Experiment 01: Implement and design the product cipher using Substitution and Transposition ciphers

Experiment 01: (a) Substitution Cipher

Learning Objective: Implement and design the product cipher using Substitution Cipher

Tools: PyCharm

Theory:

Substitution ciphers are a method of encrypting plaintext by swapping each letter or symbol in the text with a different symbol, based on a specific key. The Caesar cipher is perhaps the simplest and most well-known of these substitution ciphers. It is named after the man who first used it. This cipher is also called a shift cipher or a mono-alphabetic cipher, which differentiates it from other more complex substitution ciphers.

In a Caesar cipher, the plaintext is represented in lowercase letters, while the ciphertext is represented in uppercase letters. Spaces are added to the ciphertext for readability, but they are removed in a real application to make attacking the ciphertext more difficult. Simple substitution of single letters separately can be demonstrated by writing out the alphabet in some order to represent the substitution. This is known as a substitution alphabet. The cipher alphabet can be shifted, reversed, or scrambled in a more complex way to create different types of substitution ciphers.

Mixed alphabets or deranged alphabets can also be used to create substitution ciphers. These are traditionally created by writing out a keyword and removing any repeated letters, then writing all the remaining letters in the alphabet in their usual order. This creates a unique mixed alphabet that can be used as the basis for the cipher. Substitution ciphers have a long history, and although they are not as secure as modern encryption methods, they are still used in some applications today.

Code:

Caesar Cipher.py

```python
# A python program for Caesar Cipher Technique
def encrypt(text, s):
    result = ""
    # traverse text
    for i in range(len(text)):
        char = text[i]
        # Encrypt uppercase characters
        if (char.isupper()):
            result += chr((ord(char) + s - 65) % 26 + 65)
        # Encrypt lowercase characters
        else:
            result += chr((ord(char) + s - 97) % 26 + 97)
    return result

def decrypt(text, s):
    result = ""
    # traverse text
    for i in range(len(text)):
        char = text[i]
        # Decrypt uppercase characters
        if (char.isupper()):
            result += chr((ord(char) - s - 65) % 26 + 65)
        # Decrypt lowercase characters
        else:
            result += chr((ord(char) - s - 97) % 26 + 97)
    return result
```

```python

# Get the plain text and shift key from user input
text = input("\n"+"Enter the Plain Text: ")
s = int(input("Enter the value of the key: "))

print("\n-----------------------\n")
print("Plain Text : " + text)
print("Key: " + str(s))
a = encrypt(text, s)
print("Cipher Text: " + a)
print("Decrypted Text: " + decrypt(a, s))

print("\n-----------------------\n")

```

Output:

```
Run: Caesar Cipher
"C:\Programming Repository\PyCharm\Sem-06\CSS\venv\Scripts\python.exe" "C:\Programming Repository\PyCharm\Sem-06\CSS\Caesar Cipher.py"

Enter the Plain Text: AtTaCkAtOnCe
Enter the value of the key: 4

-----------------------

Plain Text : AtTaCkAtOnCe
Key: 4
Cipher Text: ExXeGoExSrGi
Decrypted Text: AtTaCkAtOnCe

-----------------------
```

Conclusion: After performing the experiment I was able to implement Substitution Cipher.

Experiment 01: (b) Transposition Cipher

Learning Objective: Implement and design the product cipher using Transposition Cipher

Tools: PyCharm

Theory:

Transposition ciphers are often used in combination with other encryption methods such as substitution ciphers to create a more secure encryption. By adding the additional layer of transposition, the resulting ciphertext becomes much more difficult to decipher without knowledge of both encryption methods. A common method of implementing transposition ciphers is through the use of a rectangular grid, where the plaintext is written out horizontally and then read vertically in a certain order to create the ciphertext. Other methods may involve shuffling the order of words or phrases in the plaintext message.

One of the most famous examples of a transposition cipher is the Rail Fence cipher, which involves writing the plaintext diagonally on alternate lines, and then reading the ciphertext vertically. This creates a zig-zag pattern that is difficult to decipher without knowledge of the exact transposition method used. Overall, transposition ciphers offer a flexible and relatively easy method of encryption that can be used in combination with other methods to create a more secure and complex encryption.

Code:

Transposition Cipher.py

```python
# Python3 implementation of Columnar Transposition
import math

# Encryption
def encryptMessage(msg):
    cipher = ""
    k_indx = 0          # track key indices
    msg_len = float(len(msg))
    msg_lst = list(msg)
    key_lst = sorted(list(key))
    # calculate column of the matrix
    col = len(key)
    # calculate maximum row of the matrix
    row = int(math.ceil(msg_len / col))
    # add the padding character '_' in empty
    # the empty cell of the matix
    fill_null = int((row * col) - msg_len)
    msg_lst.extend('_' * fill_null)
```

```
    # create Matrix and insert message and
    # padding characters row-wise
    matrix = [msg_lst[i: i + col]
              for i in range(0, len(msg_lst), col)]
    # read matrix column-wise using key
    for _ in range(col):
        curr_idx = key.index(key_lst[k_indx])
        cipher += ''.join([row[curr_idx]
                           for row in matrix])
        k_indx += 1
    return cipher

# Decryption
def decryptMessage(cipher):
    msg = ""
    # track key indices
    k_indx = 0
    # track msg indices
    msg_indx = 0
    msg_len = float(len(cipher))
    msg_lst = list(cipher)
    # calculate column of the matrix
    col = len(key)
    # calculate maximum row of the matrix
    row = int(math.ceil(msg_len / col))
```

```python
    # convert key into list and sort
    # alphabetically so we can access
    # each character by its alphabetical position.
    key_lst = sorted(list(key))
    # create an empty matrix to
    # store deciphered message
    dec_cipher = []
    for _ in range(row):
        dec_cipher += [[None] * col]
    # Arrange the matrix column wise according
    # to permutation order by adding into new matrix
    for _ in range(col):
        curr_idx = key.index(key_lst[k_indx])
        for j in range(row):
            dec_cipher[j][curr_idx] = msg_lst[msg_indx]
            msg_indx += 1
        k_indx += 1
    # convert decrypted msg matrix into a string
    try:
        msg = ''.join(sum(dec_cipher, []))
    except TypeError:
        raise TypeError("This program cannot",
                        "handle repeating words.")
    null_count = msg.count('_')
    if null_count > 0:
        return msg[: -null_count]
    return msg

# Driver Code
msg = input("\n"+"Enter the Plain Text: ")
key = input("Enter the Key: ")

cipher = encryptMessage(msg)
print("Encrypted Message: {}".format(cipher))
print("Decryped Message: {}".format(decryptMessage(cipher)))
```

Output:

```
Run:  Transposition Cipher
"C:\Programming Repository\PyCharm\Sem-06\CSS\venv\Scripts\python.exe" "C:\Programming Repository\PyCharm\Sem-06\CSS\Transposition Cipher.py"

Enter the Plain Text: HelloWorld
Enter the Key: 53241
Encrypted Message: odlreollHW
Decryped Message: HelloWorld

Process finished with exit code 0
```

Conclusion: After performing the experiment I was able to implement Transposition Cipher.

For Faculty Use

Correction Parameters	Formative Assessment [40%]	Timely completion of Practical [40%]	Attendance / Learning Attitude [20%]	Total
Marks Obtained				

Experiment no.2

Aim: Case study on windows and linux commands

Learning Objective: Students should be able to understand and implement commands for windows and linux.

Tools: Windows and Linux operating system.

Theory:

1. **Ping**

PING (Packet Internet Groper) command is used to check the network connectivity between host and server/host. This command takes as input the IP address or the URL and sends a data packet to the specified address with the message "PING" and gets a response from the server/host this time is recorded which is called latency. Fast ping low latency means faster connection. Ping uses ICMP(Internet Control Message Protocol) to send an ICMP echo message to the specified host; if that host is available then it sends an ICMP reply message.
By default, ping commands send multiple requests -- usually four or five -- and display the results. The echo ping results show whether a particular request received a successful response. It also includes the number of bytes received and the time it took to receive a reply or the time-to-live.

2. **ipconfig**

The ipconfig is a Windows command-line utility used often to troubleshooting computer network issues. If you are a Linux user, this utility is similar to ifconfig. This is often used to determine the local IP address, subnet mask, the gateway address, and other network configuration of a computer. Additionally, this tool is used to refresh DHCP (Dynamic Host Configuration Protocol) and DNS (Domain Name System) settings
While most of the information provided by the ipconfig command-line utility can be found via a more user-friendly graphical interface, sometimes that interface may not be available and command prompt is your only available option. If you are a help desk technician or a network professional, it is recommended that you understand the command-line method of retrieving a computer's network configuration, and it some cases, performing network functions.

Ipconfig Parameters

Parameter	Description
/all	Display the full TCP/IP configuration information for all network adapters.
/release	Release the IPv4 address for the specified adapter.
/release6	Release the IPv6 address for the specified adapter.
/renew	Renew the IPv4 address for the specified adapter.
/renew6	Renew the IPv6 address for the specified adapter.
/flushdns	Purges the DNS Resolver cache.
/registerdns	Refreshes all DHCP leases and re-registers DNS names.
/displaydns	Display the contents of the DNS Resolver Cache.
/showclassid	Displays all the DHCP class IDs allowed for adapter.
/setclassid	Modifies the DHCP class ID.
/showclassid6	Displays all the IPv6 DHCP class IDs allowed for adapter.
/setclassid6	Modifies the IPv6 DHCP class ID.
/?	Displays help information.

3. hostname

hostname command in Linux is used to obtain the DNS(Domain Name System) name and set the system's hostname or NIS(Network Information System) domain name. A hostname is a name which is given to a computer and it attached to the network. Its main purpose is to uniquely identify over a network.

Syntax :

```
hostname -[option] [file]
```

Options:

- -a : This option is used to get alias name of the host system(if any). It will return an empty line if no alias name is set. This option enumerates all configured addresses on all network interfaces.
- -A : This option is used to get all FQDNs(Fully Qualified Domain Name) of the host system. It enumerates all configured addresses on all network interfaces. An output may display same entries repetitively.
- -b : Used to always set a hostname. Default name is used if none specified.

- -d : This option is used to get the Domain if local domains are set. It will not return anything(not even a blank line) if no local domain is set.
- -f : This option is used to get the Fully Qualified Domain Name(FQDN). It contains short hostname and DNS domain name.
- -F : This option is used to set the hostname specified in a file. Can be performed by the superuser(root) only.
- -i option:This option is used to get the IP(network) addresses. This option works only if the hostname is resolvable.
- -I : This option is used to get all IP(network) addresses. The option doesn't depend on resolvability of hostname.
- -s : This option is used to get the hostname in short. The short hostname is the section of hostname before the first period/dot(.). If the hostname has no period, the full hostname is displayed.
- -V : Gives version number as output.

4. **getmac**

The getmac is a Windows command-line utility used typically when troubleshooting network issues to retrieve the MAC address, also known as the physical address, of network adapters in a computer. The getmac will only able to retrieve MAC addresses (the 6-byte 'burned-in' physical/hardware address) of connected adapters. If an adapter is disabled (in Windows Device Manager for example), or is not connected to the network, getmac will not be able to retrieve its MAC address.
The getmac is not the only way command-line tool to identify the MAC address of a network adapter. The ipconfig utility can also be used for this purpose, along with other functions.

Parameter	Description
/s system	Specifies the remote system to connect to.This can be either an IP address or a host name (do not use backslashes). The default is the local computer.
/u [domain\]user	Specifies the user context under which the command should execute. The default is the permissions of the current logged on user on the computer issuing the command.
/p [password]	Specifies the password for the given user context. Prompts for input if omitted.
/fo format	Specifies the format in which the output is to be displayed. Valid values: "TABLE", "LIST", "CSV". Teh default is Table.
/nh	Specifies that the "Column Header" should not be displayed in the output. Valid only for TABLE and CSV formats.
/v	Specifies that verbose output is displayed.
/?	Displays help information.

5. **arp**

arp command manipulates the System's ARP cache. It also allows a complete dump of the ARP cache. ARP stands for Address Resolution Protocol. The primary function of this protocol is to resolve the IP address of a system to its mac address, and hence it works between level 2(Data link layer) and level 3(Network layer).

- -a [hostname] –all: This option is used for showing entries of the specified host. If nothing is passed all entries will be displayed.

6. **Nslookup**

Nslookup (stands for "Name Server Lookup") is a useful command for getting information from the DNS server. It is a network administration tool for querying the Domain Name System (DNS) to obtain domain name or IP address mapping or any other specific DNS record. It is also used to troubleshoot DNS-related problems.

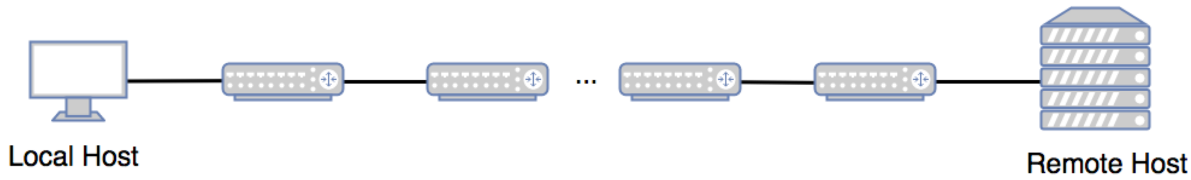
Syntax:

nslookup [option]

7. **tracert**

Traceroute is a widely used command-line utility available in almost all operating systems. It shows you the complete route to a destination address. It also shows the time is taken (or delays) between intermediate routers.

As shown in the below diagram, there are intermediate routers between source and destination.



It sends many packets toward the destination.

8. netstat

The netstat command is used to show network status.
Traditionally, it is used more for problem determination than for performance measurement. However, the netstat command can be used to determine the amount of traffic on the network to ascertain whether performance problems are due to network congestion.
The netstat command displays information regarding traffic on the configured network interfaces, such as the following:

- The address of any protocol control blocks associated with the sockets and the state of all sockets
- The number of packets received, transmitted, and dropped in the communications subsystem
- Cumulative statistics per interface
- Routes and their status

9. systeminfo

Displays detailed configuration information about a computer and its operating system, including operating system configuration, security information, product ID, and hardware properties (such as RAM, disk space, and network cards).
Syntax:
systeminfo [/s <computer> [/u <domain>\<username> [/p <password>]]] [/fo {TABLE | LIST | CSV}] [/nh]

Parameter	Description
/s `<computer>`	Specifies the name or IP address of a remote computer (do not use backslashes). The default is the local computer.
/u `<domain>\<username>`	Runs the command with the account permissions of the specified user account. If **/u** is not specified, this command uses the permissions of the user who is currently logged on to the computer that is issuing the command.
/p `<password>`	Specifies the password of the user account that is specified in the **/u** parameter.
/fo `<format>`	Specifies the output format with one of the following values: • **TABLE** - Displays output in a table. • **LIST** - Displays output in a list. • **CSV** - Displays output in comma-separated values (.csv) format.
/nh	Suppresses column headers in the output. Valid when the **/fo** parameter is set to TABLE or CSV.
/?	Displays help at the command prompt.

Implementation:

Ping

```
C:\Users\tcet>ping www.google.com

Pinging www.google.com [142.250.192.4] with 32 bytes of data:
Reply from 142.250.192.4: bytes=32 time=9ms TTL=117
Reply from 142.250.192.4: bytes=32 time=3ms TTL=117
Reply from 142.250.192.4: bytes=32 time=9ms TTL=117
Reply from 142.250.192.4: bytes=32 time=15ms TTL=117

Ping statistics for 142.250.192.4:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 3ms, Maximum = 15ms, Average = 9ms
```

Ipconfig

```
C:\Users\tcet>ipconfig

Windows IP Configuration


Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::c493:2f0d:3d3:3697%7
   IPv4 Address. . . . . . . . . . . : 175.175.1.109
   Subnet Mask . . . . . . . . . . . : 255.255.0.0
   Default Gateway . . . . . . . . . : 175.175.0.1
```

Hostname

```
C:\Users\tcet>hostname
lab304-30

C:\Users\tcet>getmac
```

Getmac

```
C:\Users\student>getmac

Physical Address    Transport Name
=================== ==========================================================
F8-BC-12-7D-E6-34   \Device\Tcpip_{EEC5AEAB-7A2A-4964-A2DE-B68C0E49C078}
0A-00-27-00-00-03   \Device\Tcpip_{1D414E46-A67A-4C80-B78E-87A6340EE0EE}

C:\Users\student>_
```

Arp –a

```
C:\Users\tcet>arp -a

Interface: 175.175.1.109 --- 0x7
  Internet Address      Physical Address      Type
  175.175.0.1           ec-1d-8b-19-ce-6e     dynamic
  175.175.0.2           d4-76-a0-09-21-68     dynamic
  175.175.1.14          e8-9f-80-6d-ae-63     dynamic
  175.175.1.20          c0-18-03-ba-c1-77     dynamic
  175.175.1.21          60-32-b1-da-f7-11     dynamic
  175.175.1.26          c8-d3-ff-b7-32-56     dynamic
  175.175.1.38          f4-8e-38-79-76-dc     dynamic
  175.175.1.42          88-51-fb-6d-74-59     dynamic
  175.175.1.45          f8-b1-56-be-bb-2e     dynamic
  175.175.1.49          f4-8e-38-80-4a-87     dynamic
  175.175.1.52          f4-8e-38-79-75-c1     dynamic
  175.175.1.54          9c-7b-ef-1c-70-84     dynamic
  175.175.1.58          f4-8e-38-79-72-53     dynamic
  175.175.1.95          f4-8e-38-7a-3c-98     dynamic
  175.175.1.99          80-47-86-63-76-42     dynamic
  175.175.1.104         a0-8c-fd-d5-96-ec     dynamic
  175.175.1.105         04-0e-3c-25-55-ee     dynamic
  175.175.1.110         f4-8e-38-80-75-af     dynamic
  175.175.1.117         78-45-c4-23-28-ce     dynamic
  175.175.1.120         f4-8e-38-7a-3d-c4     dynamic
  175.175.1.127         f8-bc-12-7e-25-88     dynamic
  175.175.1.140         f4-8e-38-77-1d-10     dynamic
  175.175.1.144         a0-8c-fd-d8-c3-94     dynamic
  175.175.1.156         f8-b1-56-bd-fd-30     dynamic
  175.175.1.169         9c-7b-ef-20-34-0e     dynamic
```

Nslookup

```
C:\Users\tcet>nslookup google.com
Server:  dns.google
Address:  8.8.8.8

Non-authoritative answer:
Name:    google.com
Addresses:  2404:6800:4009:823::200e
          142.250.66.14
```

Tracert

```
C:\Users\tcet>tracert google.com

Tracing route to google.com [142.250.76.206]
over a maximum of 30 hops:

  1     1 ms     2 ms     1 ms  175.175.0.1
  2     8 ms     9 ms     6 ms  175.175.0.2
  3     7 ms     8 ms     7 ms  123.252.147.169
  4     7 ms     *        *     10.129.10.230
  5     9 ms     9 ms     7 ms  72.14.210.20
  6    12 ms    12 ms     8 ms  108.170.248.177
  7    21 ms    22 ms    23 ms  142.250.208.149
  8    14 ms    14 ms    13 ms  bom12s10-in-f14.1e100.net [142.250.76.206]

Trace complete.
```

Netstat

```
C:\Users\student>netstat

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    175.175.1.131:7680     lab304-13:53087        TIME_WAIT
  TCP    175.175.1.131:7680     HODIT:51463            TIME_WAIT
  TCP    175.175.1.131:49866    se-in-f188:5228        ESTABLISHED
  TCP    175.175.1.131:52189    a69-192-1-99:https     CLOSE_WAIT
  TCP    175.175.1.131:52222    117.18.232.200:https   CLOSE_WAIT
  TCP    175.175.1.131:52225    204.79.197.222:https   ESTABLISHED
  TCP    175.175.1.131:52454    whatsapp-cdn-shv-02-bom1:https  ESTABLISHED
  TCP    175.175.1.131:52681    a23-54-82-201:https    CLOSE_WAIT
  TCP    175.175.1.131:52682    a23-54-82-201:https    CLOSE_WAIT
  TCP    175.175.1.131:52683    a23-54-82-201:https    CLOSE_WAIT
  TCP    175.175.1.131:52684    a23-54-82-201:https    CLOSE_WAIT
  TCP    175.175.1.131:52874    20.198.119.84:https    ESTABLISHED
  TCP    175.175.1.131:52931    bom12s11-in-f10:https  ESTABLISHED
  TCP    175.175.1.131:53221    DESKTOP-FAQP4RN:ms-do  TIME_WAIT
  TCP    175.175.1.131:53222    175.175.1.50:ms-do     TIME_WAIT
  TCP    175.175.1.131:53226    DESKTOP-F8IPQ07:13111  TIME_WAIT
  TCP    175.175.1.131:53227    dns:https              TIME_WAIT
  TCP    175.175.1.131:53228    whatsapp-cdn-shv-02-bom1:https  TIME_WAIT
  TCP    175.175.1.131:53229    bom05s15-in-f3:https   TIME_WAIT
  TCP    175.175.1.131:53230    bom07s29-in-f10:https  TIME_WAIT
  TCP    175.175.1.131:53233    kul01s09-in-f66:https  TIME_WAIT
  TCP    175.175.1.131:53235    DESKTOP-F8IPQ07:13111  TIME_WAIT
  TCP    175.175.1.131:53236    bom12s04-in-f4:https   TIME_WAIT
  TCP    175.175.1.131:53237    bom12s17-in-f14:https  TIME_WAIT
  TCP    175.175.1.131:53238    bom12s04-in-f4:https   FIN_WAIT_2
  TCP    175.175.1.131:53242    bom12s17-in-f14:https  TIME_WAIT
  TCP    175.175.1.131:53244    dns:https              FIN_WAIT_2
  TCP    175.175.1.131:53245    dns:https              TIME_WAIT
  TCP    175.175.1.131:53247    dns:https              TIME_WAIT
  TCP    175.175.1.131:53249    dns:https              FIN_WAIT_2
  TCP    175.175.1.131:53251    del11s11-in-f2:https   FIN_WAIT_2
  TCP    175.175.1.131:53252    del11s11-in-f2:https   TIME_WAIT
  TCP    175.175.1.131:53253    bom07s30-in-f2:https   TIME_WAIT
  TCP    175.175.1.131:53254    dns:https              TIME_WAIT
```

Systeminfo

```
C:\Users\student>systeminfo

Host Name:                 304-26
OS Name:                   Microsoft Windows 10 Pro
OS Version:                10.0.19044 N/A Build 19044
OS Manufacturer:           Microsoft Corporation
OS Configuration:          Standalone Workstation
OS Build Type:             Multiprocessor Free
Registered Owner:          student
Registered Organization:
Product ID:                00331-10000-00001-AA182
Original Install Date:     5/11/2022, 9:43:05 AM
System Boot Time:          2/17/2023, 9:41:57 AM
System Manufacturer:       Dell Inc.
System Model:              OptiPlex 3020
System Type:               x64-based PC
Processor(s):              1 Processor(s) Installed.
                           [01]: Intel64 Family 6 Model 60 Stepping 3 GenuineIntel ~1500 Mhz
BIOS Version:              Dell Inc. A03, 4/14/2014
Windows Directory:         C:\Windows
System Directory:          C:\Windows\system32
Boot Device:               \Device\HarddiskVolume1
System Locale:             en-us;English (United States)
Input Locale:              en-us;English (United States)
Time Zone:                 (UTC+05:30) Chennai, Kolkata, Mumbai, New Delhi
Total Physical Memory:     4,015 MB
Available Physical Memory: 556 MB
Virtual Memory: Max Size:  5,039 MB
Virtual Memory: Available: 1,352 MB
Virtual Memory: In Use:    3,687 MB
Page File Location(s):     C:\pagefile.sys
Domain:                    WORKGROUP
Logon Server:              \\304-26
Hotfix(s):                 6 Hotfix(s) Installed.
                           [01]: KB5022502
                           [02]: KB5003791
                           [03]: KB5013942
                           [04]: KB5011352
                           [05]: KB5014032
                           [06]: KB5014035
Network Card(s):           2 NIC(s) Installed.
                           [01]: Realtek PCIe GbE Family Controller
                                 Connection Name: Ethernet
```

Result and Discussion: In this experiment, we implemented different commands of windows and Linux. After completing the experiment, we are able to use and understand basic networking commands of windows and Linux.

Learning Outcomes: The student should have the ability to design & implement product cipher using Substitution and Transposition Cipher

LO1: To describe & understand about windows and Linux commands

LO2: To implement commands of windows and Linux on command prompt.

Course Outcomes: Upon completion of the course students will be able to understand & implement windows and Linux commands.

Conclusion: In this experiment, we implemented different commands of windows and Linux. After completing the experiment, we are able to use and understand basic networking commands of windows and Linux.

For Faculty Use

Correction Parameters	**Formative Assessment [40%]**	**Timely completion of Practical [40%]**	**Attendance / Learning Attitude [20%]**	
Marks Obtained				

Experiment no.3: - Diffie Hellman algorithms

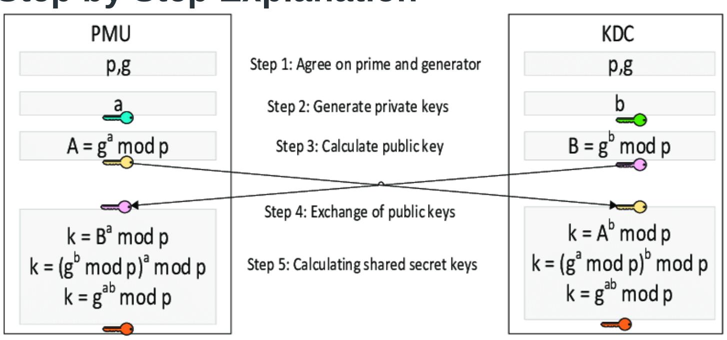
Aim Design and implement of a Secret Key for sender and receiver using Diffie Hellman algorithms

Learning Objective: Student should be able to design and implementation of a Secret Key for sender and receiver using Diffie Hellman algorithms.

Tools: C/C++/Java/Python or any computational software
Theory:
Diffie-Hellman algorithm
The Diffie-Hellman algorithm is being used to establish a shared secret that can be used for secret communications while exchanging data over a public network using the elliptic curve to generate points and get the secret key using the parameters.

- For the sake of simplicity and practical implementation of the algorithm, we will consider only 4 variables, one prime P and G (a primitive root of P) and two private values a and b.
- P and G are both publicly available numbers. Users (say Alice and Bob) pick private values a and b and they generate a key and exchange it publicly. The opposite person receives the key and that generates a secret key, after which they have the same secret key to encrypt.

Step by Step Explanation



Example:

Step 1: Alice and Bob get public numbers P = 23, G = 9
Step 2: Alice selected a private key a = 4 and
Bob selected a private key b = 3
Step 3: Alice and Bob compute public values
Alice: x = (9^4 mod 23) = (6561 mod 23) = 6
Bob: y = (9^3 mod 23) = (729 mod 23) = 16
Step 4: Alice and Bob exchange public numbers
Step 5: Alice receives public key y =16 and
Bob receives public key x = 6
Step 6: Alice and Bob compute symmetric keys
Alice: $k_a = y^a$ mod p = 65536 mod 23 = 9
Bob: $k_b = x^b$ mod p = 216 mod 23 = 9
Step 7: 9 is the shared secret.
Program: -

```
from random import randint
if __name__ == '__main__':
        P = int(input("Enter the Prime Number:- "))
        G = int(input("Enter the G value :- "))
        print('\nThe Value of P is :%d'%(P))
```

```
print('The Value of G is :%d'%(G))
a = 4
print('\nThe Private Key a for Alice is :%d'%(a))
x = int(pow(G,a,P))
b = 3
print('The Private Key b for Bob is :%d'%(b))
y = int(pow(G,b,P))
ka = int(pow(y,a,P))
kb = int(pow(x,b,P))
print('\nSecret key for the Alice is : %d'%(ka))
print('Secret Key for the Bob is : %d'%(kb))
```

Output: -

```
Enter the Prime Number:- 29
Enter the G value :- 7
The Value of P is :29
The Value of G is :7

The Private Key a for Alice is :4
The Private Key b for Bob is :3

Secret key for the Alice is : 16
Secret Key for the Bob is : 16
>
```

Applications:

Forward Secrecy

Protocols that attain forward secrecy create new key pairs for each session and cancel them at the end of the session. For such protocols, the Diffie-Hellman key exchange is a good choice because of its fast key generation

Password-Authenticated Key Agreement

When Joy and Happy share a password, they may use DH's password-authenticated key agreement to avoid man-in-the-middle attacks.

Result and Discussion:

The Diffie-Hellman is used to set up a shared secret that can be used for secret communication while exchanging data across a public channel using this elliptic curve to generate points and get the secret key using the parameters. ECC (Elliptic Curve Cryptography) is an address to public-key cryptography. It is based on the algebraic structure of elliptical curves over finite fields. The DH key exchange method allows the two parties that have zero knowledge of each other to together set up a shared secret over an insecure (public) channel.

Learning Outcomes: The student should have the ability to design & implement a Secret Key for sender and receiver using Diffie Hellman algorithms.

LO1: To describe & understand about Diffie Hellman algorithms

LO2: To implement Diffie Hellman algorithms

Course Outcomes: Upon completion of the course students will be able to understand & implement Diffie Hellman algorithms.

Conclusion: In this experiment, we implemented Diffie Hellman algorithms and understand the step by step procedure. The Diffie-Hellman Algorithm is a secure way of cryptographic keys exchange across a public channel

For Faculty Use

Correction Parameters	Formative Assessment [40%]	Timely completion of Practical [40%]	Attendance / Learning Attitude [20%]	

Marks Obtained				

Experiment no.4

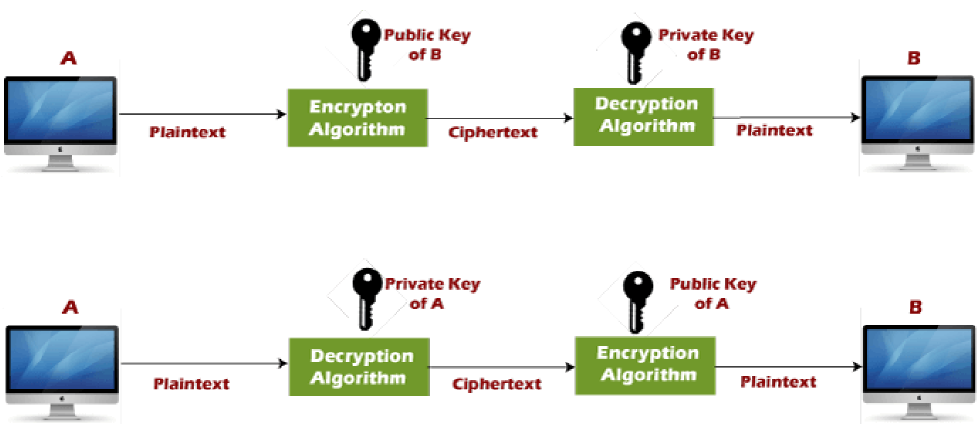
Aim :Design and implement RSA algorithm
Learning Objective: Student should be able to understand and implement the RSA algorithm.
Tools: C/C++/Java/Python or any computational software

Theory:
RSA encryption algorithm is a type of public-key encryption algorithm. To better understand RSA, lets first understand what is public-key encryption algorithm.
Public key encryption algorithm:
Public Key encryption algorithm is also called the Asymmetric algorithm. Asymmetric algorithms are those algorithms in which sender and receiver use different keys for encryption and decryption. Each sender is assigned a pair of keys:

Public key

Private key

The **Public key** is used for encryption, and the **Private Key** is used for decryption. Decryption cannot be done using a public key. The two keys are linked, but the private key cannot be derived from the public key. The public key is well known, but the private key is secret and it is known only to the user who owns the key. It means that everybody can send a message to the user using user's public key. But only the user can decrypt the message using his private key.
The Public key algorithm operates in the following manner:

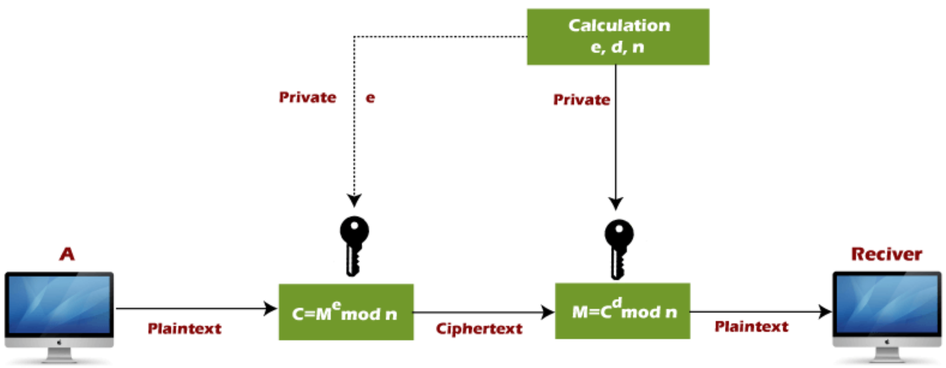


Encryption/decryption using public/private keys

The data to be sent is encrypted by sender **A** using the public key of the intended receiver
B decrypts the received ciphertext using its private key, which is known only to B. B replies to A encrypting its message using A's public key.
A decrypts the received ciphertext using its private key, which is known only to him.
RSA encryption algorithm:
Select two large prime numbers, p and q.
Multiply these numbers to find n = p x q, where n is called the modulus for encryption and decryption.
Choose a number e less than n, such that n is relatively prime to (p - 1) x (q -1). It means that e and (p - 1) x (q - 1) have no common factor except 1. Choose "e" such that 1<e < ϕ (n),e is prime to ϕ (n),
gcd (e,d(n)) =1
If n = p x q, then the public key is <e, n>. A plaintext message m is encrypted using public key <e, n>. To find ciphertext from the plain text following formula is used to get ciphertext C. \C = m^e mod n
Here, m must be less than n. A larger message (>n) is treated as a concatenation of messages, each of which is encrypted separately.

To determine the private key, we use the following formula to calculate the d such that:
$D_e \bmod \{(p - 1) \times (q - 1)\} = 1$
Or $D_e \bmod \phi(n) = 1$
The private key is <d, n>. A ciphertext message c is decrypted using private key <d, n>. To calculate plain text m from the ciphertext c following formula is used to get plain text m. $m = c^d \bmod n$

RSA is the most common public-key algorithm, named after its inventors **Rivest, Shamir, and Adelman (RSA).**



RSA

Program:

```
import math

def gcd(a, h):
    temp = 0
    while(1):
        temp = a % h
        if (temp == 0):
            return h
        a = h
        h = temp
p = 11
q = 13
n = p*q
e = 2
phi = (p-1)*(q-1)
while (e < phi):
    if(gcd(e, phi) == 1):
        break
    else:
        e = e+1
k = 2
d = (1 + (k*phi))/e
msg = 12.0

print("Message data = ", msg)

c = pow(msg, e)
c = math.fmod(c, n)
print("Encrypted data = ", c)

m = pow(c, d)
m = math.fmod(m, n)
print("Original Message Sent = ", m)
```

Output:

```
Message data =  12.0
Encrypted data =  12.0
Original Message Sent =  17.0
>
```

Learning Outcomes: The student should have the ability to design & implement RSA algorithm using python
LO1: To understand the RSA algorithm.

LO2: To implement RSA algorithm.

Course Outcomes: Upon completion of the course students will be able to understand & implement the RSA algorithm.

Conclusion:
In this experiment we learned about the RSA algorithm, implemented it using python and even understood how to solve problems related to it.

For Faculty Use

Correction Parameters	Formative Assessment [40%]	Timely completion of Practical [40%]	Attendance / Learning Attitude [20%]	
Marks Obtained				

Experiment no.5

Aim: To study and understand hashing algorithm.

Learning Objective: Student should be able to understand about hashing function and its algorithm like MD5,SHA etc.

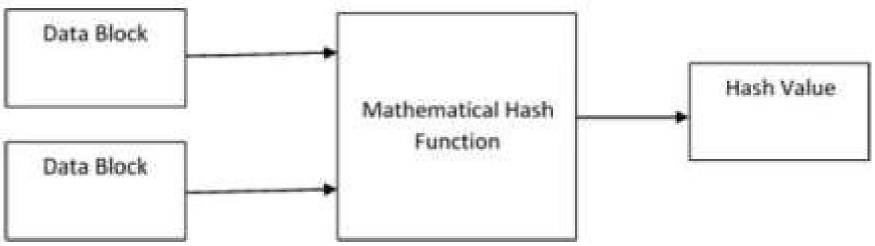
Tools: C++/Java/Python

Theory:

Hash functions are extremely useful and appear in almost all information security applications.A hash function is a mathematical function that converts a numerical input value into another compressed numerical value. The input to the hash function is of arbitrary length but output is always of fixed length.Values returned by a hash function are called message digest or simply hash values.

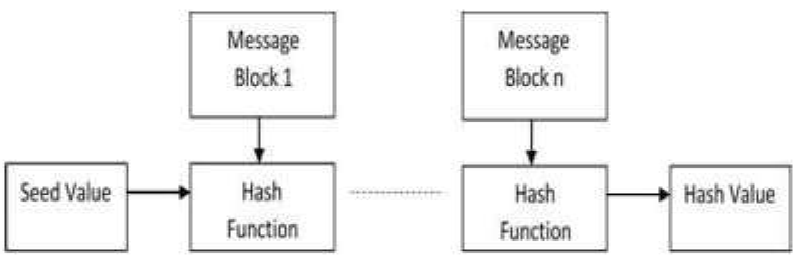
At the heart of a hashing is a mathematical function that operates on two fixed-size blocks of data to create a hash code. This hash function forms the part of the hashing algorithm.

The size of each data block varies depending on the algorithm. Typically the block sizes are from 128 bits to 512 bits. The following illustration demonstrates hash function –



Hashing algorithm involves rounds of above hash function like a block cipher. Each round takes an input of a fixed size, typically a combination of the most recent message block and the output of the last round.

This process is repeated for as many rounds as are required to hash the entire message. Schematic of hashing algorithm is depicted in the following illustration –

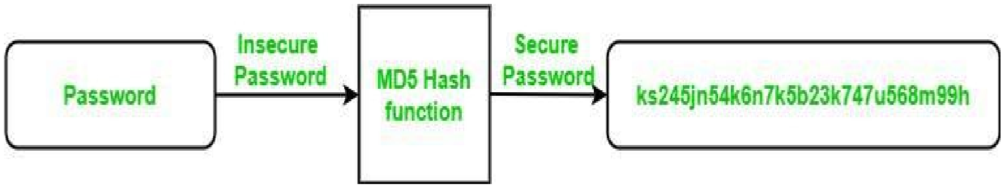


Since, the hash value of first message block becomes an input to the second hash operation, output of which alters the result of the third operation, and so on. This effect, known as an **avalanche** effect of hashing.

MD5 is a cryptographic hash function algorithm that takes the message as input of any length and changes it into a fixed-length message of 16 bytes. MD5 algorithm stands for the **message-digest algorithm**. MD5 was developed as an improvement of MD4, with advanced security purposes. The output of MD5 (Digest size) is always **128 bits. MD5** was developed in 1991 by **Ronald Rivest.**

Use Of MD5 Algorithm:

- It is used for file authentication.
- In a web application, it is used for security purposes. e.g. Secure password of users etc. 🎬Using this algorithm, We can store our password in 128 bits format.



MD5 Algorithm

Implementation :

Code :

```
import math
import hashlib
rotate_by = [7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22,
          5,  9, 14, 20, 5,  9, 14, 20, 5,  9, 14, 20, 5,  9, 14, 20,
          4, 11, 16, 23, 4, 11, 16, 23, 4, 11, 16, 23, 4, 11, 16, 23,
          6, 10, 15, 21, 6, 10, 15, 21, 6, 10, 15, 21, 6, 10, 15, 21]
constants = [int(abs(math.sin(i+1)) * 4294967296) & 0xFFFFFFFF for i in range(64)]
def pad(msg):
   msg_len_in_bits = (8*len(msg)) & 0xffffffffffffffff
   msg.append(0x80)
   while len(msg)%64 != 56:
      msg.append(0)
      msg += msg_len_in_bits.to_bytes(8, byteorder='little')
   return msg
init_MDBuffer = [0x67452301, 0xefcdab89, 0x98badcfe, 0x10325476]
def leftRotate(x, amount):
   x &= 0xFFFFFFFF
   return (x << amount | x >> (32-amount)) & 0xFFFFFFFF
def processMessage(msg):
 init_temp = init_MDBuffer[:]
 for offset in range(0, len(msg), 64):
    A, B, C, D = init_temp
    block = msg[offset : offset+64]
    for i in range(64):
      if i < 16:
          func = lambda b, c, d: (b & c) | (~b & d)

          index_func = lambda i: i
      elif i >= 16 and i < 32:
         func = lambda b, c, d: (d & b) | (~d & c)

         index_func = lambda i: (5*i + 1)%16

      elif i >= 32 and i < 48:
         func = lambda b, c, d: b ^ c ^ d
         index_func = lambda i: (3*i + 5)%16
      elif i >= 48 and i < 64:
         func = lambda b, c, d: c ^ (b | ~d)

         index_func = lambda i: (7*i)%16
      F = func(B, C, D)
      G = index_func(i)
      to_rotate = A + F + constants[i] + int.from_bytes(block[4*G : 4*G + 4], byteorder='little')
      newB = (B + leftRotate(to_rotate, rotate_by[i])) & 0xFFFFFFFF
      A, B, C, D = D, newB, B, C
      for i, val in enumerate([A, B, C, D]):
         init_temp[i] += val
         init_temp[i] &= 0xFFFFFFFF
 return sum(buffer_content<<(32*i) for i, buffer_content in enumerate(init_temp))
def MD_to_hex(digest):
   raw = digest.to_bytes(16, byteorder='little')
   return '{:032x}'.format(int.from_bytes(raw, byteorder='big'))
def md5(msg):
   msg = bytearray(msg, 'ascii')
   msg = pad(msg)
   processed_msg = processMessage(msg)
   message_hash = MD_to_hex(processed_msg)
   print("Hash Value: ", message_hash)
def hash_value(msg):
   hashvalue = hashlib.md5(msg.encode()).hexdigest()
   print("Hash value using hashlib: ", hashvalue)
if __name__ == '__main__':
   print ("Enter the message to be hashed: ")
   message = input()
   md5(message)
   hash_value(message)
```

Output:

Enter the message to be hashed: Thakur College
Hash Value: 4082e278a88b6458bab5c705b0b07b7e

Hash value
using hashlib: 3141df0027caeb3659c237145ed6b404

Result and Discussion : In this experiment we successfully understood the concept of Hashing function algorithm and implemented the MD5 algorithm using python.

Learning Outcomes: The student will be able to

LO1: Understand the Concept of Hashing Functions

LO2: Understand the Steps for implementing the hashing function algorithm.

Course Outcomes: Upon completion of the course students will be able to study the various network reconnaissance tools & how to use them to gather primary network information.

Conclusion: We have implemented hashing algorithms and understood the concept of hash value algorithms

For Faculty Use

Correction Parameter s	Formative Assessmen t [40%]	Timely completion of Practical [40%]	Attendance / Learning Attitude [20%]	
Marks Obtained				

CSS

Experiment 6

Aim: Perform various attacks using Burp Suite for security testing of web applications

Tools: Burp Suite

Theory:

Burp Suite: Burp Suite is an integrated platform/graphical tool for performing security testing of web applications. Its various tools work seamlessly together to support the entire testing process, from initial mapping and analysis of an application's attack surface, through to finding and exploiting security vulnerabilities.

Burp or Burp Suite is a set of tools used for penetration testing of web applications. It is developed by the company named Portswigger, which is also the alias of its founder Dafydd Stuttard. BurpSuite aims to be an all in one set of tools and its capabilities can be enhanced by installing add-ons that are called BApps.

Attacks performed using Burp Suite are as follows:

- **Brute Force Attack:**

  A brute force attack is a hacking method that uses trial and error to crack passwords, login credentials, and encryption keys. It is a simple yet reliable tactic for gaining unauthorized access to individual accounts and organizations' systems and networks. The hacker tries multiple usernames and passwords, often using a computer to test a wide range of combinations, until they find the correct login information.

  The name "brute force" comes from attackers using excessively forceful attempts to gain access to user accounts. Despite being an old cyberattack method, brute force attacks are tried and tested and remain a popular tactic with hackers.

  Types of Brute Force Attack:

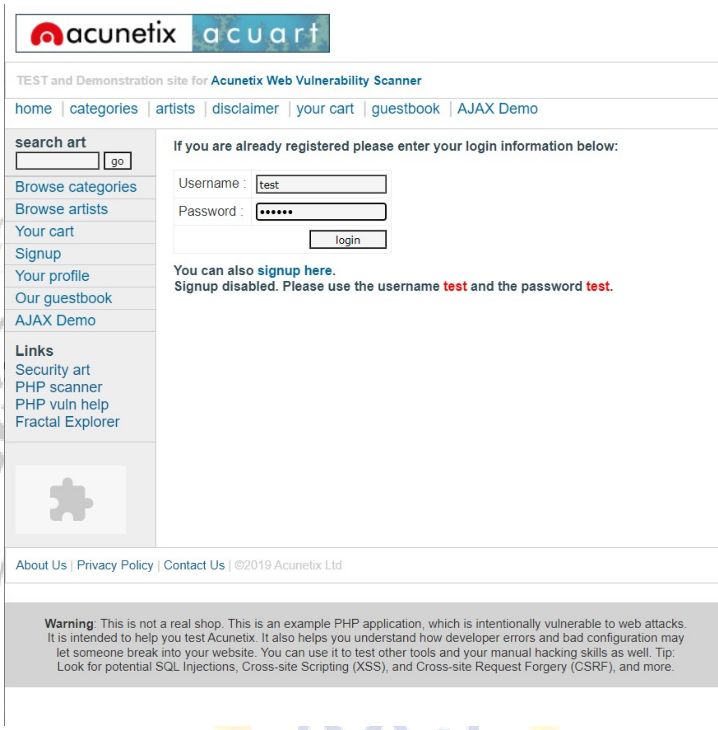
  - Simple Brute Force Attack
  - Dictionary Attack
  - Hybrid Brute Force Attack
  - Reverse Brute Force Attacks
  - Credential Stuffing

- **OTP Attack:**

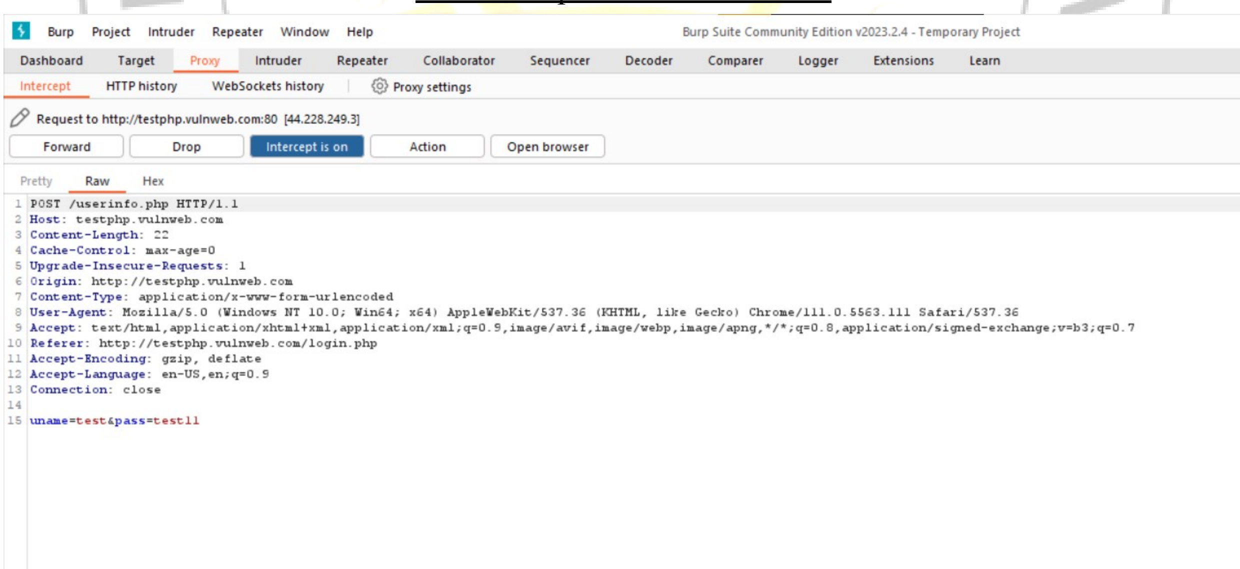
  We work with phone numbers. We send one-time PINs (OTP) through SMS, voice, etc. to phone numbers so users can recite the OTP back to us as proof that they have access to/own the phone, which is a form of 2-Factor Authentication (2FA).

  Each phone verification attempt incurs cost as it involves sending a OTP through short message (SMS) or voice. Attackers can rack up phone verification bill by requesting OTPs with no intention of use. We term this as a resource exhaustion attack

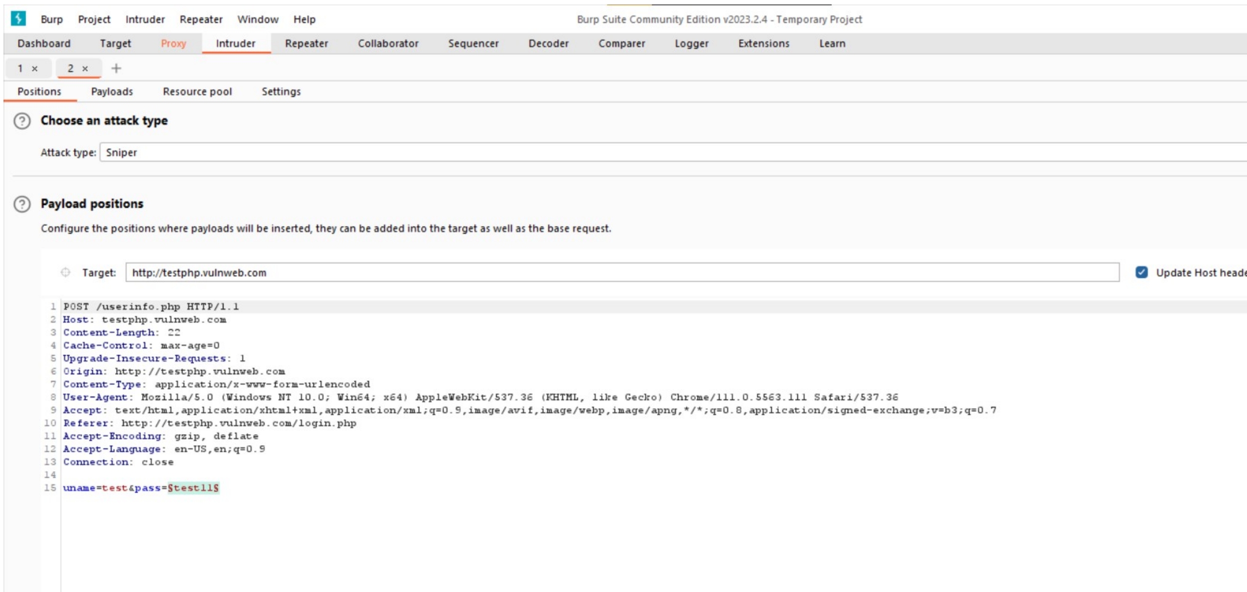
Implementation:



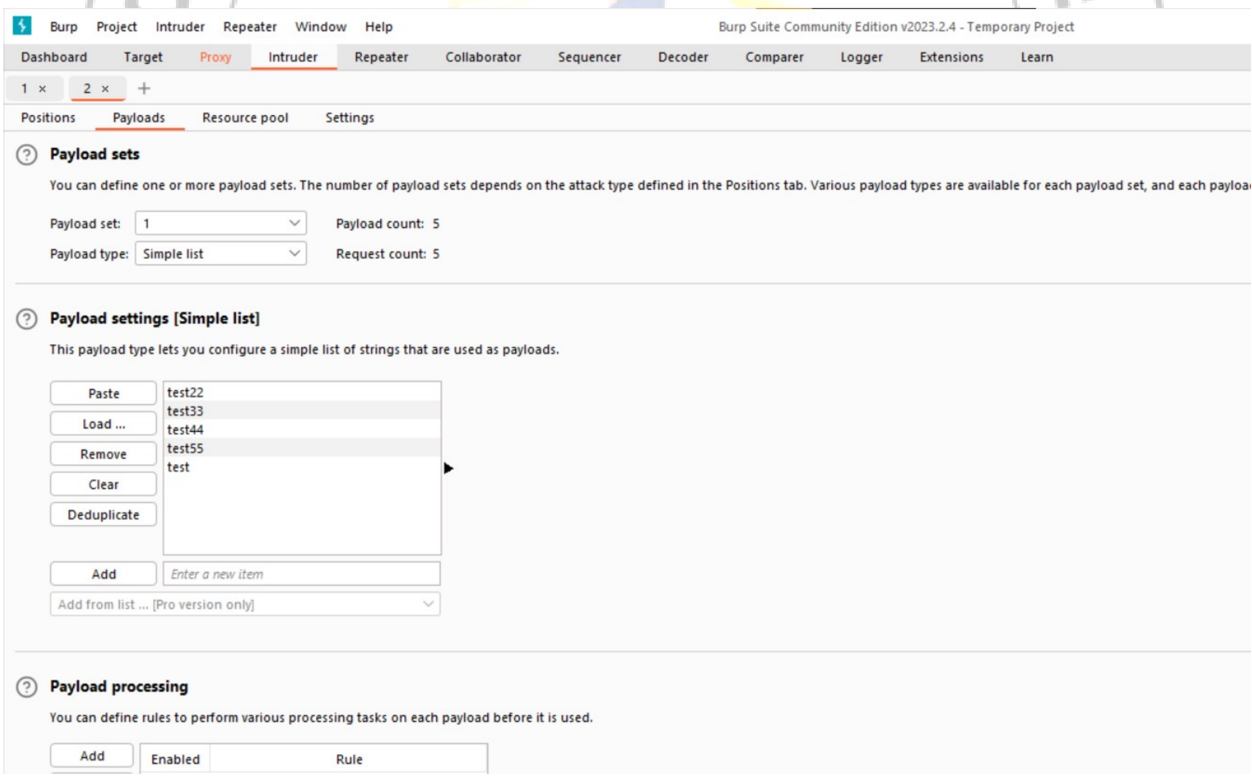
Website to perform the attack on



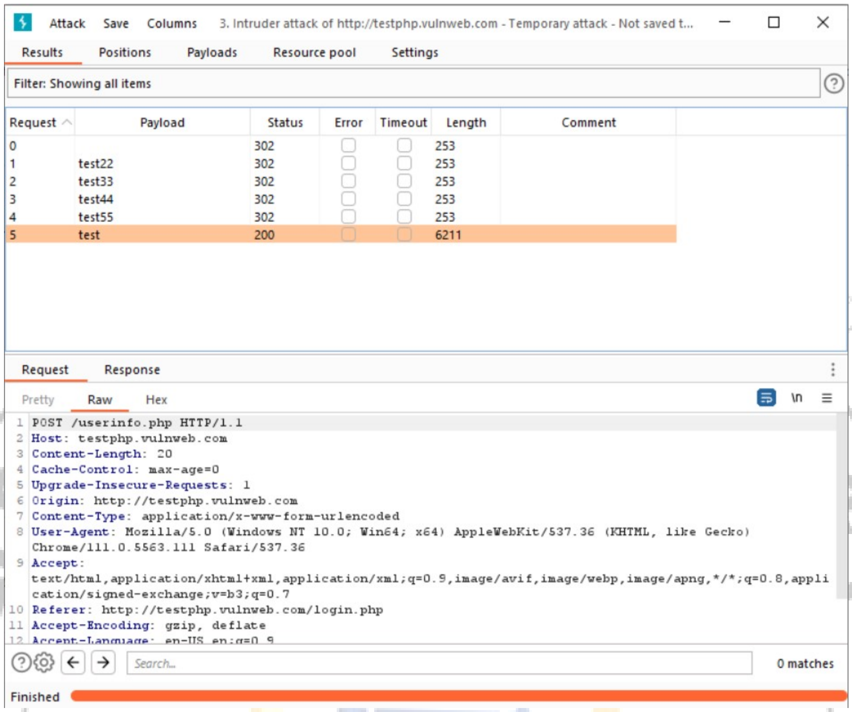
POST Method details after typing credentials listing attributes and variables



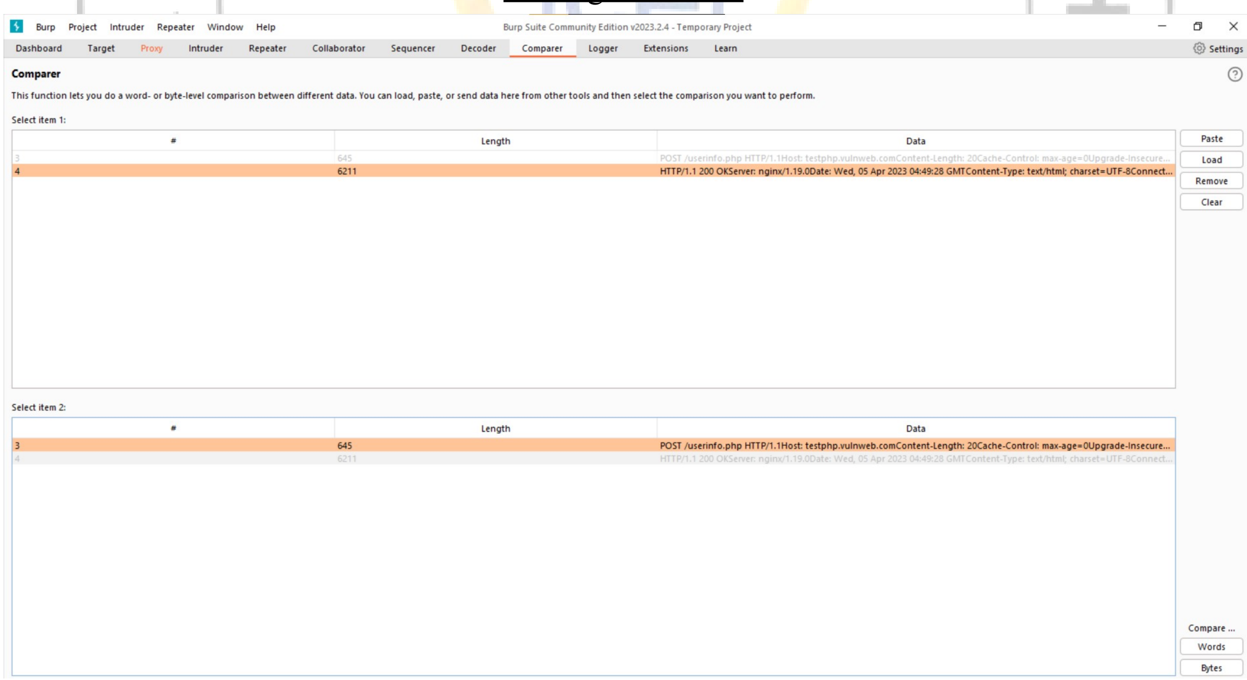
Adding $ to the variables that need changing



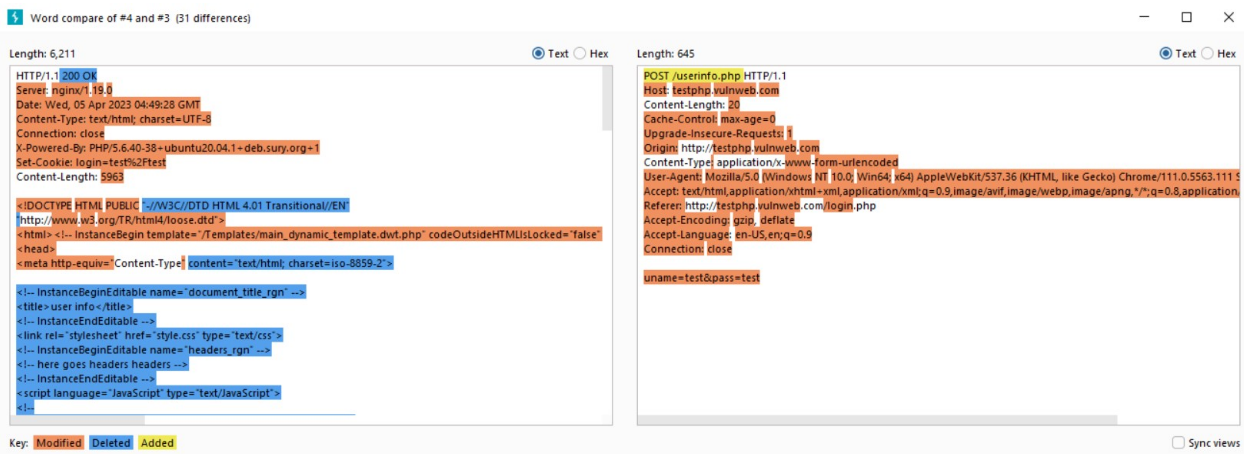
Adding list of strings as payloads



<u>Starting the Attack</u>



<u>Comparing the results with words for the correct and faulty password</u>



Results after comparing

Conclusion: We learned about the brute force attack and how it can be implemented through the Burp Suite using its various tools. We executed the attack for an attack and compared the results for the same. Several concepts related to the attacks were revised while performing the experiment.

For Faculty Use

Correction Parameters	Formative Assessment [40%]	Timely completion of Practical [40%]	Attendance / Learning Attitude [20%]	
Marks Obtained				

Experiment 7: DOS Attack

Aim: Write the implementation of DOS attack

Theory:

A denial-of-service (DoS) attack is a type of cyber attack in which a malicious actor aims to render a computer or other device unavailable to its intended users by interrupting the device's normal functioning. DoS attacks typically function by overwhelming or flooding a targeted machine with requests until normal traffic is unable to be processed, resulting in denial-of-service to addition users. A DoS attack is characterized by using a single computer to launch the attack.

A distributed denial-of-service (DDoS) attack is a type of DoS attack that comes from many distributed sources, such as a botnet DDoS attack.

The primary focus of a DoS attack is to oversaturate the capacity of a targeted machine, resulting in denial-of-service to additional requests. The multiple attack vectors of DoS attacks can be grouped by their similarities.

An attack type in which a memory buffer overflow can cause a machine to consume all available hard disk space, memory, or CPU time. This form of exploit often results in sluggish behavior, system crashes, or other deleterious server behaviors, resulting in denial-of-service.

A few common historic DoS attacks include:

- Smurf attack - a previously exploited DoS attack in which a malicious actor utilizes the broadcast address of vulnerable network by sending spoofed packets, resulting in the flooding of a targeted IP address.

- Ping flood - this simple denial-of-service attack is based on overwhelming a target with ICMP (ping) packets. By inundating a target with more pings than it is able to respond to efficiently, denial-of-service can occur. This attack can also be used as a DDoS attack.

- Ping of Death - often conflated with a ping flood attack, a ping of death attack involves sending a malformed packet to a targeted machine, resulting in deleterious behavior such as system crashes.

How can you tell if a computer is experiencing a DoS attack?

While it can be difficult to separate an attack from other network connectivity errors or heavy bandwidth consumption, some characteristics may indicate an attack is underway.

Indicators of a DoS attack include:

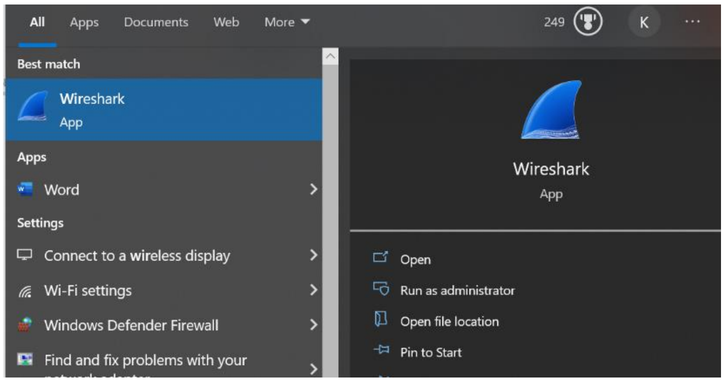
- Atypically slow network performance such as long load times for files or websites
- The inability to load a particular website such as your web property
- A sudden loss of connectivity across devices on the same network

The distinguishing difference between DDoS and DoS is the number of connections utilized in the attack. Some DoS attacks, such as “low and slow” attacks like Slowloris, derive their power in the simplicity and minimal requirements needed to them be effective.

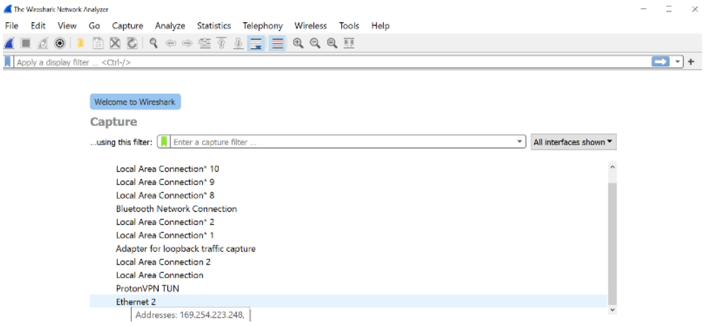
DoS utilizes a single connection, while a DDoS attack utilizes many sources of attack traffic, often in the form of a botnet. Generally speaking, many of the attacks are fundamentally similar and can be attempted using one more many sources of malicious traffic. Learn how Cloudflare's DDoS protection stops denial-of-service attacks.

<u>Implementation:</u>

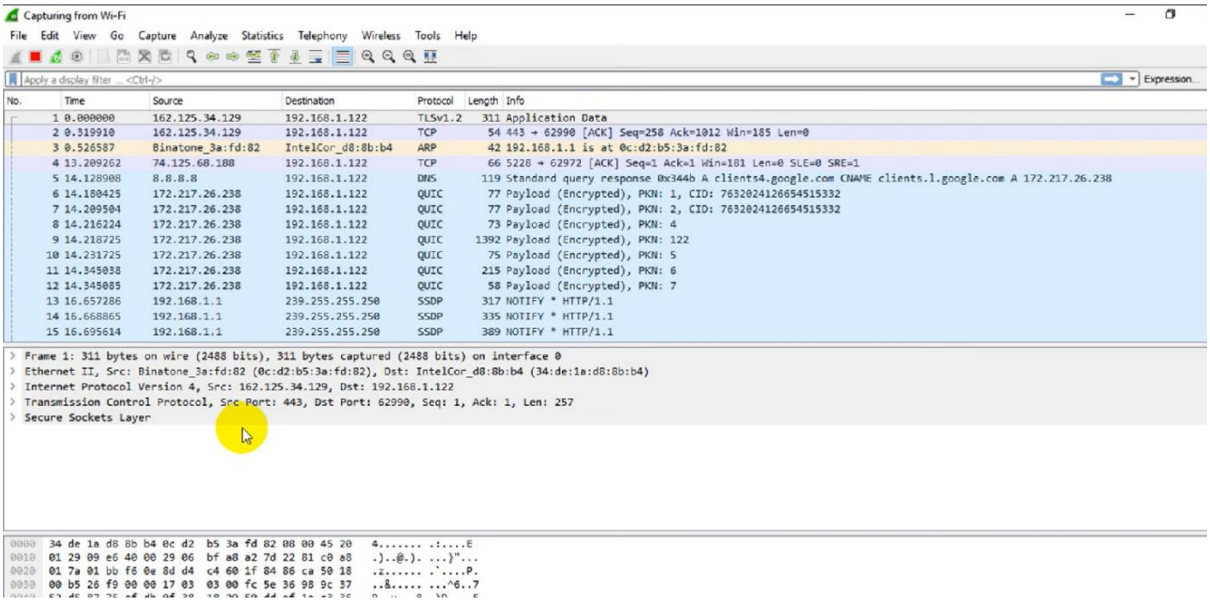
Step 1



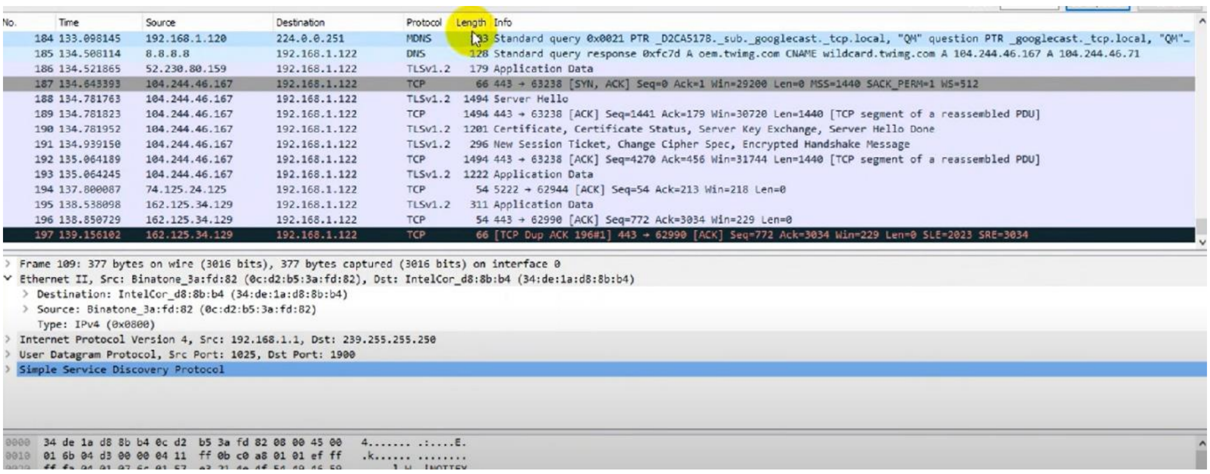
Step 2


The Wireshark Network Analyzer
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Welcome to Wireshark
Capture
...using this filter:
Enter a capture filter ...
All interfaces shown
Local Area Connection* 10
Local Area Connection* 9
Local Area Connection* 8
Bluetooth Network Connection
Local Area Connection* 2
Local Area Connection* 1
Adapter for loopback traffic capture
Local Area Connection 2
Local Area Connection
ProtonVPN TUN
Ethernet 2
Addresses: 169.254.223.248,

Step 3


Capturing from Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...
No. Time Source Destination Protocol Length Info
1 0.000000 162.125.34.129 192.168.1.122 TLSv1.2 311 Application Data
2 0.319910 162.125.34.129 192.168.1.122 TCP 54 443 → 62990 [ACK] Seq=258 Ack=1012 Win=185 Len=0
3 0.526587 Binatone_3a:fd:82 IntelCor_d8:8b:b4 ARP 42 192.168.1.1 is at 0c:d2:b5:3a:fd:82
4 13.209262 74.125.68.188 192.168.1.122 TCP 66 5228 → 62972 [ACK] Seq=1 Ack=1 Win=181 Len=0 SLE=0 SRE=1
5 14.128908 8.8.8.8 192.168.1.122 DNS 119 Standard query response 0x344b A clients4.google.com CNAME clients.l.google.com A 172.217.26.238
6 14.180425 172.217.26.238 192.168.1.122 QUIC 77 Payload (Encrypted), PKN: 1, CID: 7632024126654515332
7 14.209504 172.217.26.238 192.168.1.122 QUIC 77 Payload (Encrypted), PKN: 2, CID: 7632024126654515332
8 14.216224 172.217.26.238 192.168.1.122 QUIC 73 Payload (Encrypted), PKN: 4
9 14.218725 172.217.26.238 192.168.1.122 QUIC 1392 Payload (Encrypted), PKN: 122
10 14.231725 172.217.26.238 192.168.1.122 QUIC 75 Payload (Encrypted), PKN: 5
11 14.345038 172.217.26.238 192.168.1.122 QUIC 215 Payload (Encrypted), PKN: 6
12 14.345085 172.217.26.238 192.168.1.122 QUIC 58 Payload (Encrypted), PKN: 7
13 16.657286 192.168.1.1 239.255.255.250 SSDP 317 NOTIFY * HTTP/1.1
14 16.668865 192.168.1.1 239.255.255.250 SSDP 335 NOTIFY * HTTP/1.1
15 16.695614 192.168.1.1 239.255.255.250 SSDP 389 NOTIFY * HTTP/1.1
Frame 1: 311 bytes on wire (2488 bits), 311 bytes captured (2488 bits) on interface 0
Ethernet II, Src: Binatone_3a:fd:82 (0c:d2:b5:3a:fd:82), Dst: IntelCor_d8:8b:b4 (34:de:1a:d8:8b:b4)
Internet Protocol Version 4, Src: 162.125.34.129, Dst: 192.168.1.122
Transmission Control Protocol, Src Port: 443, Dst Port: 62990, Seq: 1, Ack: 1, Len: 257
Secure Sockets Layer
0000 34 de 1a d8 8b b4 0c d2 b5 3a fd 82 08 00 45 20 4.......:....E
0010 01 29 09 e6 40 00 29 06 bf a8 a2 7d 22 81 c0 a8 .)..@.)....}"...
0020 01 7a 01 bb f6 0e 8d d4 c4 60 1f 84 86 ca 50 18 .z.......`....P.
0030 00 b5 26 f9 00 00 17 03 03 00 fc 5e 36 98 9c 37 ..&........^6..7

Step 4


No. Time Source Destination Protocol Length Info
184 133.098145 192.168.1.120 224.0.0.251 MDNS 133 Standard query 0x0021 PTR _D2CA5178._sub._googlecast._tcp.local, "QM" question PTR _googlecast._tcp.local, "QM"...
185 134.508114 8.8.8.8 192.168.1.122 DNS 128 Standard query response 0xfc7d A oem.twimg.com CNAME wildcard.twimg.com A 104.244.46.167 A 104.244.46.71
186 134.521865 52.230.80.159 192.168.1.122 TLSv1.2 179 Application Data
187 134.643393 104.244.46.167 192.168.1.122 TCP 66 443 → 63238 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1440 SACK_PERM=1 WS=512
188 134.781763 104.244.46.167 192.168.1.122 TLSv1.2 1494 Server Hello
189 134.781823 104.244.46.167 192.168.1.122 TCP 1494 443 → 63238 [ACK] Seq=1441 Ack=179 Win=30720 Len=1440 [TCP segment of a reassembled PDU]
190 134.781952 104.244.46.167 192.168.1.122 TLSv1.2 1201 Certificate, Certificate Status, Server Key Exchange, Server Hello Done
191 134.939150 104.244.46.167 192.168.1.122 TLSv1.2 296 New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
192 135.064189 104.244.46.167 192.168.1.122 TCP 1494 443 → 63238 [ACK] Seq=4270 Ack=456 Win=31744 Len=1440 [TCP segment of a reassembled PDU]
193 135.064245 104.244.46.167 192.168.1.122 TLSv1.2 1222 Application Data
194 137.800087 74.125.24.125 192.168.1.122 TCP 54 5222 → 62944 [ACK] Seq=54 Ack=213 Win=218 Len=0
195 138.538098 162.125.34.129 192.168.1.122 TLSv1.2 311 Application Data
196 138.850729 162.125.34.129 192.168.1.122 TCP 54 443 → 62990 [ACK] Seq=772 Ack=3034 Win=229 Len=0
197 139.156102 162.125.34.129 192.168.1.122 TCP 66 [TCP Dup ACK 196#1] 443 → 62990 [ACK] Seq=772 Ack=3034 Win=229 Len=0 SLE=2023 SRE=3034
Frame 109: 377 bytes on wire (3016 bits), 377 bytes captured (3016 bits) on interface 0
Ethernet II, Src: Binatone_3a:fd:82 (0c:d2:b5:3a:fd:82), Dst: IntelCor_d8:8b:b4 (34:de:1a:d8:8b:b4)
Destination: IntelCor_d8:8b:b4 (34:de:1a:d8:8b:b4)
Source: Binatone_3a:fd:82 (0c:d2:b5:3a:fd:82)
Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 192.168.1.1, Dst: 239.255.255.250
User Datagram Protocol, Src Port: 1025, Dst Port: 1900
Simple Service Discovery Protocol
0000 34 de 1a d8 8b b4 0c d2 b5 3a fd 82 08 00 45 00 4.......:....E.
0010 01 6b 04 d3 00 00 04 11 ff 0b c0 a8 01 01 ef ff .k..............

Step 5

Conclusion:

For Faculty Use:

Correction Parameters	Formative Assessment [40%]	Timely completion of Practical [40%]	Attendance/ Learning Attitude [20%]	
Marks Obtained				

CSS

Experiment 8

Aim: 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

Objectives:

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

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.

Wireshark is the most often-used packet sniffer in the world. Like any other packet sniffer, Wireshark does three things:

1. **Packet Capture:** Wireshark listens to a network connection in real time and then grabs entire streams of traffic – quite possibly tens of thousands of packets at a time.
2. **Filtering:** Wireshark is capable of slicing and dicing all of this random live data using filters. By applying a filter, you can obtain just the information you need to see.
3. **Visualization:** Wireshark, like any good packet sniffer, allows you to dive right into the very middle of a network packet. It also allows you to visualize entire conversations and network streams.

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.

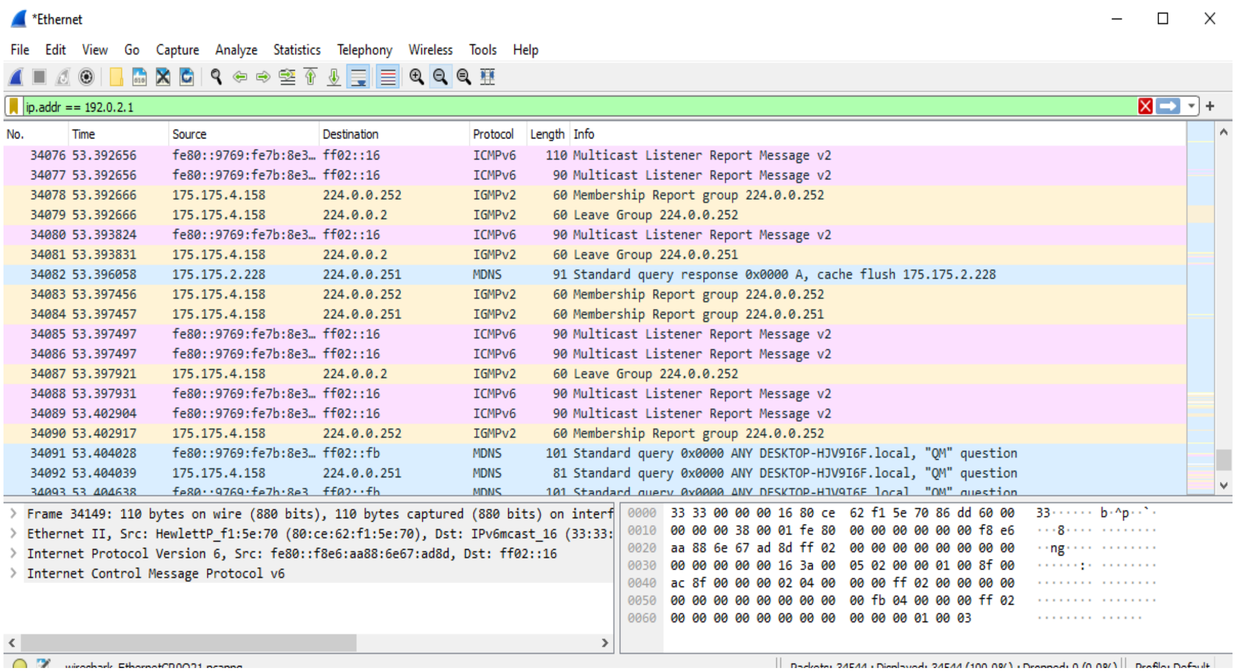
Capturing Packets

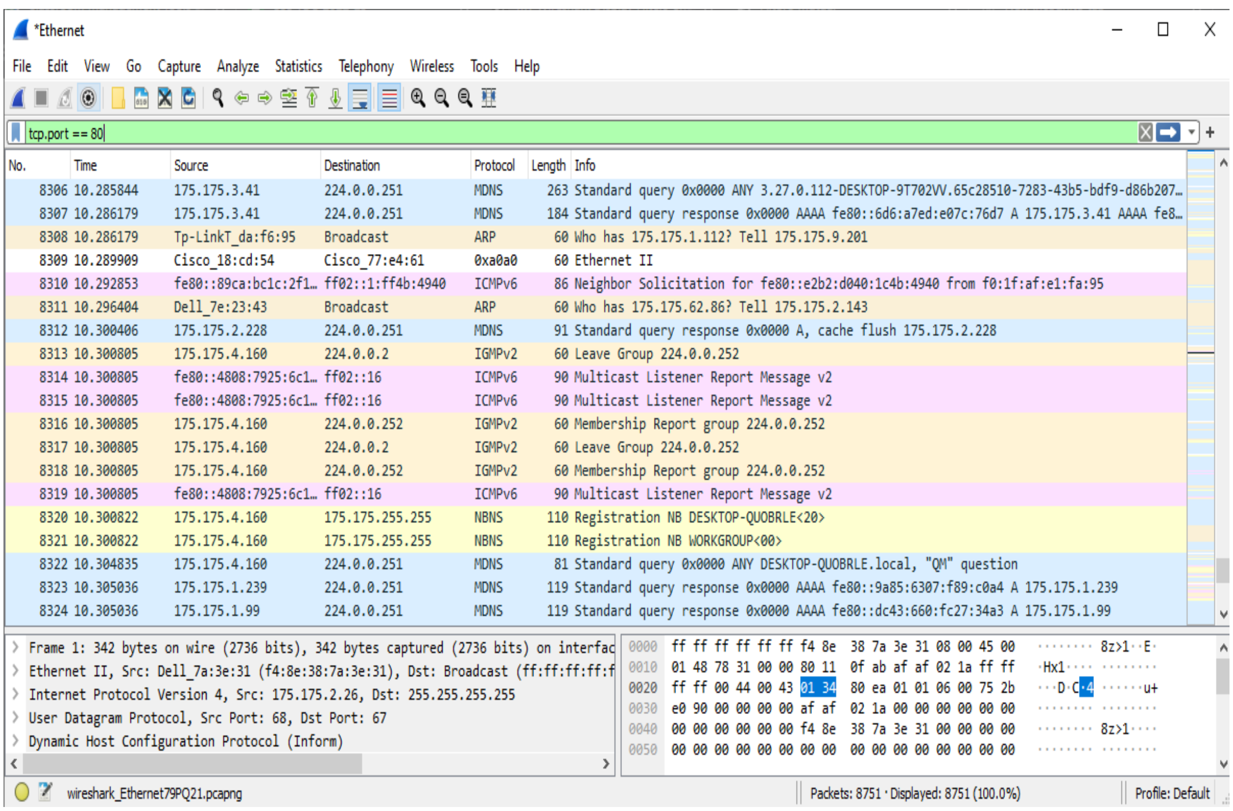
After downloading and installing wireshark, you can launch it and click the name of an interface under Interface List to start capturing packets on that interface. For example, if you want to capture traffic on the wireless network, click your wireless interface. You can configure advanced features by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

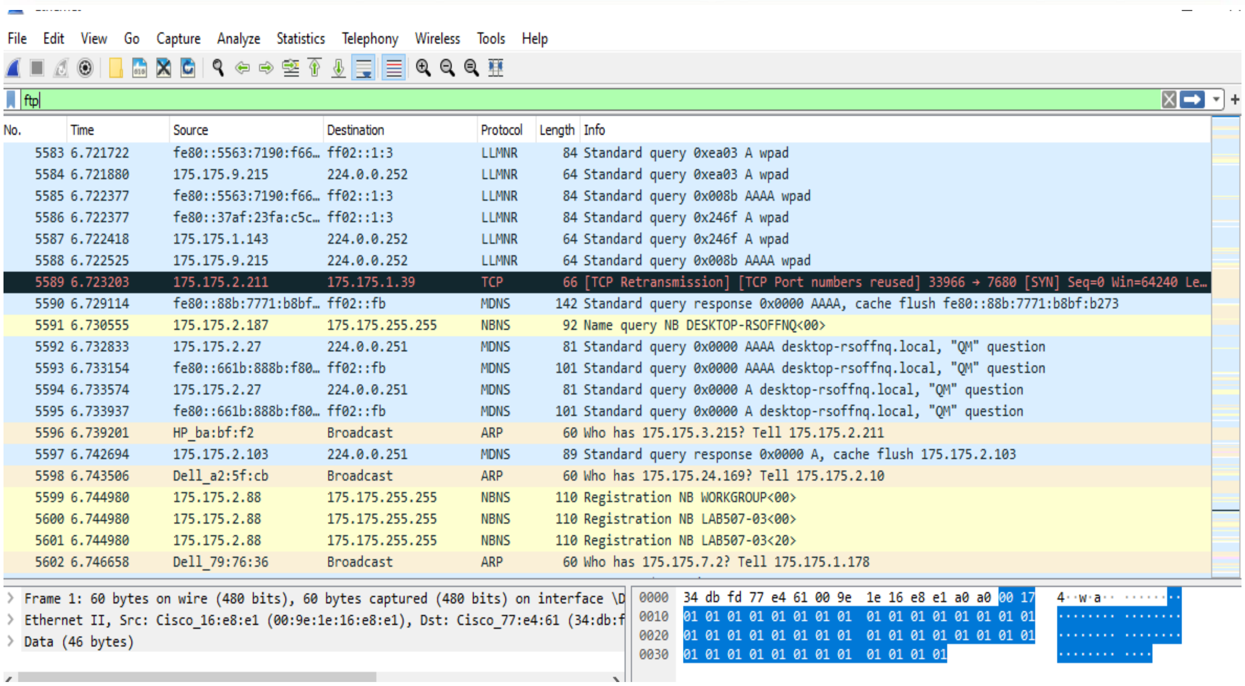
Conclusion: We learned about the Injection Attacks and their types and how they can be used by personnel with bad intentions to exploit an organization and get access to important information. We also understood how cross-site scripting (XSS) works and how it has an impact on security.

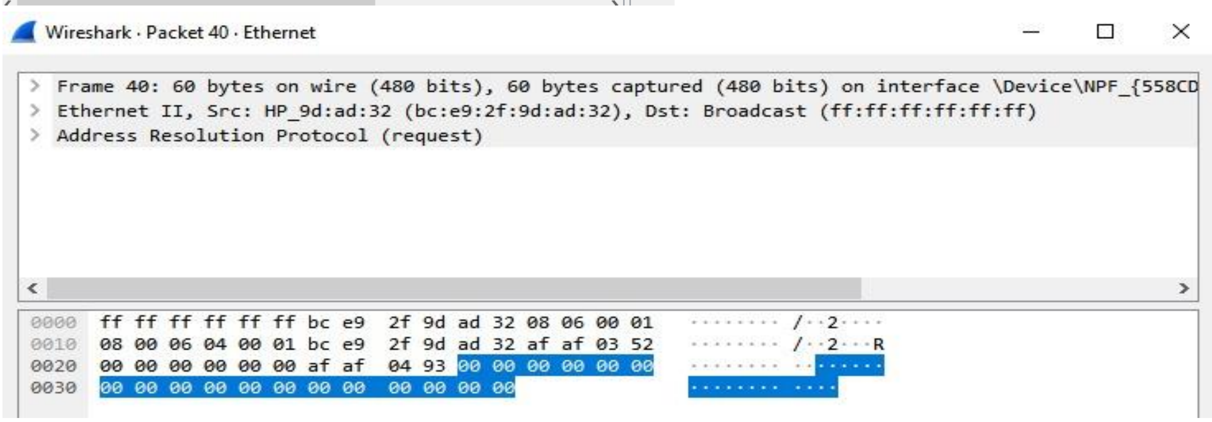
For Faculty Use

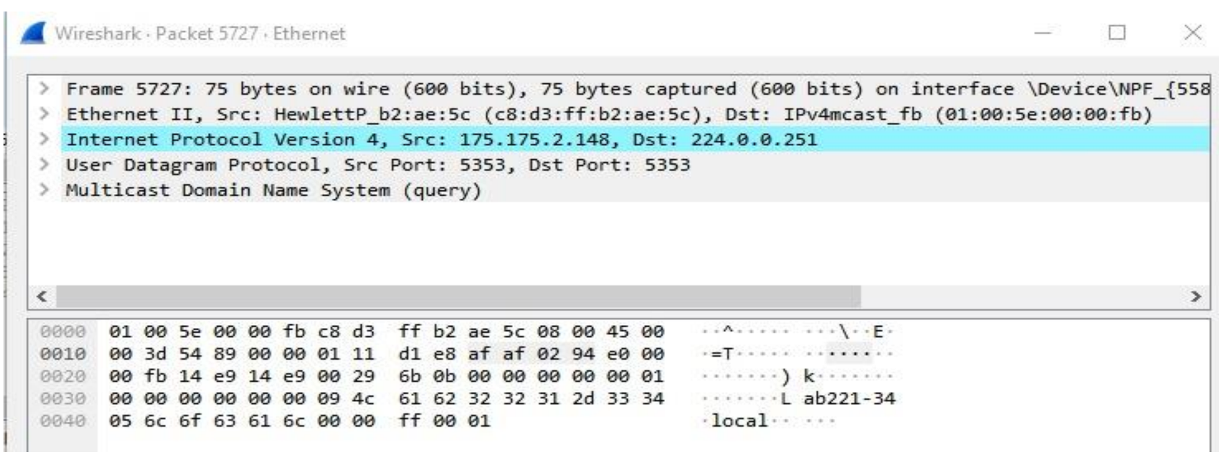
Correction Parameters	Formative Assessment [40%]	Timely completion of Practical [40%]	Attendance / Learning Attitude [20%]	
Marks Obtained				

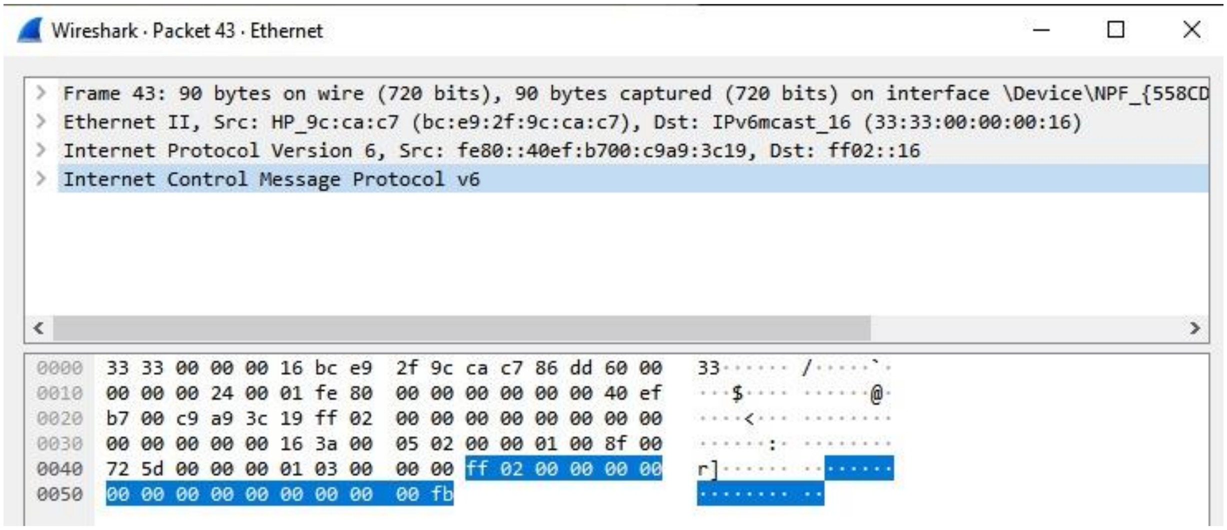

*Ethernet
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
ip.addr == 192.0.2.1
No. Time Source Destination Protocol Length Info
34076 53.392656 fe80::9769:fe7b:8e3… ff02::16 ICMPv6 110 Multicast Listener Report Message v2
34077 53.392656 fe80::9769:fe7b:8e3… ff02::16 ICMPv6 90 Multicast Listener Report Message v2
34078 53.392666 175.175.4.158 224.0.0.252 IGMPv2 60 Membership Report group 224.0.0.252
34079 53.392666 175.175.4.158 224.0.0.2 IGMPv2 60 Leave Group 224.0.0.252
34080 53.393824 fe80::9769:fe7b:8e3… ff02::16 ICMPv6 90 Multicast Listener Report Message v2
34081 53.393831 175.175.4.158 224.0.0.2 IGMPv2 60 Leave Group 224.0.0.251
34082 53.396058 175.175.2.228 224.0.0.251 MDNS 91 Standard query response 0x0000 A, cache flush 175.175.2.228
34083 53.397456 175.175.4.158 224.0.0.252 IGMPv2 60 Membership Report group 224.0.0.252
34084 53.397457 175.175.4.158 224.0.0.251 IGMPv2 60 Membership Report group 224.0.0.251
34085 53.397497 fe80::9769:fe7b:8e3… ff02::16 ICMPv6 90 Multicast Listener Report Message v2
34086 53.397497 fe80::9769:fe7b:8e3… ff02::16 ICMPv6 90 Multicast Listener Report Message v2
34087 53.397921 175.175.4.158 224.0.0.2 IGMPv2 60 Leave Group 224.0.0.252
34088 53.397931 fe80::9769:fe7b:8e3… ff02::16 ICMPv6 90 Multicast Listener Report Message v2
34089 53.402904 fe80::9769:fe7b:8e3… ff02::16 ICMPv6 90 Multicast Listener Report Message v2
34090 53.402917 175.175.4.158 224.0.0.252 IGMPv2 60 Membership Report group 224.0.0.252
34091 53.404028 fe80::9769:fe7b:8e3… ff02::fb MDNS 101 Standard query 0x0000 ANY DESKTOP-HJV9I6F.local, "QM" question
34092 53.404039 175.175.4.158 224.0.0.251 MDNS 81 Standard query 0x0000 ANY DESKTOP-HJV9I6F.local, "QM" question
Frame 34149: 110 bytes on wire (880 bits), 110 bytes captured (880 bits) on interf
Ethernet II, Src: HewlettP_f1:5e:70 (80:ce:62:f1:5e:70), Dst: IPv6mcast_16 (33:33:
Internet Protocol Version 6, Src: fe80::f8e6:aa88:6e67:ad8d, Dst: ff02::16
Internet Control Message Protocol v6
0000 33 33 00 00 00 16 80 ce 62 f1 5e 70 86 dd 60 00
0010 00 00 00 38 00 01 fe 80 00 00 00 00 00 00 f8 e6
0020 aa 88 6e 67 ad 8d ff 02 00 00 00 00 00 00 00 00
0030 00 00 00 00 00 16 3a 00 05 02 00 00 01 00 8f 00
0040 ac 8f 00 00 00 02 04 00 00 00 ff 02 00 00 00 00
0050 00 00 00 00 00 00 00 00 00 fb 04 00 00 00 ff 02
0060 00 00 00 00 00 00 00 00 00 00 00 00 01 00 03

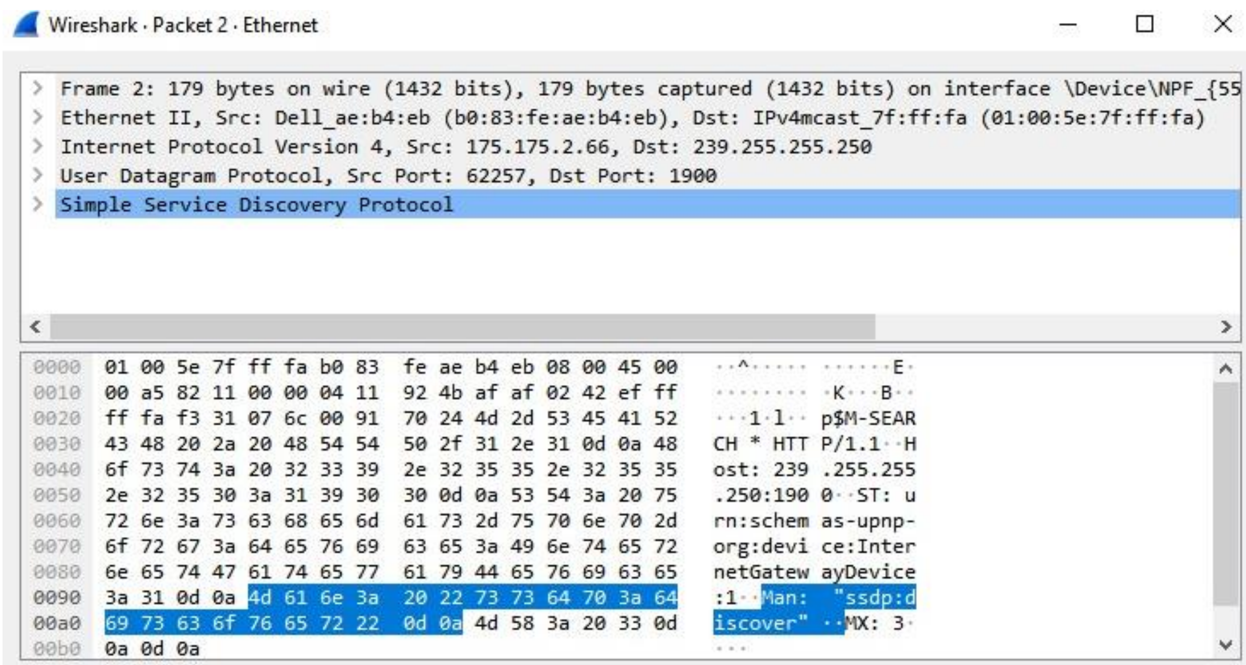

*Ethernet
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp.port == 80
No. Time Source Destination Protocol Length Info
8306 10.285844 175.175.3.41 224.0.0.251 MDNS 263 Standard query 0x0000 ANY 3.27.0.112-DESKTOP-9T702VV.65c28510-7283-43b5-bdf9-d86b207…
8307 10.286179 175.175.3.41 224.0.0.251 MDNS 184 Standard query response 0x0000 AAAA fe80::6d6:a7ed:e07c:76d7 A 175.175.3.41 AAAA fe8…
8308 10.286179 Tp-LinkT_da:f6:95 Broadcast ARP 60 Who has 175.175.1.112? Tell 175.175.9.201
8309 10.289909 Cisco_18:cd:54 Cisco_77:e4:61 0xa0a0 60 Ethernet II
8310 10.292853 fe80::89ca:bc1c:2f1… ff02::1:ff4b:4940 ICMPv6 86 Neighbor Solicitation for fe80::e2b2:d040:1c4b:4940 from f0:1f:af:e1:fa:95
8311 10.296404 Dell_7e:23:43 Broadcast ARP 60 Who has 175.175.62.86? Tell 175.175.2.143
8312 10.300406 175.175.2.228 224.0.0.251 MDNS 91 Standard query response 0x0000 A, cache flush 175.175.2.228
8313 10.300805 175.175.4.160 224.0.0.2 IGMPv2 60 Leave Group 224.0.0.252
8314 10.300805 fe80::4808:7925:6c1… ff02::16 ICMPv6 90 Multicast Listener Report Message v2
8315 10.300805 fe80::4808:7925:6c1… ff02::16 ICMPv6 90 Multicast Listener Report Message v2
8316 10.300805 175.175.4.160 224.0.0.252 IGMPv2 60 Membership Report group 224.0.0.252
8317 10.300805 175.175.4.160 224.0.0.2 IGMPv2 60 Leave Group 224.0.0.252
8318 10.300805 175.175.4.160 224.0.0.252 IGMPv2 60 Membership Report group 224.0.0.252
8319 10.300805 fe80::4808:7925:6c1… ff02::16 ICMPv6 90 Multicast Listener Report Message v2
8320 10.300822 175.175.4.160 175.175.255.255 NBNS 110 Registration NB DESKTOP-QUOBRLE<20>
8321 10.300822 175.175.4.160 175.175.255.255 NBNS 110 Registration NB WORKGROUP<00>
8322 10.304835 175.175.4.160 224.0.0.251 MDNS 81 Standard query 0x0000 ANY DESKTOP-QUOBRLE.local, "QM" question
8323 10.305036 175.175.1.239 224.0.0.251 MDNS 119 Standard query response 0x0000 AAAA fe80::9a85:6307:f89:c0a4 A 175.175.1.239
8324 10.305036 175.175.1.99 224.0.0.251 MDNS 119 Standard query response 0x0000 AAAA fe80::dc43:660:fc27:34a3 A 175.175.1.99
Frame 1: 342 bytes on wire (2736 bits), 342 bytes captured (2736 bits) on interfac
Ethernet II, Src: Dell_7a:3e:31 (f4:8e:38:7a:3e:31), Dst: Broadcast (ff:ff:ff:ff:f
Internet Protocol Version 4, Src: 175.175.2.26, Dst: 255.255.255.255
User Datagram Protocol, Src Port: 68, Dst Port: 67
Dynamic Host Configuration Protocol (Inform)
0000 ff ff ff ff ff ff f4 8e 38 7a 3e 31 08 00 45 00
0010 01 48 78 31 00 00 80 11 0f ab af af 02 1a ff ff
0020 ff ff 00 44 00 43 01 34 80 ea 01 01 06 00 75 2b
0030 e0 90 00 00 00 00 af af 02 1a 00 00 00 00 00 00
0040 00 00 00 00 00 00 f4 8e 38 7a 3e 31 00 00 00 00
0050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
wireshark_Ethernet79PQ21.pcapng
Packets: 8751 · Displayed: 8751 (100.0%)
Profile: Default


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
ftp
No. Time Source Destination Protocol Length Info
5583 6.721722 fe80::5563:7190:f66… ff02::1:3 LLMNR 84 Standard query 0xea03 A wpad
5584 6.721880 175.175.9.215 224.0.0.252 LLMNR 64 Standard query 0xea03 A wpad
5585 6.722377 fe80::5563:7190:f66… ff02::1:3 LLMNR 84 Standard query 0x008b AAAA wpad
5586 6.722377 fe80::37af:23fa:c5c… ff02::1:3 LLMNR 84 Standard query 0x246f A wpad
5587 6.722418 175.175.1.143 224.0.0.252 LLMNR 64 Standard query 0x246f A wpad
5588 6.722525 175.175.9.215 224.0.0.252 LLMNR 64 Standard query 0x008b AAAA wpad
5589 6.723203 175.175.2.211 175.175.1.39 TCP 66 [TCP Retransmission] [TCP Port numbers reused] 33966 → 7680 [SYN] Seq=0 Win=64240 Le…
5590 6.729114 fe80::88b:7771:b8bf… ff02::fb MDNS 142 Standard query response 0x0000 AAAA, cache flush fe80::88b:7771:b8bf:b273
5591 6.730555 175.175.2.187 175.175.255.255 NBNS 92 Name query NB DESKTOP-RSOFFNQ<00>
5592 6.732833 175.175.2.27 224.0.0.251 MDNS 81 Standard query 0x0000 AAAA desktop-rsoffnq.local, "QM" question
5593 6.733154 fe80::661b:888b:f80… ff02::fb MDNS 101 Standard query 0x0000 AAAA desktop-rsoffnq.local, "QM" question
5594 6.733574 175.175.2.27 224.0.0.251 MDNS 81 Standard query 0x0000 A desktop-rsoffnq.local, "QM" question
5595 6.733937 fe80::661b:888b:f80… ff02::fb MDNS 101 Standard query 0x0000 A desktop-rsoffnq.local, "QM" question
5596 6.739201 HP_ba:bf:f2 Broadcast ARP 60 Who has 175.175.3.215? Tell 175.175.2.211
5597 6.742694 175.175.2.103 224.0.0.251 MDNS 89 Standard query response 0x0000 A, cache flush 175.175.2.103
5598 6.743506 Dell_a2:5f:cb Broadcast ARP 60 Who has 175.175.24.169? Tell 175.175.2.10
5599 6.744980 175.175.2.88 175.175.255.255 NBNS 110 Registration NB WORKGROUP<00>
5600 6.744980 175.175.2.88 175.175.255.255 NBNS 110 Registration NB LAB507-03<00>
5601 6.744980 175.175.2.88 175.175.255.255 NBNS 110 Registration NB LAB507-03<20>
5602 6.746658 Dell_79:76:36 Broadcast ARP 60 Who has 175.175.7.2? Tell 175.175.1.178
Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface \D
Ethernet II, Src: Cisco_16:e8:e1 (00:9e:1e:16:e8:e1), Dst: Cisco_77:e4:61 (34:db:f
Data (46 bytes)
0000 34 db fd 77 e4 61 00 9e 1e 16 e8 e1 a0 a0 00 17
0010 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
0020 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
0030 01 01 01 01 01 01 01 01 01 01 01 01


Wireshark · Packet 40 · Ethernet
Frame 40: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface \Device\NPF_{558CD
Ethernet II, Src: HP_9d:ad:32 (bc:e9:2f:9d:ad:32), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Address Resolution Protocol (request)
0000 ff ff ff ff ff ff bc e9 2f 9d ad 32 08 06 00 01
0010 08 00 06 04 00 01 bc e9 2f 9d ad 32 af af 03 52
0020 00 00 00 00 00 00 af af 04 93 00 00 00 00 00 00
0030 00 00 00 00 00 00 00 00 00 00 00 00


Wireshark · Packet 5727 · Ethernet
Frame 5727: 75 bytes on wire (600 bits), 75 bytes captured (600 bits) on interface \Device\NPF_{558
Ethernet II, Src: HewlettP_b2:ae:5c (c8:d3:ff:b2:ae:5c), Dst: IPv4mcast_fb (01:00:5e:00:00:fb)
Internet Protocol Version 4, Src: 175.175.2.148, Dst: 224.0.0.251
User Datagram Protocol, Src Port: 5353, Dst Port: 5353
Multicast Domain Name System (query)
0000 01 00 5e 00 00 fb c8 d3 ff b2 ae 5c 08 00 45 00
0010 00 3d 54 89 00 00 01 11 d1 e8 af af 02 94 e0 00
0020 00 fb 14 e9 14 e9 00 29 6b 0b 00 00 00 00 00 01
0030 00 00 00 00 00 00 09 4c 61 62 32 32 31 2d 33 34
0040 05 6c 6f 63 61 6c 00 00 ff 00 01
L ab221-34
local


Wireshark · Packet 43 · Ethernet
Frame 43: 90 bytes on wire (720 bits), 90 bytes captured (720 bits) on interface \Device\NPF_{558CD
Ethernet II, Src: HP_9c:ca:c7 (bc:e9:2f:9c:ca:c7), Dst: IPv6mcast_16 (33:33:00:00:00:16)
Internet Protocol Version 6, Src: fe80::40ef:b700:c9a9:3c19, Dst: ff02::16
Internet Control Message Protocol v6
0000 33 33 00 00 00 16 bc e9 2f 9c ca c7 86 dd 60 00
0010 00 00 00 24 00 01 fe 80 00 00 00 00 00 00 40 ef
0020 b7 00 c9 a9 3c 19 ff 02 00 00 00 00 00 00 00 00
0030 00 00 00 00 00 16 3a 00 05 02 00 00 01 00 8f 00
0040 72 5d 00 00 00 01 03 00 00 00 ff 02 00 00 00 00
0050 00 00 00 00 00 00 00 00 00 fb


Wireshark · Packet 2 · Ethernet
Frame 2: 179 bytes on wire (1432 bits), 179 bytes captured (1432 bits) on interface \Device\NPF_{55
Ethernet II, Src: Dell_ae:b4:eb (b0:83:fe:ae:b4:eb), Dst: IPv4mcast_7f:ff:fa (01:00:5e:7f:ff:fa)
Internet Protocol Version 4, Src: 175.175.2.66, Dst: 239.255.255.250
User Datagram Protocol, Src Port: 62257, Dst Port: 1900
Simple Service Discovery Protocol
0000 01 00 5e 7f ff fa b0 83 fe ae b4 eb 08 00 45 00
0010 00 a5 82 11 00 00 04 11 92 4b af af 02 42 ef ff
0020 ff fa f3 31 07 6c 00 91 70 24 4d 2d 53 45 41 52 p$M-SEAR
0030 43 48 20 2a 20 48 54 54 50 2f 31 2e 31 0d 0a 48 CH * HTT P/1.1
0040 6f 73 74 3a 20 32 33 39 2e 32 35 35 2e 32 35 35 ost: 239 .255.255
0050 2e 32 35 30 3a 31 39 30 30 0d 0a 53 54 3a 20 75 .250:190 0 ST: u
0060 72 6e 3a 73 63 68 65 6d 61 73 2d 75 70 6e 70 2d rn:schem as-upnp-
0070 6f 72 67 3a 64 65 76 69 63 65 3a 49 6e 74 65 72 org:devi ce:Inter
0080 6e 65 74 47 61 74 65 77 61 79 44 65 76 69 63 65 netGatew ayDevice
0090 3a 31 0d 0a 4d 61 6e 3a 20 22 73 73 64 70 3a 64 :1 Man: "ssdp:d
00a0 69 73 63 6f 76 65 72 22 0d 0a 4d 58 3a 20 33 0d iscover" MX: 3
00b0 0a 0d 0a