



DEPARTMENT OF COMPUTER SCIENCE WITH CYBER SECURITY

Practical Record

Name :

Register Number :

Subject Code : 23USEP04

Subject Title : PRACTICAL: CYBER SECURITY LAB

Year / Sem : II / IV

ACADEMIC YEAR: 2024 - 2025



Certificate

This is to Certify that the Practical Record **“Practical: Cyber Security Lab”** is a bonafide work done by _____

Reg. No. _____ submitted to the Department of Computer Science with Cyber Security, during the academic year 2024 - 2025.

SUBJECT IN-CHARGE
(Mr. K. Gopinath)

HEAD OF THE DEPARTMENT
(Dr. S. Mohanapriya)

Submitted for University Practical Examination held on _____

INTERNAL EXAMINER

EXTERNAL EXAMINER

INDEX

S. No.	Date	Title	Page No.	Signature
1.		Implement the following Substitution & Transposition Techniques concepts: a) Caesar Cipher b) Railfence row & Column Transformation		
2.		Implement the Diffie-Hellman Key Exchange mechanism using HTML and JavaScript		
3.		Implement the following Attack: a) Dictionary Attack b) Brute Force Attack		
4.		Installation of Wire shark, tcp dump, etc and observe data transferred in client server communication using UDP/TCP and identify the UDP/TCP datagram		
5.		Installation of rootkits and study about the variety of options		
6.		Demonstrate intrusion detection system using any tool (snort or any other s/w).		
7.		Demonstrate how to provide secure data storage, secure data transmission and for creating digital signatures		

Ex.No: 01.a)	Implement the following Substitution & Transposition Techniques concepts: Caesar Cipher
Date:	

AIM

To write the program in python to implement the Substitution and Transposition Techniques concepts Caesar Cipher.

ALGORITHM

Step 1. Input Validation

- a) Validate user input for text and shift values.
- b) Ensure shift is an integer.

Step 2. Encryption/Decryption Mode Selection

- a) Determine whether to encrypt (E) or decrypt (D) based on user input.

Step 3. Caesar Cipher Formula Application

- a) Apply the Caesar cipher formula to each character in the text:
- b) $\text{encrypted_char} = (\text{ord}(\text{char}) - \text{shift_base} + \text{shift}) \% 26 + \text{shift_base}$
- c) Handle non-alphabetic characters by leaving them unchanged.

Step 4. Shift Value Adjustment

- a) Adjust the shift value based on the encryption/decryption mode:
 - $\text{shift} = -\text{shift}$ for decryption

Step 5. Output Generation: Generate the encrypted or decrypted text

based on the transformed characters.

Step 6. Result Output - Print the final encrypted or decrypted text.

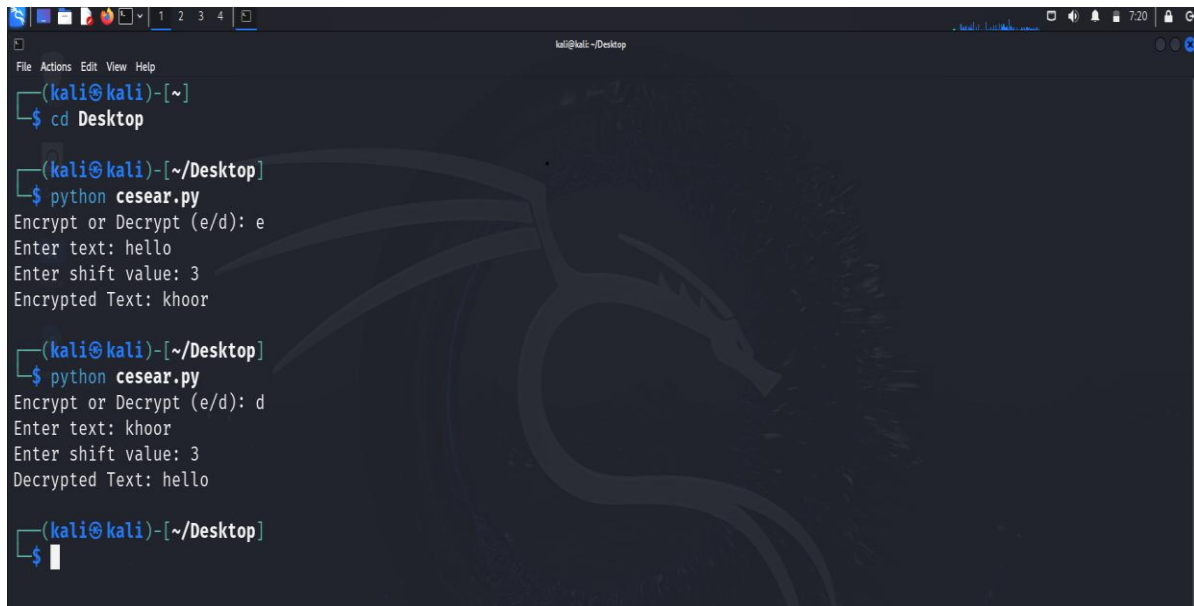
CODING:

```
def caesar_cipher(text, shift, decrypt=False):
    result = ""
    If decrypt:
        shift = -shift # Reverse shift for decryption
    for char in text:
        if char.isalpha():
            shift_base = 65 if char.isupper() else 97
            result += chr((ord(char) - shift_base + shift) % 26 + shift_base)
        else:
            result += char
    return result

choice = input("Encrypt or Decrypt (e/d): ").lower()
text = input("Enter text: ")
shift = int(input("Enter shift value: "))

if choice == 'e':
    print("Encrypted Text:", caesar_cipher(text, shift))
elif choice == 'd':
    print("Decrypted Text:", caesar_cipher(text, shift, decrypt=True))
else:
    print("Invalid Choice!")
```

OUTPUT:

A terminal window titled 'kali@kali: ~/Desktop' with a menu bar (File, Actions, Edit, View, Help) and a taskbar at the top. The terminal shows the execution of a Python script named 'cesear.py'. The first run encrypts the text 'hello' with a shift of 3, resulting in 'khoor'. The second run decrypts 'khoor' with a shift of 3, resulting in 'hello'. The background of the terminal features a faint, stylized dragon logo.

```
(kali@kali)-[~]
$ cd Desktop

(kali@kali)-[~/Desktop]
$ python cesear.py
Encrypt or Decrypt (e/d): e
Enter text: hello
Enter shift value: 3
Encrypted Text: khoor

(kali@kali)-[~/Desktop]
$ python cesear.py
Encrypt or Decrypt (e/d): d
Enter text: khoor
Enter shift value: 3
Decrypted Text: hello

(kali@kali)-[~/Desktop]
$
```

RESULT

The Caesar Cipher program successfully encrypts the plaintext by shifting each letter by the specified key value, producing the corresponding ciphertext.

Ex.No:01.b)	Implement the following Substitution & Transposition Techniques concepts: Rail Fence Cipher
Date:	

AIM

To write the program in python to implement the Substitution and Transposition Techniques concepts Railfence Rows & Columns transformation.

ALGORITHM

Step1. Input Validation

- a) Validate user input for text and rails values.
- b) Ensure rails is an integer.

Step 2. Rail Fence Structure Creation

- a) Create a rail fence structure with rails number of rows and text length number of columns.

Step 3. Text Weaving

- a) Weave the text into the rail fence structure by moving diagonally up and down the rails.

Step 4. Encryption/Decryption Mode Selection

- a) Determine whether to encrypt (E) or decrypt (D) based on user input.

Step 5. Ciphertext/Plaintext Generation

- a) Generate the ciphertext (encrypted text) or plaintext (decrypted text) by reading the characters from the rail fence structure.

Step 6. Result Output - Print the final encrypted or decrypted text.

CODING

```
def rail_fence_encrypt(text, rails):
    fence = [['\n'] * len(text) for _ in range(rails)]
    rail = 0
    var = 1

    for i in range(len(text)):
        fence[rail][i] = text[i]
        rail += var
        if rail == rails - 1 or rail == 0:
            var = -var

    result = []
    for rail in fence:
        for char in rail:
            if char != '\n':
                result.append(char)
    return "".join(result)

def rail_fence_decrypt(cipher, rails):
    fence = [['\n'] * len(cipher) for _ in range(rails)]
    rail = 0
    var = 1

    for i in range(len(cipher)):
        fence[rail][i] = '*'
        rail += var
        if rail == rails - 1 or rail == 0:
            var = -var

    idx = 0
    for i in range(rails):
        for j in range(len(cipher)):
            if fence[i][j] == '*' and idx < len(cipher):
```



```

        fence[i][j] = cipher[idx]
    I        dx += 1

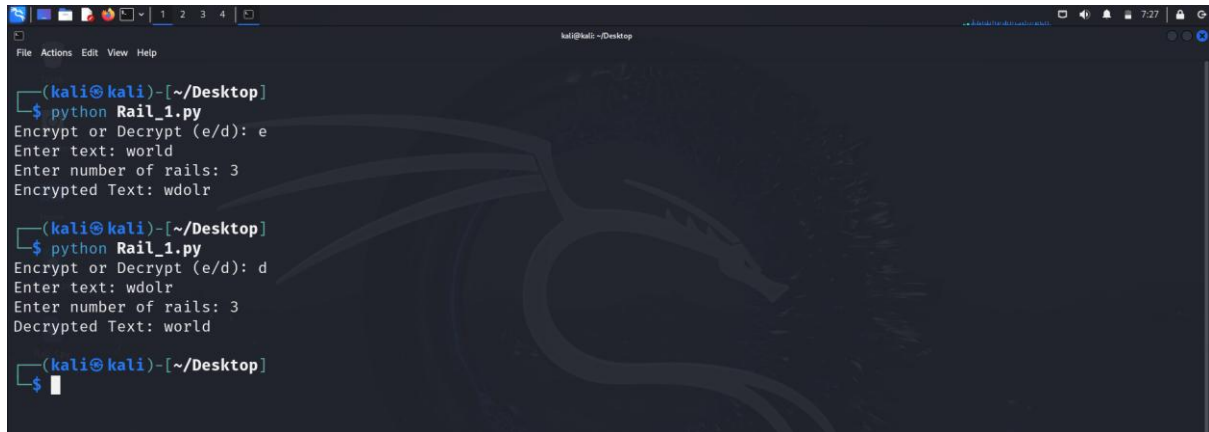
result = []
rail = 0
var = 1
for i in range(len(cipher)):
    result.append(fence[rail][i])
    rail += var
    if rail == rails - 1 or rail == 0:
        var = -var
    return "".join(result)

choice = input("Encrypt or Decrypt (e/d): ").lower()
text = input("Enter text: ")
rails = int(input("Enter number of rails: "))

if choice == 'e':
    print("Encrypted Text:", rail_fence_encrypt(text, rails))
elif choice == 'd':
    print("Decrypted Text:", rail_fence_decrypt(text, rails))
else:
    print("Invalid Choice!")

```

OUTPUT



```
(kali㉿kali)-[~/Desktop]
$ python Rail_1.py
Encrypt or Decrypt (e/d): e
Enter text: world
Enter number of rails: 3
Encrypted Text: wdolr

(kali㉿kali)-[~/Desktop]
$ python Rail_1.py
Encrypt or Decrypt (e/d): d
Enter text: wdolr
Enter number of rails: 3
Decrypted Text: world

(kali㉿kali)-[~/Desktop]
$
```

RESULT

The Rail Fence Cipher successfully encrypts and decrypts the plaintext by arranging characters in a zigzag pattern across multiple rails and reading them row by row.

Ex.No: 02	Implement the Diffie-Hellman Key Exchange mechanism using HTML and JavaScript
Date:	

AIM

To implement the Diffie-Hellman key exchange algorithm for securely sharing a secret key between two parties.

ALGORITHM

Step 1: Input the prime number p , base g , private key a for User A, and private key b for User B.

Step 2: Calculate $A = g^a \mod p$ for User A.

Step 3: Calculate $B = g^b \mod p$ for User B.

Step 4: Send A to User B and B to User A.

Step 5: Calculate the shared secret for User A as

$$\text{sharedSecretA} = B^a \mod p$$

Step 6: Calculate the shared secret for User B as

$$\text{sharedSecretB} = A^b \mod p$$

Step 7: Compare if both shared secrets match.

Step 8: If the shared secrets match, display success; otherwise, display failure.

Step 9: Ensure that p and g are large prime numbers to maintain security.

Step 10: Implement error handling to check for invalid inputs and computation errors during the process.

CODING

```
<!DOCTYPE html>

<html lang="en">

<head>

  <title>Diffie-Hellman</title>

</head>

<body>

  <h3>Diffie-Hellman Key Exchange</h3>

  <input id="p" placeholder="Prime (p)" type="number">

  <input id="g" placeholder="Base (g)" type="number">

  <input id="a" placeholder="Private Key A" type="number">

  <input id="b" placeholder="Private Key B" type="number">

  <button onclick="dh()">Calculate</button>

  <p id="out"></p>

  <script>

    // Modular Exponentiation function (iterative)

    const modExp = (base, exp, mod) => {

      let result = 1;

      base = base % mod;

      while (exp > 0) {

        if (exp % 2 === 1) result = (result * base) % mod;

        exp = Math.floor(exp / 2);

        base = (base * base) % mod;

      }

      return result;

    };

    // Diffie-Hellman Key Exchange Logic

    function dh() {

      const p = parseInt(document.getElementById('p').value);

      const g = parseInt(document.getElementById('g').value);

      const a = parseInt(document.getElementById('a').value);

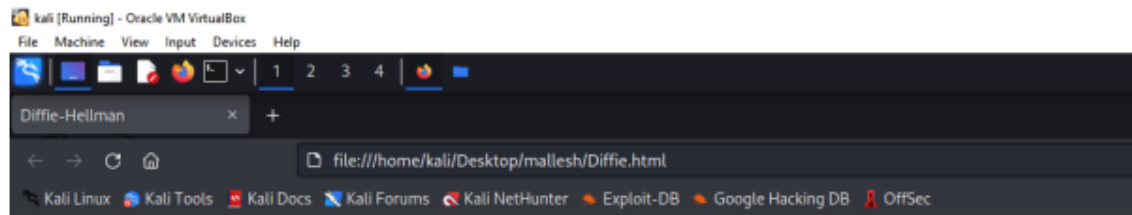
      const b = parseInt(document.getElementById('b').value);

      // Calculate A and B using modular exponentiation

      const A = modExp(g, a, p); // A = g^a mod p
```

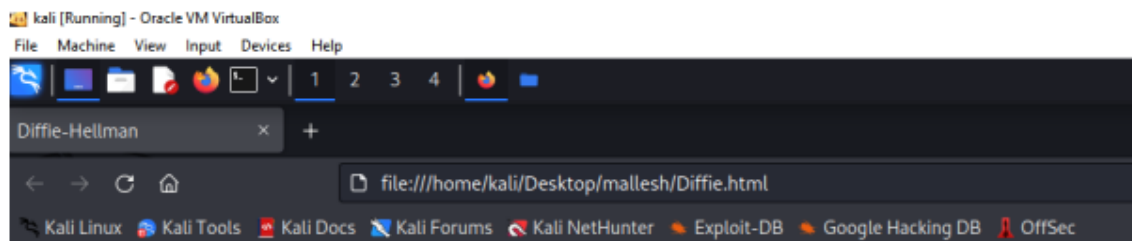
```
const B = modExp(g, b, p); // B = g^b mod p
// Calculate the shared secret
const sharedSecretA = modExp(B, a, p); // sharedSecretA = B^a mod p
const sharedSecretB = modExp(A, b, p); // sharedSecretB = A^b mod p
// Output the shared secret and check if they match
document.getElementById('out').innerText = "Shared Secret: ${sharedSecretA}
(User
A), ${sharedSecretB} (User B)";
if (sharedSecretA === sharedSecretB) {
  alert("Key Exchange Successful! Shared Secret: " + sharedSecretA);
} else {
  alert("Key Exchange Failed!");
}
}
</script>
</body>
</html>
```

OUTPUT



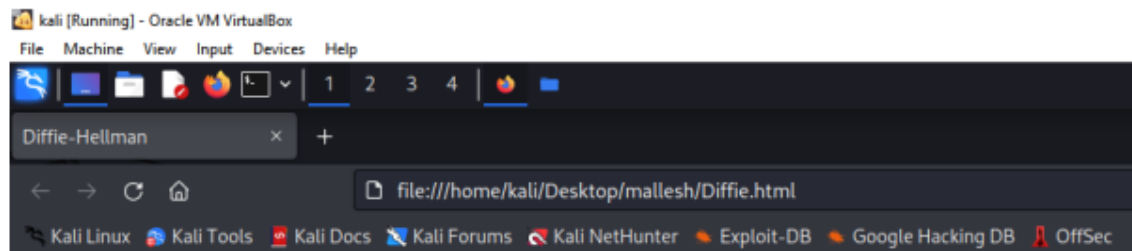
Diffie-Hellman Key Exchange

Prime (p) Base (g) Private Key A Private Key B Calculate



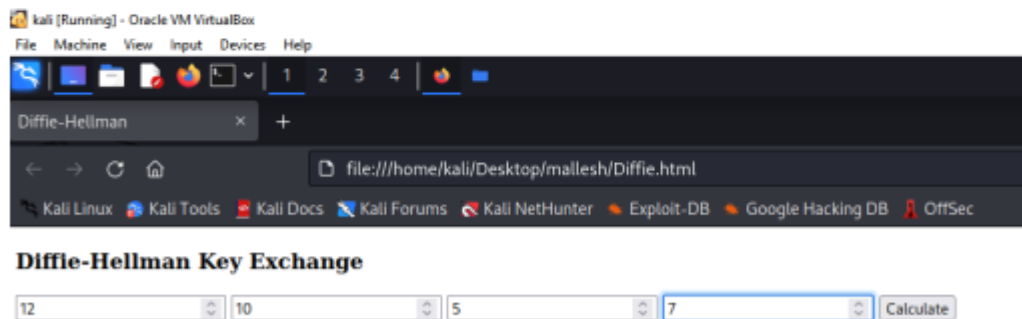
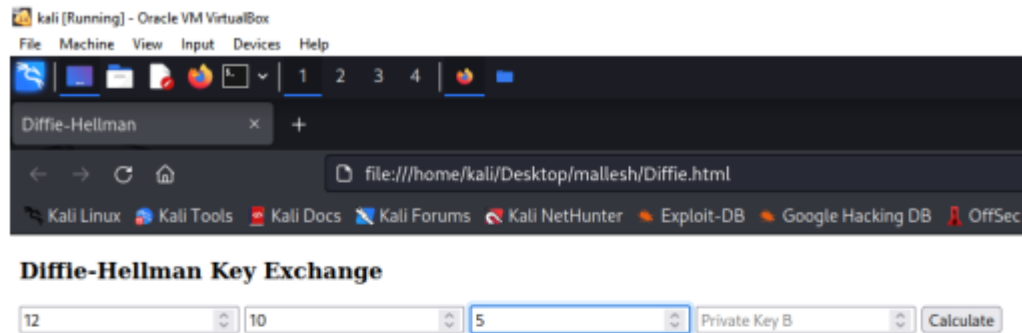
Diffie-Hellman Key Exchange

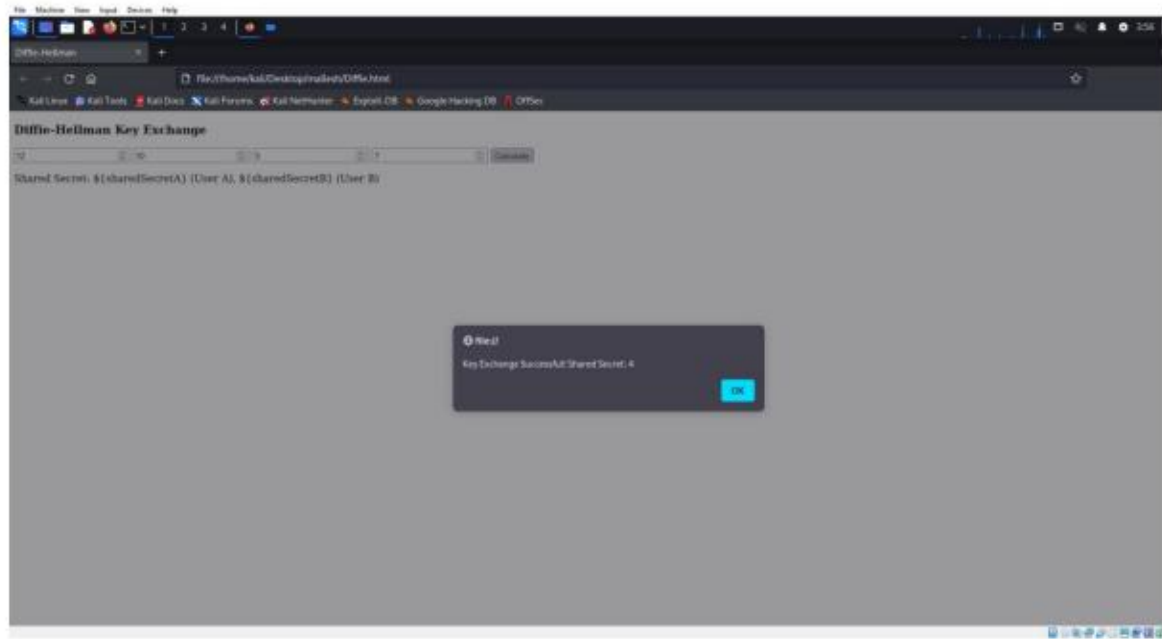
12 Base (g) Private Key A Private Key B Calculate



Diffie-Hellman Key Exchange

12 10 Private Key A Private Key B Calculate





RESULT

Thus the above implement the Diffie-Hellman Key Exchange mechanism using HTML and JavaScript successfully verified

Ex. No: 03 a)	Implement the following Attack: Dictionary attack
Date:	

AIM

To crack passwords using a dictionary attack method by employing a predefined wordlist (e.g., rockyou.txt) with **John the Ripper**.

ALGORITHM

Step 1: Locate the wordlist: locate rockyou.txt.

Step 2: Run John the Ripper with the wordlist: john wordlist = /usr/share/wordlists/rockyou.txt.gz.

Step 3: Locate the target password file: locate password.txt.

Step 4: Crack passwords in the target file: john -- wordlist = /usr/share/wordlists/rockyou.txt.gz password.txt.

COMMAND

Locate rockyou.txt

```
john --wordlist=/usr/share/wordlists/rockyou.txt.gz
```

locate password.txt john

```
wordlist=/usr/share/wordlists/rockyou.txt.gz password.txt
```

OUTPUT

```
(kali㉿kali)-[~/Desktop/GURU]
$ john --wordlist=/usr/share/wordlists/rockyou.txt.gz password.txt

Warning: detected hash type "cryptoSafe", but the string is also recognized as "gost"
Use the "--format=gost" option to force loading these as that type instead
Warning: detected hash type "cryptoSafe", but the string is also recognized as "HAVAL-256-3"
Use the "--format=HAVAL-256-3" option to force loading these as that type instead
Warning: detected hash type "cryptoSafe", but the string is also recognized as "Panama"
Use the "--format=Panama" option to force loading these as that type instead
Warning: detected hash type "cryptoSafe", but the string is also recognized as "po"
Use the "--format=po" option to force loading these as that type instead
Warning: detected hash type "cryptoSafe", but the string is also recognized as "Raw-Keccak-256"
Use the "--format=Raw-Keccak-256" option to force loading these as that type instead
Warning: detected hash type "cryptoSafe", but the string is also recognized as "Raw-SHA256"
Use the "--format=Raw-SHA256" option to force loading these as that type instead
Warning: detected hash type "cryptoSafe", but the string is also recognized as "skein-256"
Use the "--format=skein-256" option to force loading these as that type instead
Warning: detected hash type "cryptoSafe", but the string is also recognized as "Snefru-256"
Use the "--format=Snefru-256" option to force loading these as that type instead
Warning: detected hash type "cryptoSafe", but the string is also recognized as "Stribog-256"
Use the "--format=Stribog-256" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 4 password hashes with 4 different salts (cryptoSafe [AES-256-CBC])
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: UTF-16 BOM seen in wordlist. File may not be read properly unless you re-encode it
0g 0:00:00:00 DONE (2025-01-07 14:23) 0g/s 163521p/s 654087c/s 654087C/s Q♦C,♦)♦♦♦y♦E>♦.♦)░░o♦T♦♦b♦♦♦.. ♦♦♦♦♦V
Session completed.
```

RESULT

Password successfully cracked using a predefined wordlist by using Dictionary attack.

Ex.No:03 b)

Date:

Implement the following Attack:

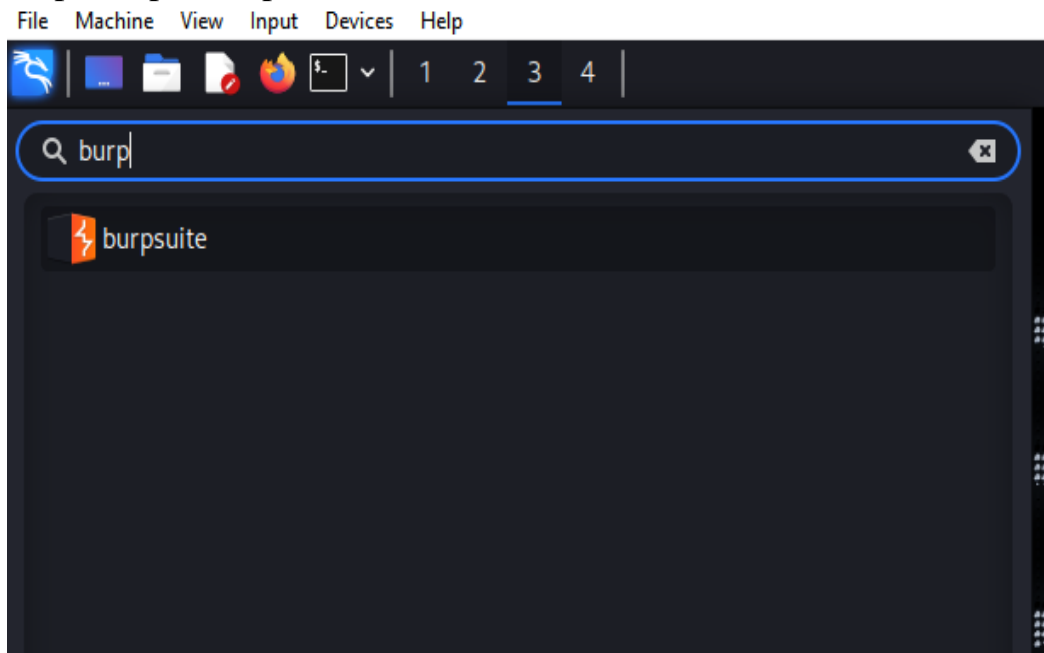
Brute Force Attack

Aim:

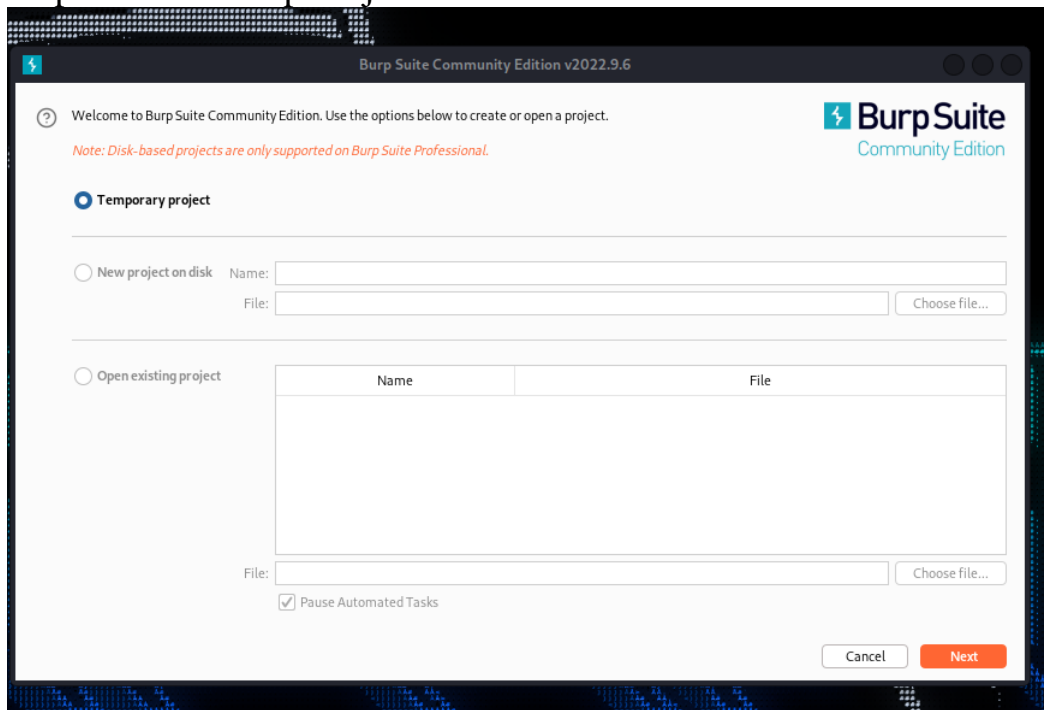
To perform a Brute Force Attack in the website using kali linux.

Procedure:

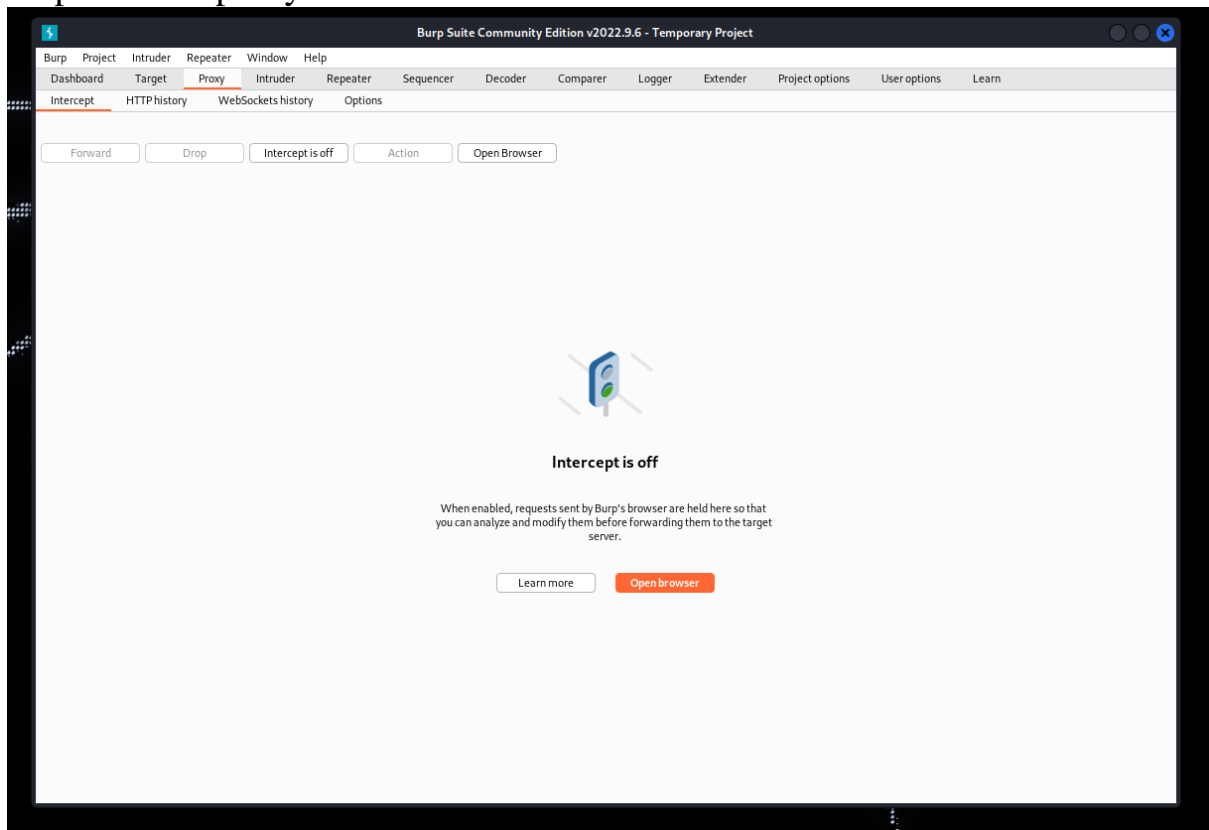
Step-1: Open burpsuit



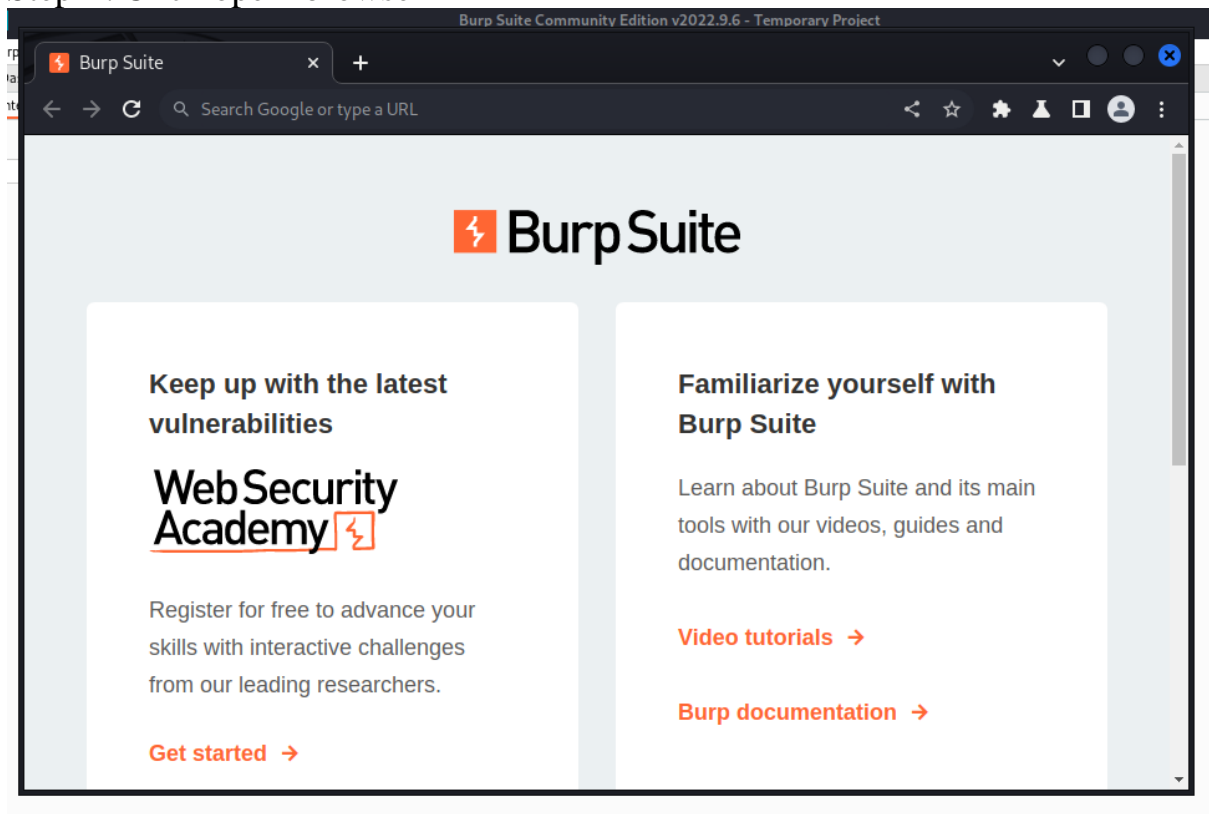
Step-2: Start a temp Project



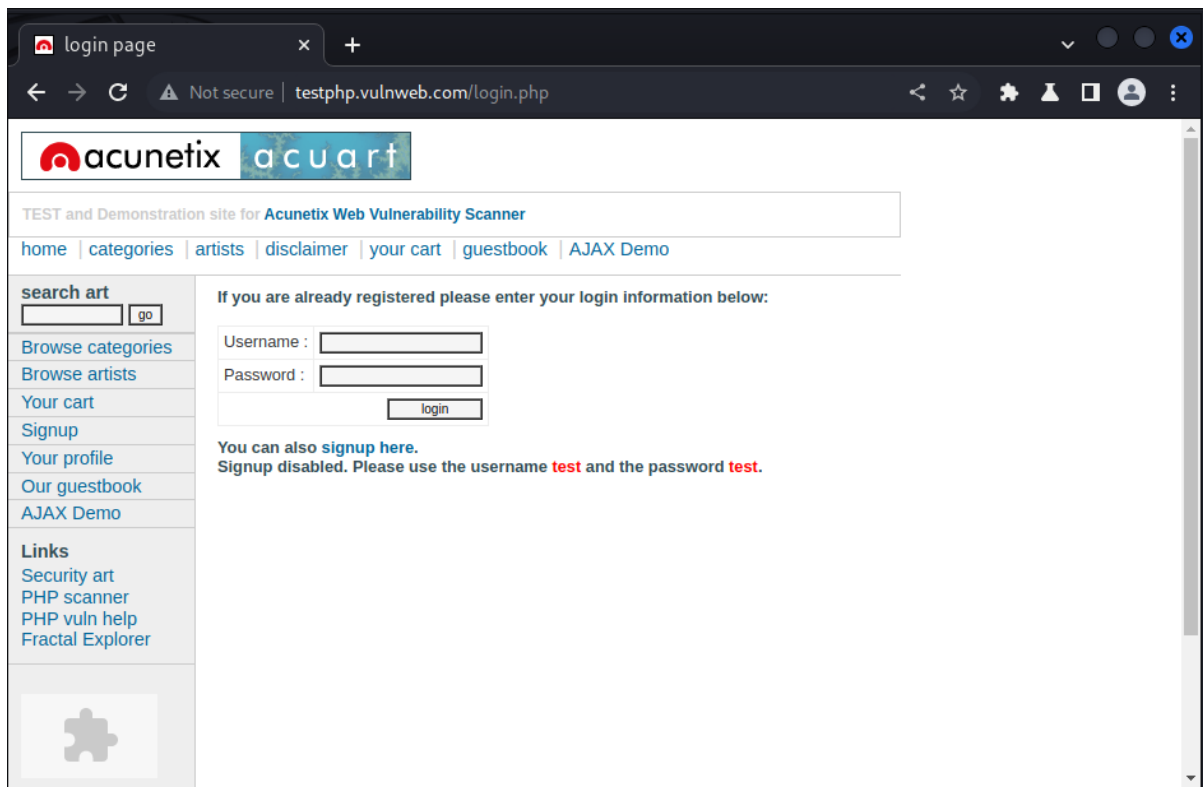
Step-3: Go to proxy



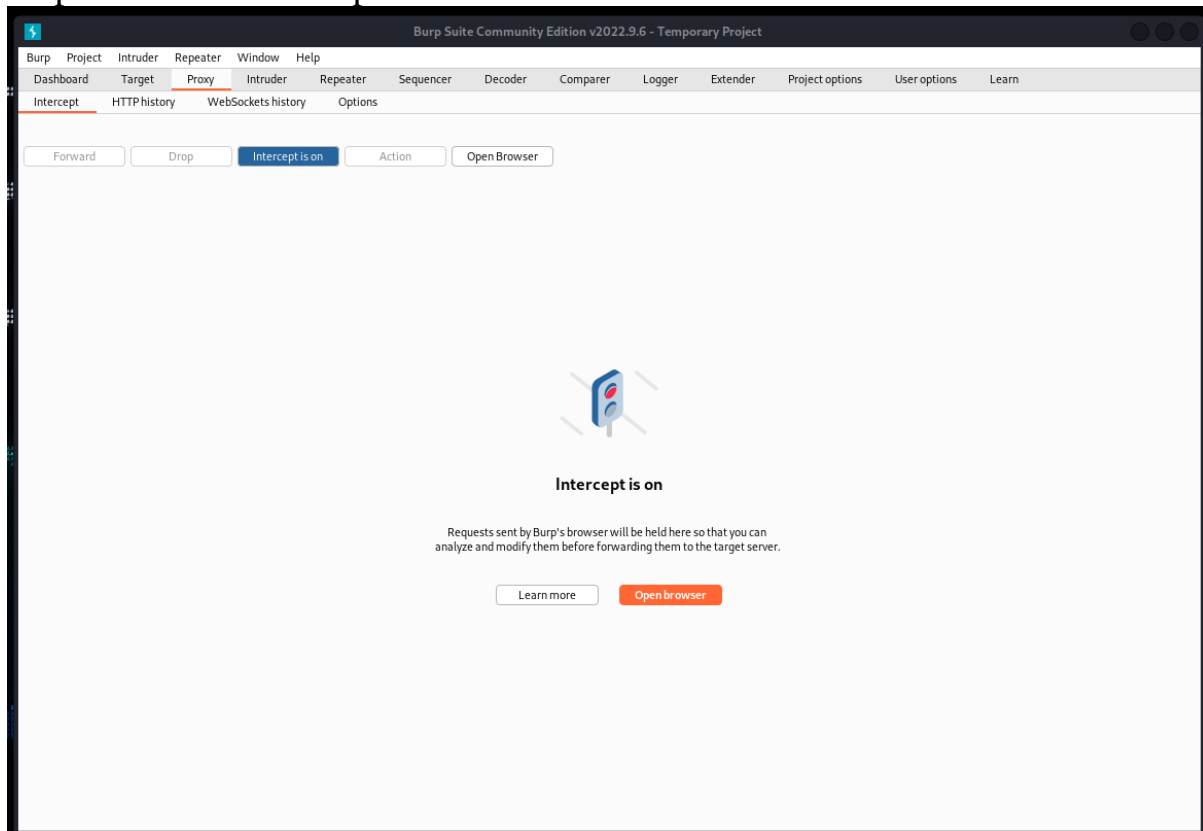
Step-4: Click open browser



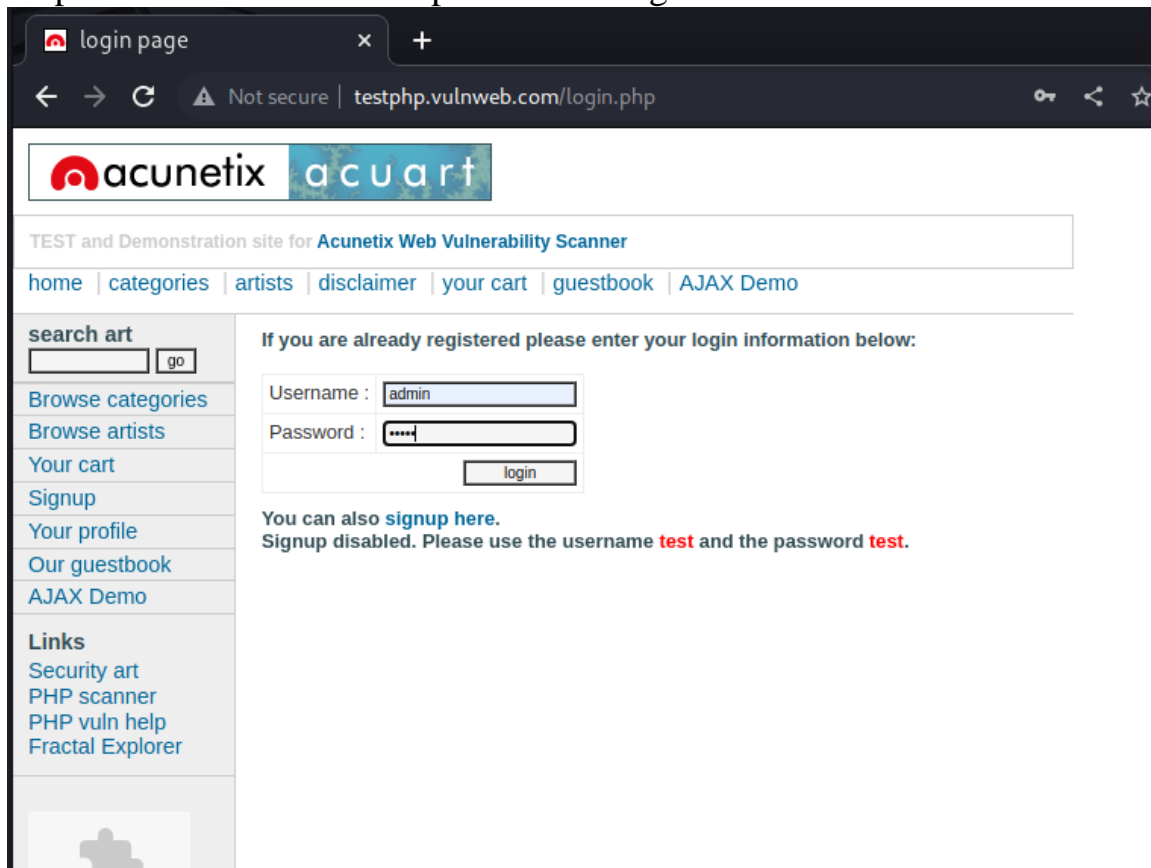
Step-5: Navigate to testphp.vulnweb.com/login.php



Step-6: Turn on intercept

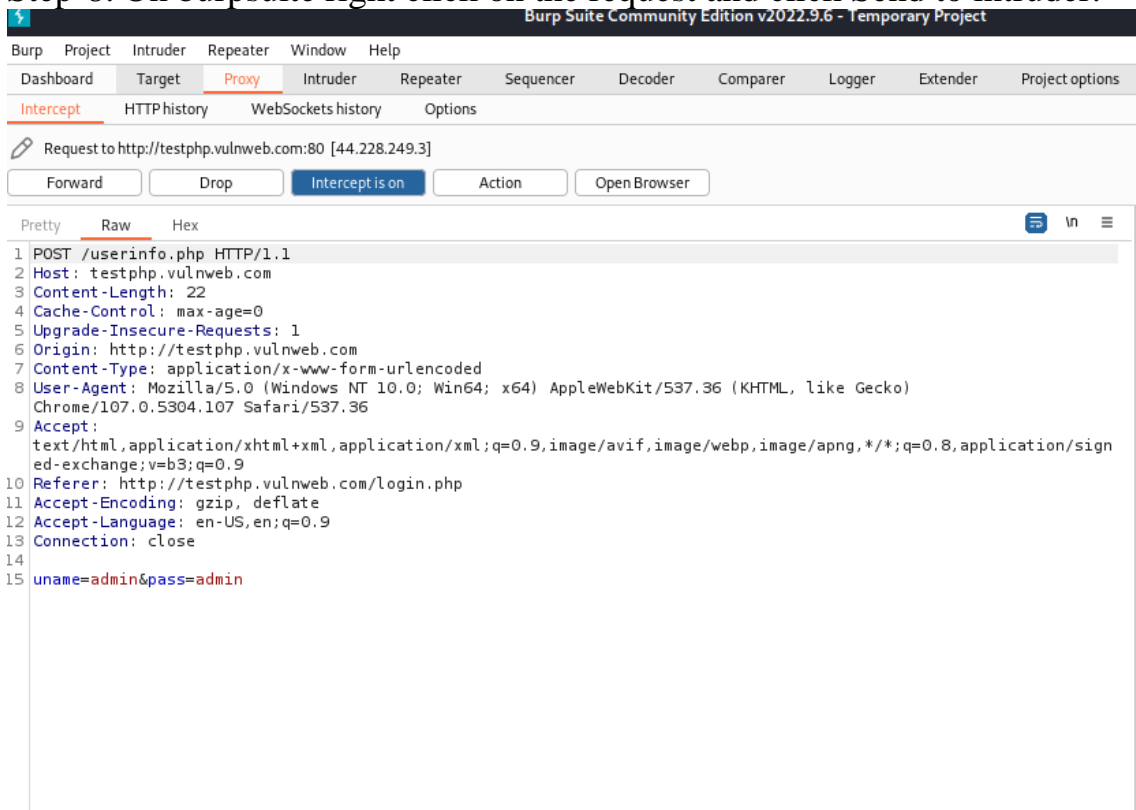


Step-7: Use a username and password to login



The screenshot shows a web browser window with the address bar displaying "testphp.vulnweb.com/login.php". The page features the Acunetix logo and a navigation menu with links like "home", "categories", "artists", "disclaimer", "your cart", "guestbook", and "AJAX Demo". A search bar is present on the left. The main content area contains a login form with fields for "Username" (containing "admin") and "Password" (containing "****"), and a "login" button. A message below the form states: "If you are already registered please enter your login information below: You can also [signup here](#). Signup disabled. Please use the username **test** and the password **test**."

Step-8: On burpsuite right click on the request and click Send to intruder.



Step-9: Go to Intruder

Choose an attack type

Attack type: Sniper

Choose an attack type

Payload Positions

Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Target: http://testphp.vulnweb.com ☒ Update Host header to match target

```
1 POST /userinfo.php HTTP/1.1
2 Host: testphp.vulnweb.com
3 Content-Length: 22
4 Cache-Control: max-age=0
5 Upgrade-Insecure-Requests: 1
6 Origin: http://testphp.vulnweb.com
7 Content-Type: application/x-www-form-urlencoded
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.5304.107 Safari/537.36
9 Accept:
10 text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
11 Referer: http://testphp.vulnweb.com/login.php
12 Accept-Encoding: gzip, deflate
13 Accept-Language: en-US,en;q=0.9
14 Connection: close
15 uname=$admin$&pass=$admin$
```

Step-10: Go to position and set attack type to cluster bomb.

Choose an attack type

Attack type: Sniper

Choose an attack type

Payload Positions

Configure the

Target

Cluster bomb

This attack uses multiple payload sets. There is a different payload set for each defined position (up to a maximum of 20). The attack iterates through each payload set in turn, so that all permutations of payload combinations are tested.

```
1 POST
2 Host:
3 Conte
4 Cache
5 Upgra
6 Origin: http://testphp.vulnweb.com
7 Content-Type: application/x-www-form-urlencoded
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.5304.107 Safari/537.36
9 Accept:
10 text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
11 Referer: http://testphp.vulnweb.com/login.php
12 Accept-Encoding: gzip, deflate
13 Accept-Language: en-US,en;q=0.9
14 Connection: close
15 uname=$admin$&pass=$admin$
```

Step-11: Add payloads [admin, user, test]

BurpProjectIntruderRepeaterWindowHelp

DashboardTargetProxyIntruderRepeaterSequencerDecoderComparerLogger

1 x2 x+

PositionsPayloadsResource PoolOptions

?

Payload Sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various different ways.

Payload set:1

Payload count:3

Payload type:Simple list

Request count:0

?

Payload Options [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

Paste

Load...

Remove

Clear

Deduplicate

admin

user

test

Add

Add from list ... [Pro version only]

Step-12: Setup for both payloads

BurpProjectIntruderRepeaterWindowHelp

DashboardTargetProxyIntruderRepeaterSequencerDecoderComparer

1 x2 x+

PositionsPayloadsResource PoolOptions

?

Payload Sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various different ways.

Payload set:2

Payload count:3

Payload type:Simple list

Request count:9

?

Payload Options [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

Paste

Load...

Remove

Clear

Deduplicate

admin

user

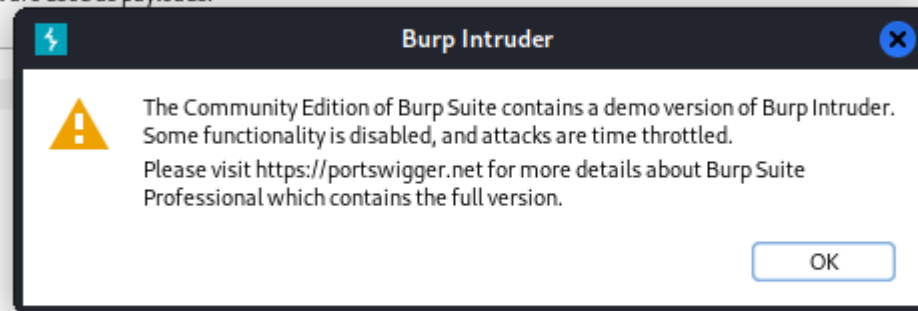
test

Add

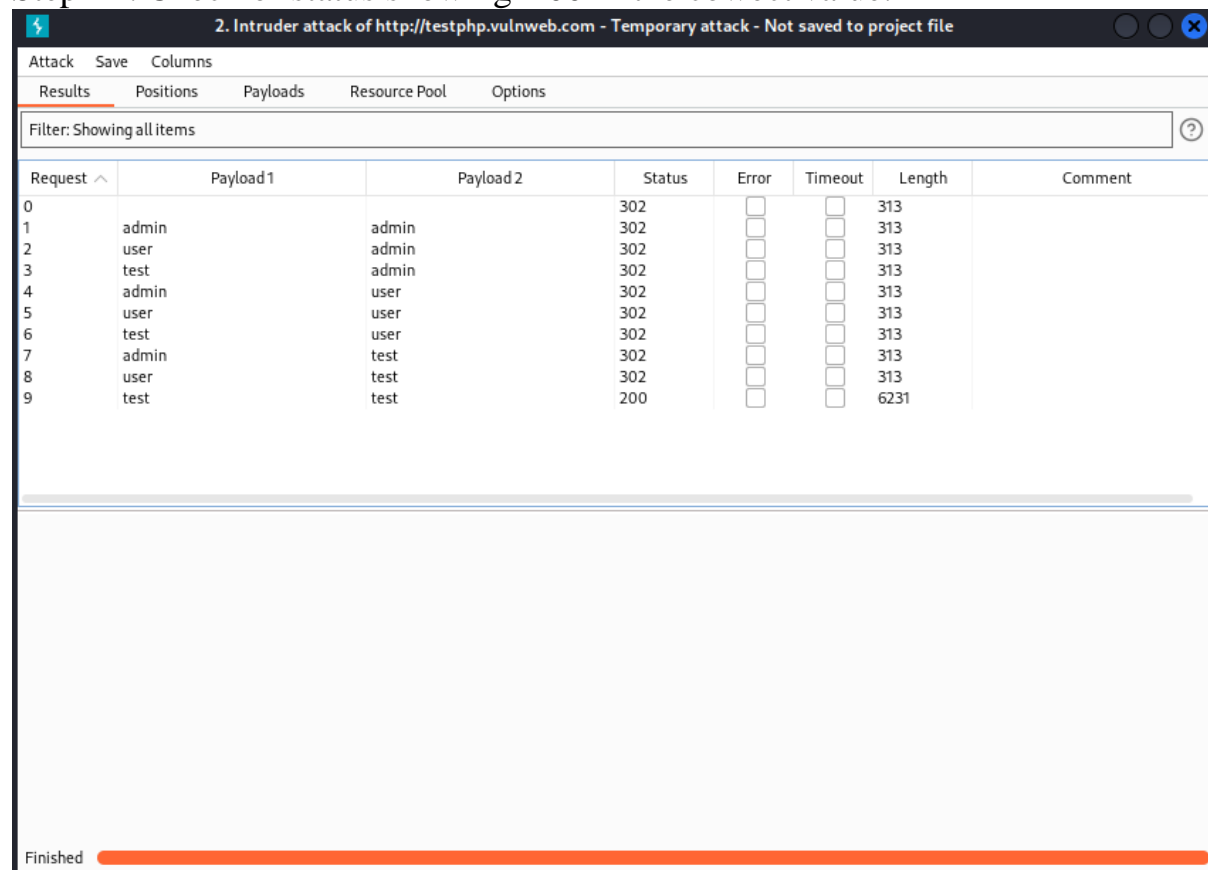
Add from list ... [Pro version only]

Step-13: Start attack and click ok

that are used as payloads.



Step-14: Check of status showing '200' if the correct value.

A screenshot of the Burp Intruder "Results" tab. The window title is "2. Intruder attack of http://testphp.vulnweb.com - Temporary attack - Not saved to project file". It has tabs for "Attack", "Save", "Columns", "Results", "Positions", "Payloads", "Resource Pool", and "Options". The "Results" tab is active, showing a table of attack results. A filter bar at the top says "Filter: Showing all items". The table has columns: Request, Payload1, Payload2, Status, Error, Timeout, Length, and Comment. The "Status" column shows 302 for requests 1-8 and 200 for request 9. The "Error" and "Timeout" columns have checkboxes. The "Length" column shows 313 for requests 1-8 and 6231 for request 9. At the bottom, there is a "Finished" label and a progress bar that is 100% complete.

Request	Payload1	Payload2	Status	Error	Timeout	Length	Comment
0			302	<input type="checkbox"/>	<input type="checkbox"/>	313	
1	admin	admin	302	<input type="checkbox"/>	<input type="checkbox"/>	313	
2	user	admin	302	<input type="checkbox"/>	<input type="checkbox"/>	313	
3	test	admin	302	<input type="checkbox"/>	<input type="checkbox"/>	313	
4	admin	user	302	<input type="checkbox"/>	<input type="checkbox"/>	313	
5	user	user	302	<input type="checkbox"/>	<input type="checkbox"/>	313	
6	test	user	302	<input type="checkbox"/>	<input type="checkbox"/>	313	
7	admin	test	302	<input type="checkbox"/>	<input type="checkbox"/>	313	
8	user	test	302	<input type="checkbox"/>	<input type="checkbox"/>	313	
9	test	test	200	<input type="checkbox"/>	<input type="checkbox"/>	6231	

Result:

Thus the above methods are used to open and execute the Brute Force Attack .

Ex.No:04

Date:

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

AIM

To install Wireshark and TCPDump, capture client-server communication using UDP/TCP, and identify the transmitted data packets.

PROCEDURE

Step 1: Login to kali Linux(user:kali,password:kali)



Step 2: Open Terminal

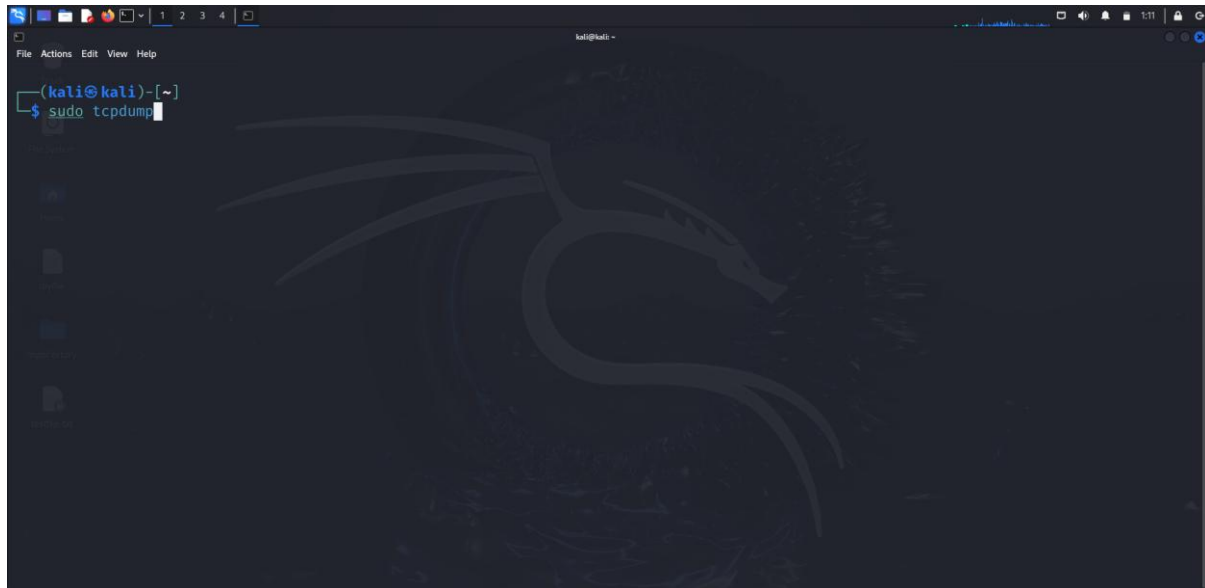
Step 3: Install tcpdump

COMMAND: sudo apt install tcpdump

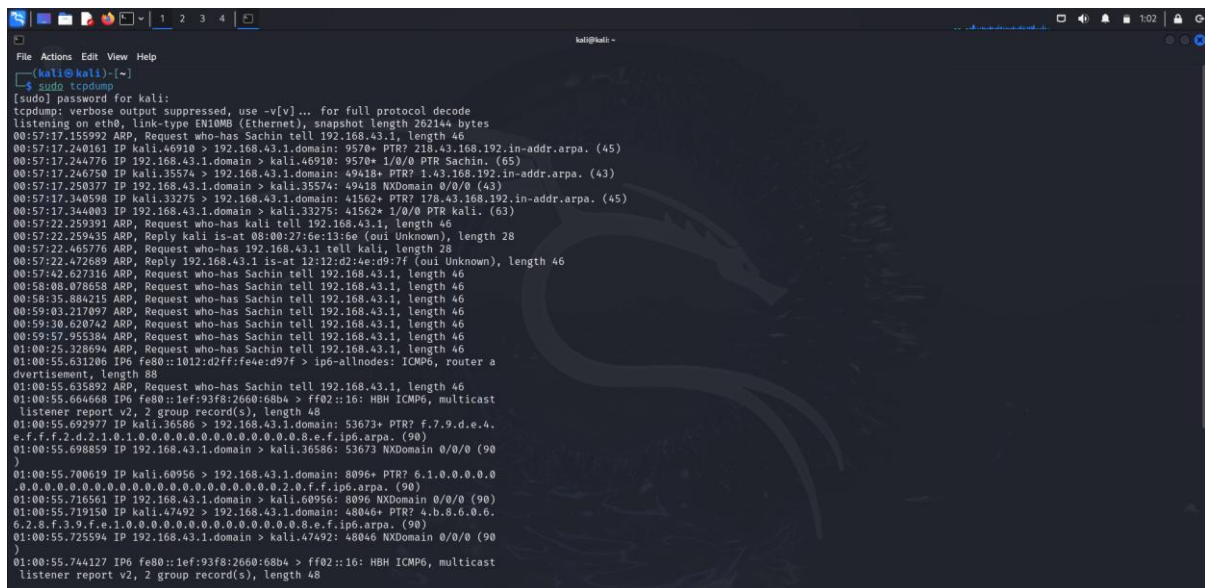


Step 4: Then type command tcpdump below

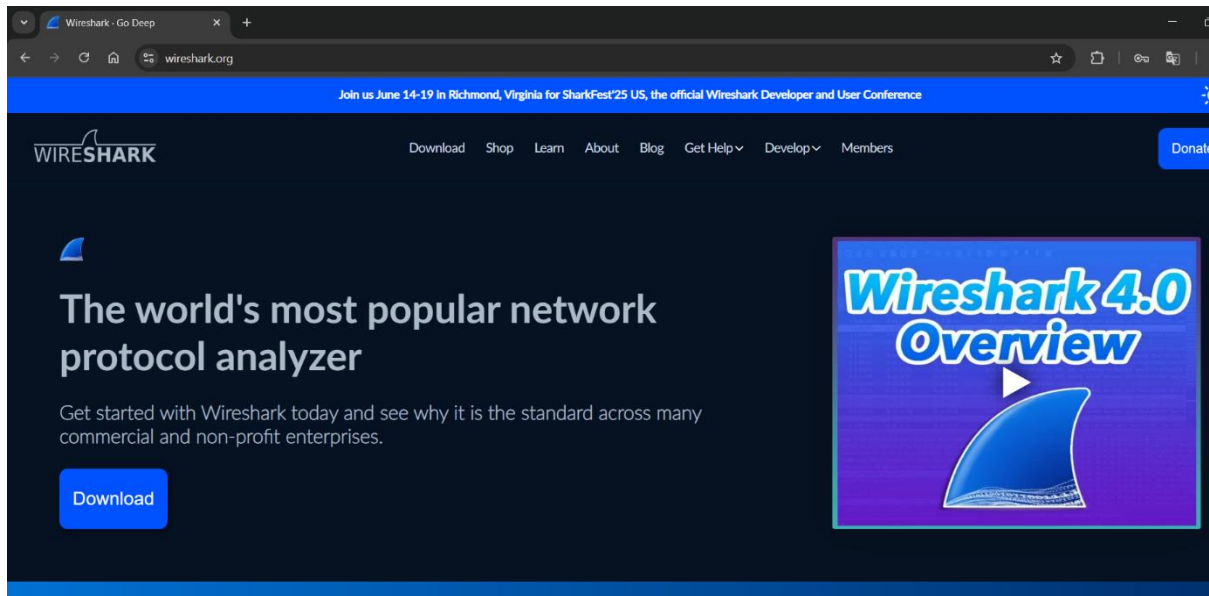
COMMAND : sudo tcpdump



Step 5: Watch the data traffic in the terminal



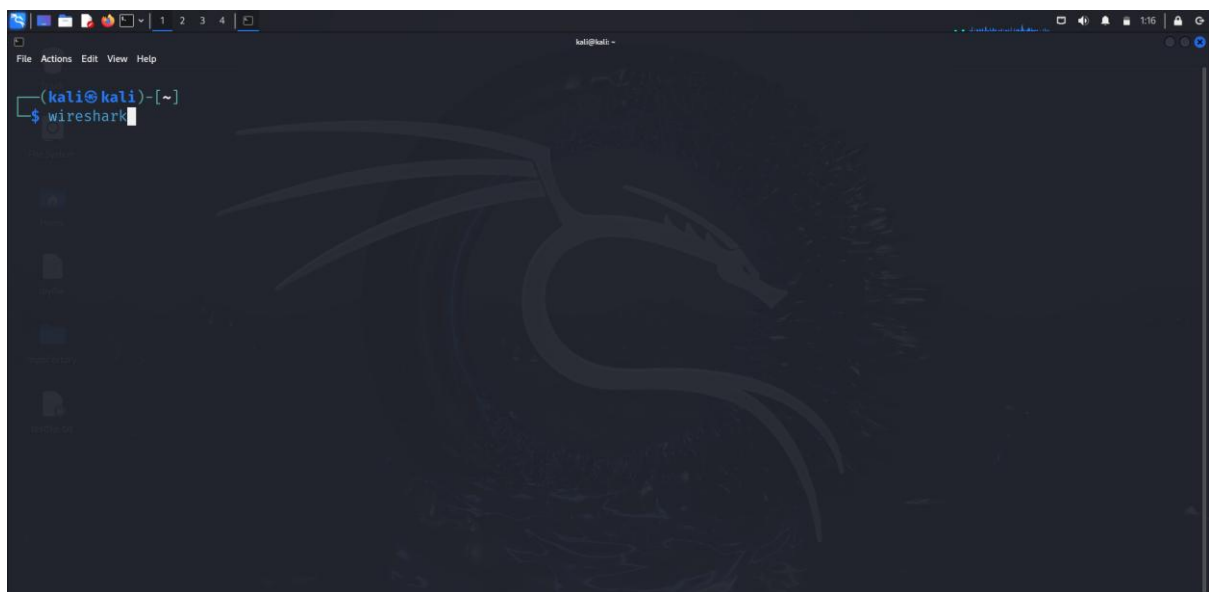
Step 6: Install wireshark visit the official website and follow the instruction



Click the download Button that your desktop support it

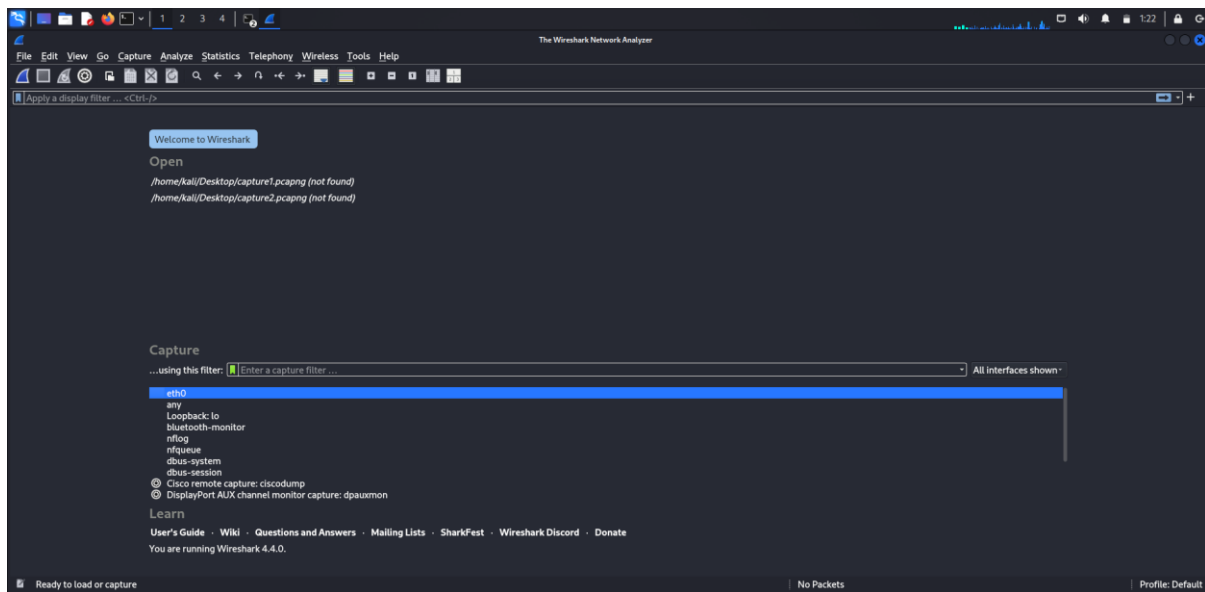
Step 7: To start capturing enter the command wireshark in terminal and run it

COMMAND: wireshark



This command open the wireshark in the kali linux

Step 8: click the start button to start capturing



RESULT

Wireshark and TCPDump were successfully installed. The captured network packets were analyzed, and the TCP/UDP datagrams were identified in the client-server communication.

Ex.No:05	Installation of rootkits and study of various options
Date:	

AIM

To install and use the **Chkrootkit** tool in Kali Linux to detect rootkits and unauthorized access attempts.

PROCEDURE

Step1: Begin by installing the Chkrootkit tool, which is available for Debian-based Linux distributions, including Kali Linux.

Step 2: Once the installation is complete, run the Chkrootkit tool to initiate a scan for potential rootkits present in the system.

Step 3: The results of the scan can be lengthy, so using a paginated view makes it easier to read and analyze the output.

Step 4: By default, Chkrootkit requires manual execution. However, to enable automatic daily scans, modify the configuration file and update the setting to allow scheduled scanning.

Step 5: Save the changes made in the configuration file to ensure the system runs automatic scans in the future.

Step 6: Finally, verify the successful execution of Chkrootkit and check whether any rootkits have been detected in the system.

Command

1. `sudo apt install chkrootkit -y` → Installs Chkrootkit on the system.
2. `sudo chkrootkit` → Scans the system for rootkits.
3. `sudo chkrootkit | less` → Displays scan results in a scrollable format.
4. `sudo nano /etc/chkrootkit.conf` → Opens the configuration file for editing.
5. `sudo chkrootkit -q` → Runs a scan in quiet mode, showing only infected files.

Output

After executing the Chkrootkit scan, the output will display the results of various security checks.

1. If no rootkits are found, the output may display: Searching for rootkit... No rootkits found.
2. If a potential threat is detected, it will show warnings such as: INFECTED: Possible rootkit detected.
3. Additional options like version checks and help commands will return relevant information about the tool's functionality.

Result

Chkrootkit was successfully installed and executed. The system was scanned for rootkits, ensuring enhanced security against unauthorized access and potential malware threats.

Ex.No:06	Intrusion Detection using SNORT in Kali Linux
Date:	

AIM

To set up and use **Snort**, an open-source **Intrusion Detection System (IDS)**, on **Kali Linux** to monitor and analyze network traffic for potential threats.

PROCEDURE

1. Install **Snort** on Kali Linux.
2. Configure **Snort** by setting up network variables in the configuration file.
3. Run Snort in **different modes**:
 - a. **Packet logging mode**
 - b. **Intrusion detection mode**
4. Create and add **custom Snort rules**.

Test Snort by generating network activity and checking logs

COMMANDS

Install Snort:

```
sudo apt update  
sudo apt install snort -y
```

Verify Installation:

```
snort -V
```

Configure Snort:

```
sudo nano /etc/snort/snort.conf
```

Modify the HOME_NET variable with your network range: ipvar HOME_NET 192.168.1.0/24

Test Snort Configuration:

```
sudo snort -T -c /etc/snort/snort.conf
```

Run Snort in Packet Logging Mode:

```
sudo snort -dev -l /var/log/snort/
```

Run Snort in Intrusion Detection Mode:

```
sudo snort -c /etc/snort/snort.conf -i eth0
```

(Replace eth0 with your actual network interface. Use ip a to check interfaces.)

Add Custom Snort Rules:

```
sudo nano /etc/snort/rules/local.rules
```

1. Add the rule: alert icmp any any -> any any (msg:"ICMP Packet Detected"; sid:1000001; rev:1;)

Restart Snort to Apply Rules:

```
sudo snort -q -c /etc/snort/snort.conf -i eth0
```

Generate Network Traffic (Ping Another System):

```
ping -c 3 <Target_IP>
```

Check Snort Alerts:

```
cat /var/log/snort/alert
```

Output:

1. Snort Version Check:

```
„_    -*> Snort! <*-  
o" )~  Version 2.x.x ""
```

2. Configuration Test Success:

Snort successfully validated the configuration! Snort exiting

3. ICMP Alert in Log File:

```
[**] [1:1000001:1] ICMP Packet Detected [**]
```

RESULT

Thus the above Intrusion Detection using SNORT in Kali Linux verified successfully.

Ex.No:07	Secure Data Storage, Secure Data Transmission, and Digital Signatures in Kali Linux
Date:	

Aim

To demonstrate methods for providing **secure data storage, secure data transmission, and digital signatures** using encryption and cryptographic tools in Kali Linux.

Procedure

1. Secure Data Storage:

- Encrypt the data using **GnuPG (GPG)** to protect files from unauthorized access.
- Use strong encryption algorithms to prevent data leaks.
- Decrypt the data whenever required using the correct passphrase or private key.

2. Secure Data Transmission:

- Use **OpenSSL** or **SSH** to encrypt data before transmitting it over the network.
- Establish a secure connection using protocols like **SCP (Secure Copy Protocol)** or **SFTP (Secure File Transfer Protocol)** to prevent interception.

3. Creating Digital Signatures:

- Generate a **private key** and use it to create a digital signature for a file.
- Verify the authenticity of the file using the corresponding **public key**.
- This ensures that the file has not been tampered with during transmission.

COMMAND

1. Secure Data Storage:

- a. `gpg -c file.txt` → Encrypts a file using GPG.
- b. `gpg file.txt.gpg` → Decrypts an encrypted file.

2. Secure Data Transmission:

- a. `scp file.txt user@remote_host:/destination/path/` → Securely transfers a file over SSH.
- b. `openssl enc -aes-256-cbc -salt -in file.txt -out file.enc` → Encrypts a file using OpenSSL.

3. Creating Digital Signatures:

- a. `gpg --output file.sig --sign file.txt` → Creates a digital signature.
- b. `gpg --verify file.sig file.txt` → Verifies the digital signature.

OUTPUT

- 1. **Encrypted File Output:** File 'file.txt.gpg' created (encrypted).
- 2. **Secure Transmission Output:** file.txt 100% transferred securely.
- 3. **Digital Signature Verification Output:** gpg: Signature made using RSA key
gpg: Good signature from "User Name"

Result

Successfully demonstrated **secure data storage** using encryption, **secure data transmission** using encrypted transfer protocols, and **digital signature creation and verification** to ensure file authenticity in Kali Linux.