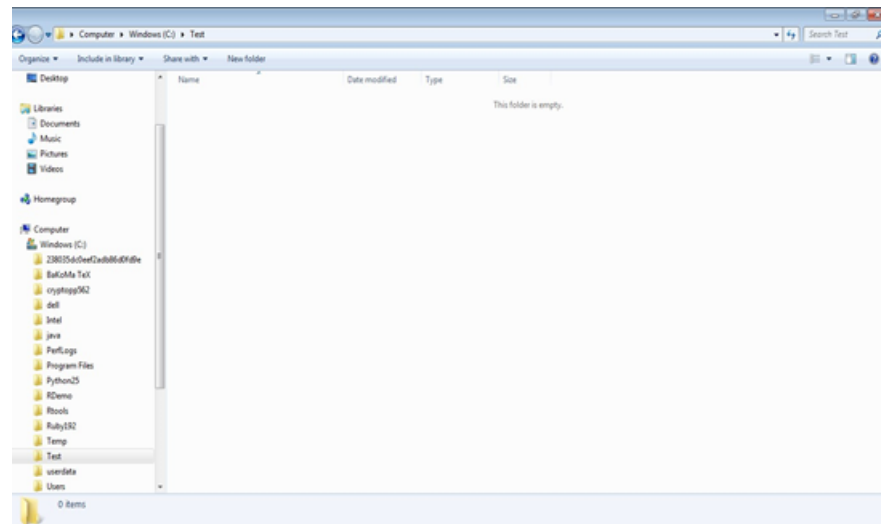
## Creation of Antivirus

An antivirus, identifies and removes (quarantines) the infection.

Step 1: Create batch file to remove infection of virus (antibloatware.bat)

```
@echo off
color 0a
msg *Removing BloatWarez Virus
del /s *.virus.txt
```

Step 2: Run the antibloatware.bat to remove the files.



6. **Conclusion:** Implemented virus and antivirus using simple script writing.

7. **Viva Questions:**

- What is Virus? How it can be neutralized?
- How to check whether system is affected by Virus?
- How to find spreading pattern of virus?

8. **References:**

6. William Stallings, "*Cryptography and Network Security: Principles and Practice*", Pearson education, Fifth edition.
7. Bernard Menezes, "*Network Security and Cryptography*", Cengage Learning, Second Edition.
8. Behrouz A Forouzan, Debdeep Mukhopadhyay, "*Cryptography and Network Security*", Tata McGraw Hill, Second edition
9. Behrouz A. Forouzan, "*Cryptography and Network Security*", Tata McGraw Hill.
10. Charles P. Pfleeger, "*Security in Computing*", Pearson Education.