BRAC UNIVERSITY

Inspiring Excellence

CSE496: Ethical Hacking

Fall 2024

Project Report

Project: Penetration testing on techEcommerce_project

Group: Zero Day Defenders	
ID	**Name**
21201613	MD. Shafiur Rahman
21201009	Md. Nafizur Rahman Bhuiya
21301186	Faisal Ahmed
23241088	Naima Nawar Achol
22101509	Maisa Tarannum Srizee

Table of Contents

Statement of Confidentiality

This document and the findings are the result of a penetration testing engagement conducted by Zero Day Defenders. All information, vulnerabilities and recommendations provided are strictly confidential and intended solely for the use of the authorized stakeholders of the techEcommerce_project.

Disclosure of any part of this document or the findings it contains to unauthorized parties is prohibited unless prior written consent is obtained from both Zero Day Defenders and the owners of the techEcommerce_project. Unauthorized access, use or distribution of this information may result in legal consequences. Zero Day Defenders remains committed to maintaining the highest standards of professionalism and confidentiality throughout this engagement.

Introduction

In the realm of cybersecurity, prevention starts with understanding vulnerabilities. This project follows a systematic penetration testing approach to identify and exploit possible vulnerabilities in a localhost PHP application. It uses common attack techniques to expose hidden flaws that could put at risk the application's security. These testing processes involve replicating real world attacks in a controlled environment to establish the scope of each vulnerability and its possible effect. The knowledge it creates from the testing not only points out the risks but also lays a basis for recommending relevant remedies. Closing the gap in vulnerability resolution, this research resonates with an active security response as the ultimate choice for preventing emerging cyberattacks on the website systems.

Executive Summary

Zero Day Defenders will perform a comprehensive penetration test of its internally facing network to identify security vulnerabilities, assess their potential impact and provide practical effective suggestions. The overall goal of this engagement was to assist TechEcommerce_Project in enhancing its security posture by discovering potential vulnerabilities that may allow unauthorized access, data breaches or disruption of operations.

Approach

Zero Day Defenders tested TechEcommerce_Project's internal security without any prior credentials or insider knowledge. The assessment was conducted remotely from a secure environment set up especially for this assignment.

A private testing strategy focused on detecting configuration errors, security vulnerabilities and other gaps. In depth documentation of each finding, its manual verification and evaluation were performed with the aim of studying possible exploitation scenarios and their impact.

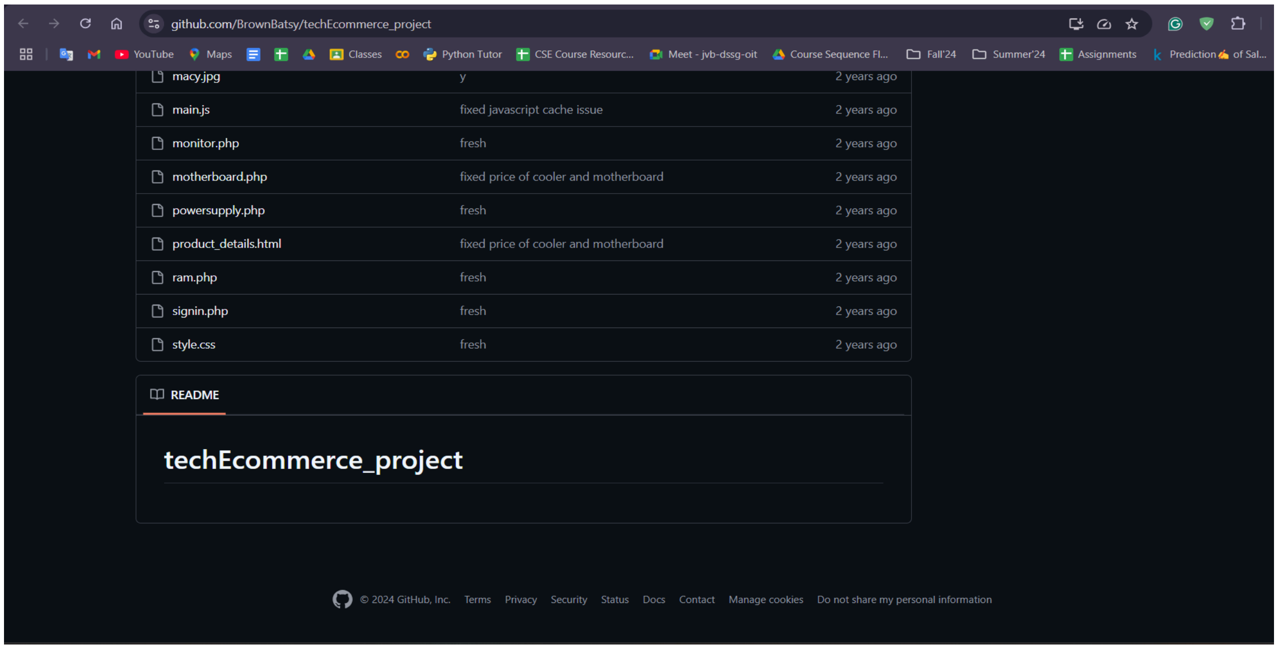
In order to demonstrate the possible consequences of a network breach, Zero Day Defenders conducted additional testing to evaluate lateral movement, gaining access and other technical attack scenarios in the event that an administrative position was gained in the network. Comprehensive findings, practical repair recommendations and guidelines for enhancing internal system safety for TechEcommerce_Project will all be included in the final product.

Project Setup

To set up the environment for running a PHP project, we first installed the local web server environment XAMPP which includes essential tools such as PHP. Once installed, we need to open XAMPP and start Apache and MySQL. After that, we placed our PHP project in the htdocs folder (default location: C:\xampp\htdocs on Windows or /opt/xampp/htdocs on Linux). Lastly, we need to go to a web browser and on a new tab, we have to write localhost/project_name. In our case, it is: http://localhost/Project%20E-commerce%20Tech

Project Bug Handling

1. At first, during the setup and review of the PHP project environment, it was observed that no database files (e.g., `.sql` dumps) were provided. This indicates that the project is either designed to function without a database or there has been an oversight in including necessary database files or documentation. The project files do not contain any `.sql` files for database import. If the absence of a database is unintentional, the core features of the project might not function as expected.



github.com/BrownBatsy/techEcommerce_project

YouTube Maps Classes Python Tutor CSE Course Resourc... Meet - jvb-dssg-oit Course Sequence Fl... Fall'24 Summer'24 Assignments Prediction of Sal...

macy.jpg	y	2 years ago
main.js	fixed javascript cache issue	2 years ago
monitor.php	fresh	2 years ago
motherboard.php	fixed price of cooler and motherboard	2 years ago
powersupply.php	fresh	2 years ago
product_details.html	fixed price of cooler and motherboard	2 years ago
ram.php	fresh	2 years ago
signin.php	fresh	2 years ago
style.css	fresh	2 years ago

README

techEcommerce_project

© 2024 GitHub, Inc. Terms Privacy Security Status Docs Contact Manage cookies Do not share my personal information

Prevention:

After thoroughly reviewing the project code, we identified the database structure and manually generated the required database based on the application's logic and functionality. By analyzing the code for database queries, table references and data handling processes, we reconstructed the schema, including table names, columns and relationships ensuring alignment with the project’s requirements. This manual generation allowed us to replicate the intended database environment accurately, enabling the application to function as expected. Moving forward, we recommend documenting the database schema and providing an exportable `techdata.sql` file to streamline future deployments and reduce setup complexities.

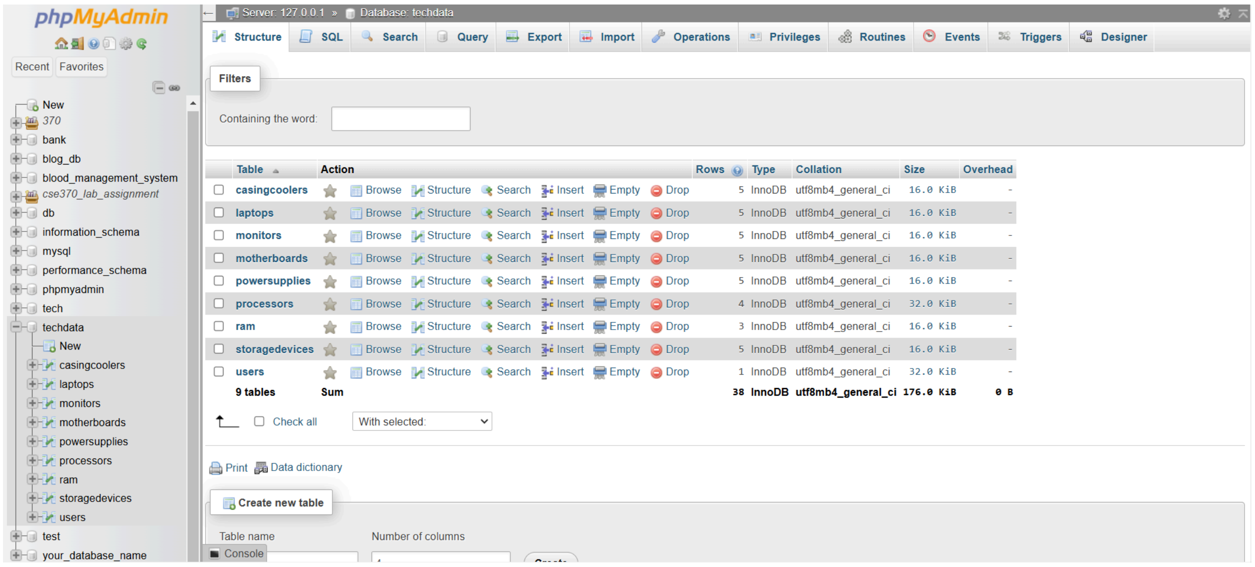
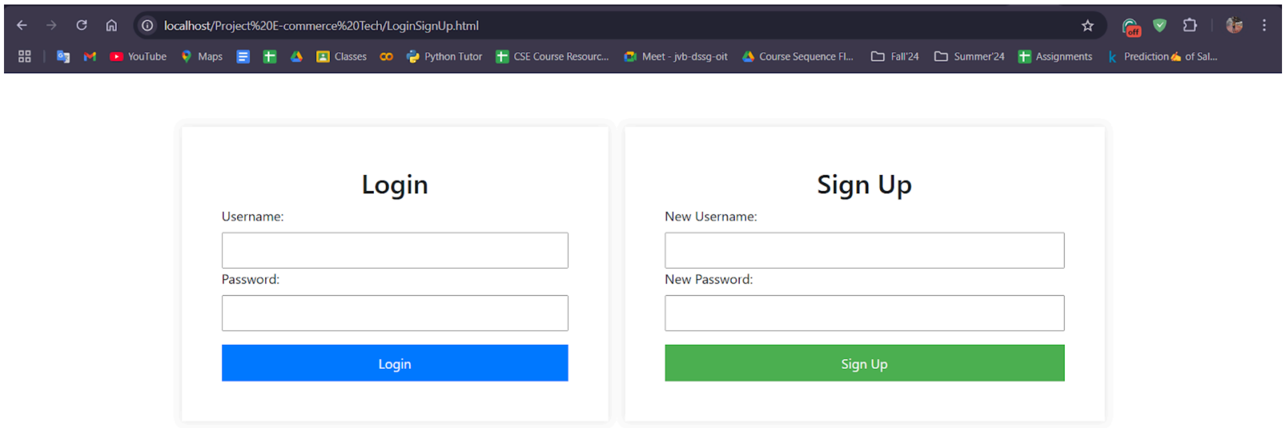


Table	Action							Rows	Type	Collation	Size	Overhead
casingcoolers	☆	Browse	Structure	Search	Insert	Empty	Drop	5	InnoDB	utf8mb4_general_ci	16.0 KiB	-
laptops	☆	Browse	Structure	Search	Insert	Empty	Drop	5	InnoDB	utf8mb4_general_ci	16.0 KiB	-
monitors	☆	Browse	Structure	Search	Insert	Empty	Drop	5	InnoDB	utf8mb4_general_ci	16.0 KiB	-
motherboards	☆	Browse	Structure	Search	Insert	Empty	Drop	5	InnoDB	utf8mb4_general_ci	16.0 KiB	-
powersupplies	☆	Browse	Structure	Search	Insert	Empty	Drop	5	InnoDB	utf8mb4_general_ci	16.0 KiB	-
processors	☆	Browse	Structure	Search	Insert	Empty	Drop	4	InnoDB	utf8mb4_general_ci	32.0 KiB	-
ram	☆	Browse	Structure	Search	Insert	Empty	Drop	3	InnoDB	utf8mb4_general_ci	16.0 KiB	-
storagedevices	☆	Browse	Structure	Search	Insert	Empty	Drop	5	InnoDB	utf8mb4_general_ci	16.0 KiB	-
users	☆	Browse	Structure	Search	Insert	Empty	Drop	1	InnoDB	utf8mb4_general_ci	32.0 KiB	-
9 tables	**Sum**							**38**	**InnoDB**	**utf8mb4_general_ci**	**176.0 KiB**	**0 B**

2. After carefully reviewing the project code and manually generating the required database by analyzing the application's logic, we attempted to run the project. However, the project ran but most of the functionalities didn't work properly because of numerous mistakes and inconsistencies in the code. These errors include syntax issues, undefined variables, missing function calls and incorrect handling of database queries all of which hindered the proper interaction between the code and the manually created database. Despite resolving the database setup, the project could not operate as intended due to these coding flaws.



Login

Username:

Password:

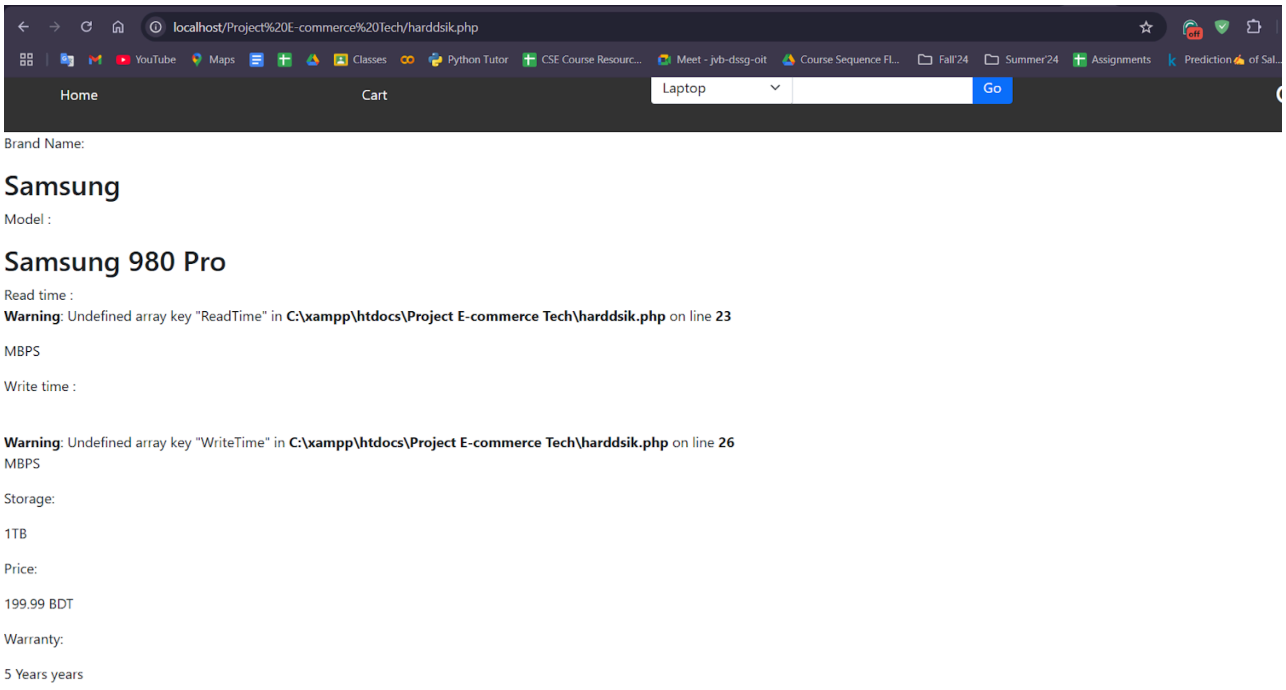
Login

Sign Up

New Username:

New Password:

Sign Up



Brand Name:

Samsung

Model :

Samsung 980 Pro

Read time :
Warning: Undefined array key "ReadTime" in **C:\xampp\htdocs\Project E-commerce Tech\harddsik.php** on line **23**

MBPS

Write time :

Warning: Undefined array key "WriteTime" in **C:\xampp\htdocs\Project E-commerce Tech\harddsik.php** on line **26**
MBPS

Storage:

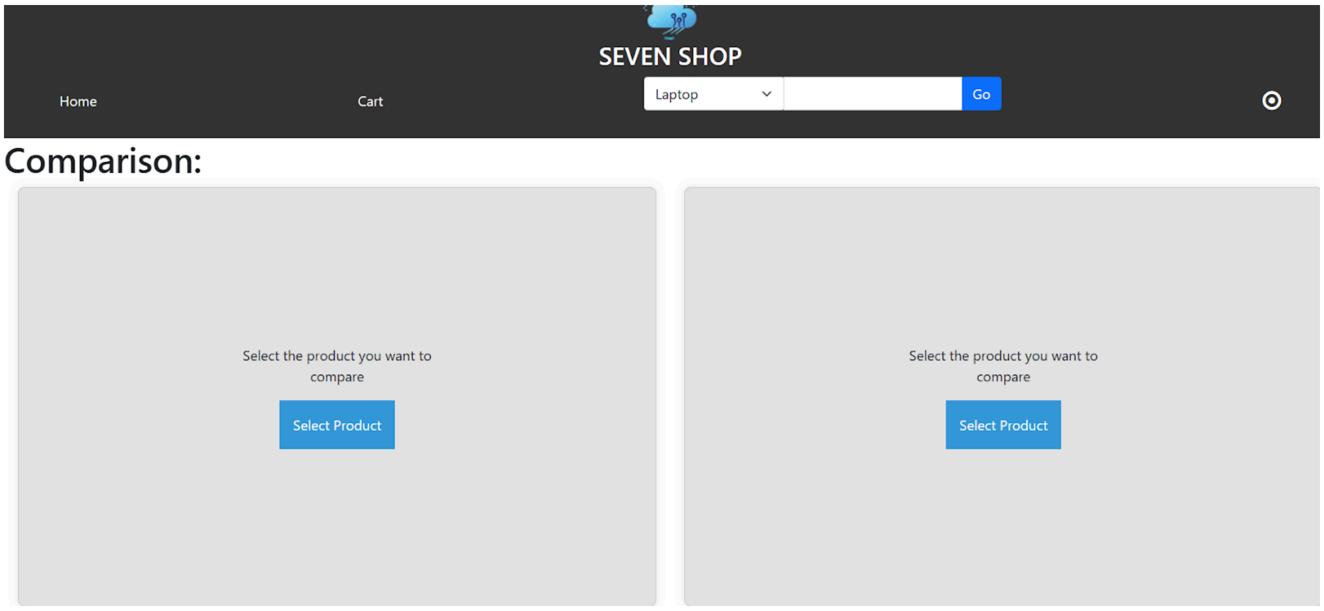
1TB

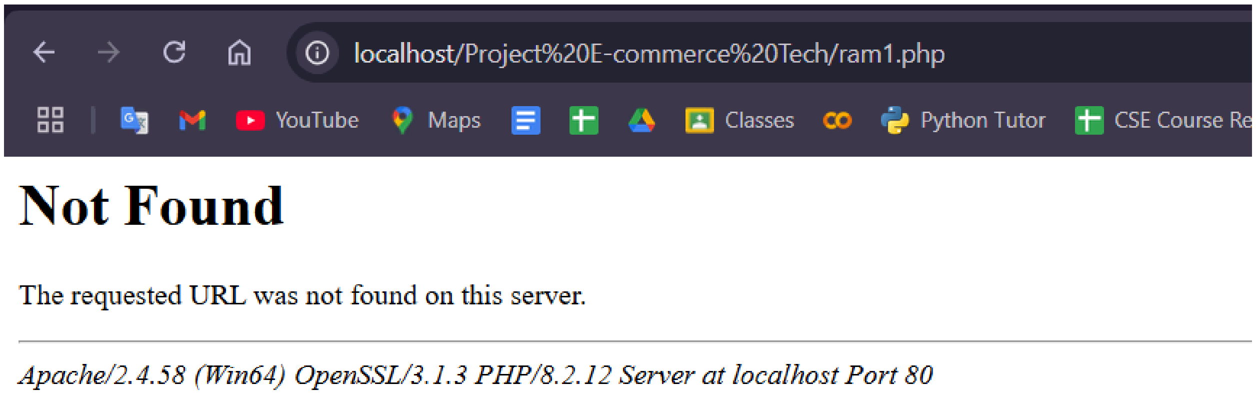
Price:

199.99 BDT

Warranty:

5 Years years


SEVEN SHOP
Home
Cart
Laptop
Go
Comparison:
Select the product you want to compare
Select Product
Select the product you want to compare
Select Product



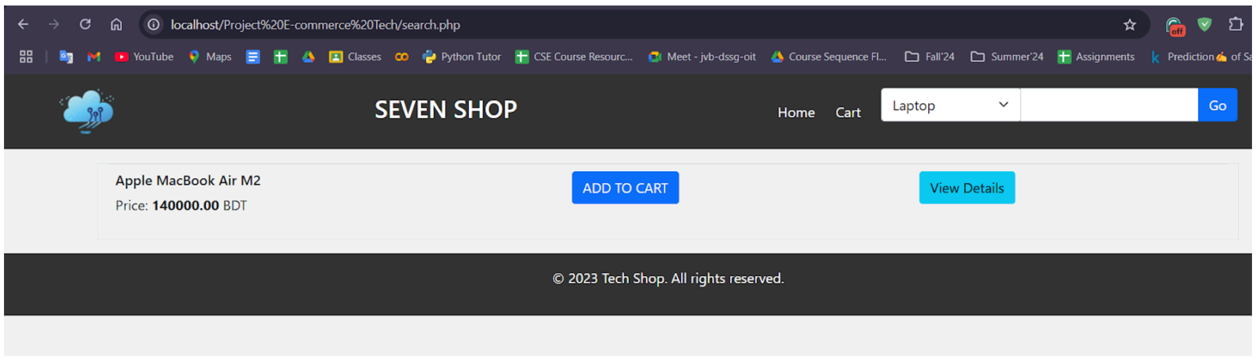
Not Found

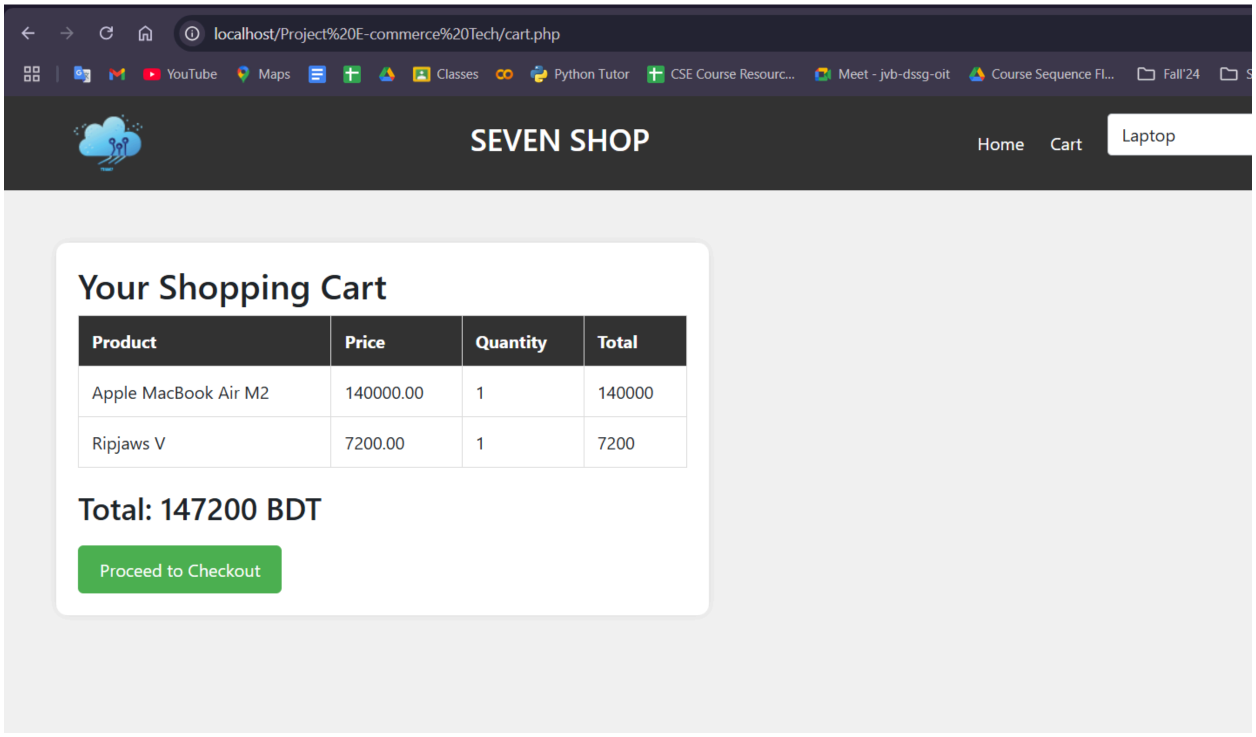
The requested URL was not found on this server.

Apache/2.4.58 (Win64) OpenSSL/3.1.3 PHP/8.2.12 Server at localhost Port 80

Prevention:

To address the issues and ensure the successful execution of the project, we conducted a comprehensive code review, identifying and resolving syntax errors, logical flaws and inconsistencies within the codebase. The code was refactored to align with best practices, improving its readability, maintainability and overall structure. We implemented a robust testing framework including unit tests to validate individual components and integration tests to ensure seamless interaction between modules. Additionally, debugging processes were optimized and errors were closely monitored to resolve issues effectively. Despite our efforts in reviewing, refactoring and testing the code, we were unable to successfully run compare.php due to its higher complexity and deeply rooted issues. The file contained intricate logic and dependencies that required significant restructuring, which exceeded the scope of our current efforts.




localhost/Project%20E-commerce%20Tech/cart.php

SEVEN SHOP

Home Cart Laptop

Your Shopping Cart

Product	Price	Quantity	Total
Apple MacBook Air M2	140000.00	1	140000
Ripjaws V	7200.00	1	7200

Total: 147200 BDT

Proceed to Checkout

Attack Summary

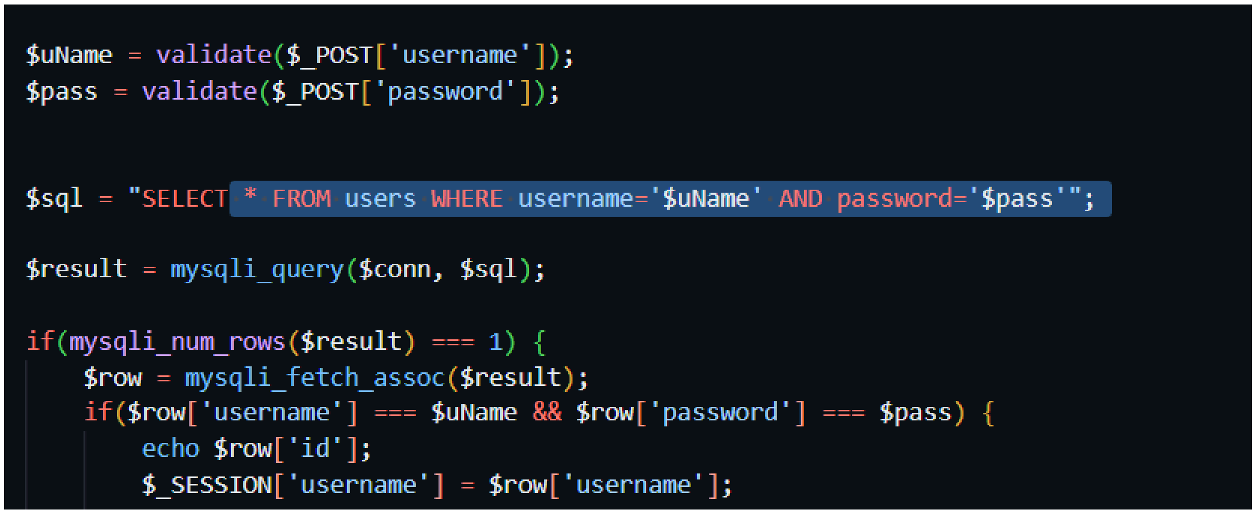
Cyber-attacks are the malicious attempts by individuals or organizations to penetrate through devices, networks or systems. These attacks often aim at unauthorized access, disruption of services or theft of confidential information. Understanding the different attack types and the good defense and protection of digital assets in this fast-evolving world of cybersecurity depend basically on that. A few basic categories of cyberattacks are listed here.

1. SQL Injection Attack

SQL injection is a kind of input manipulation attack whereby a web application is filled with SQL statements with the appropriate purpose of modifying or procuring unauthorized access to a database.

Working Principle:

SQL Injection is a technique used by hackers to inject malicious SQL code via input fields of an application or URLs because of poor input validation. If the database executes the injected code, it is most likely to alter database content, allow unauthorized access to private information or bypass authentication procedures. This attack manipulates queries by exploiting dynamic SQL construction errors. Most of them affect programs that don't use prepared statements or parameterized queries.



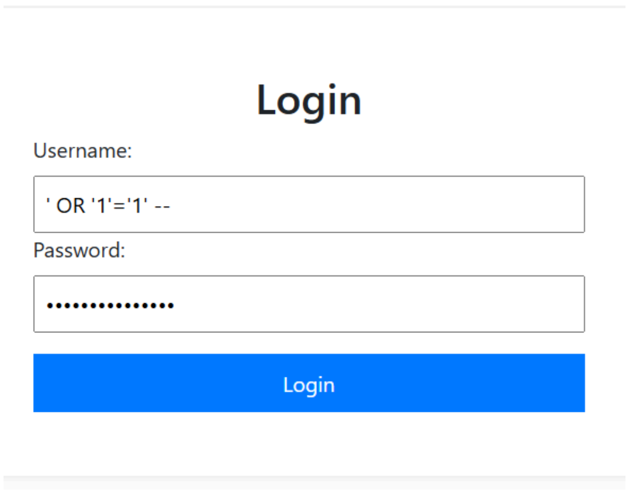
```
$uName = validate($_POST['username']);
$pass = validate($_POST['password']);


$sql = "SELECT * FROM users WHERE username='$uName' AND password='$pass'";

$result = mysqli_query($conn, $sql);

if(mysqli_num_rows($result) === 1) {
    $row = mysqli_fetch_assoc($result);
    if($row['username'] === $uName && $row['password'] === $pass) {
        echo $row['id'];
        $_SESSION['username'] = $row['username'];
```

Here in this part of the code, we can clearly see that there is an SQL injection vulnerability. If we insert ' OR 1=1 -- in the username field, then the $sql turns True and the user gets in.

Attack Methodology:

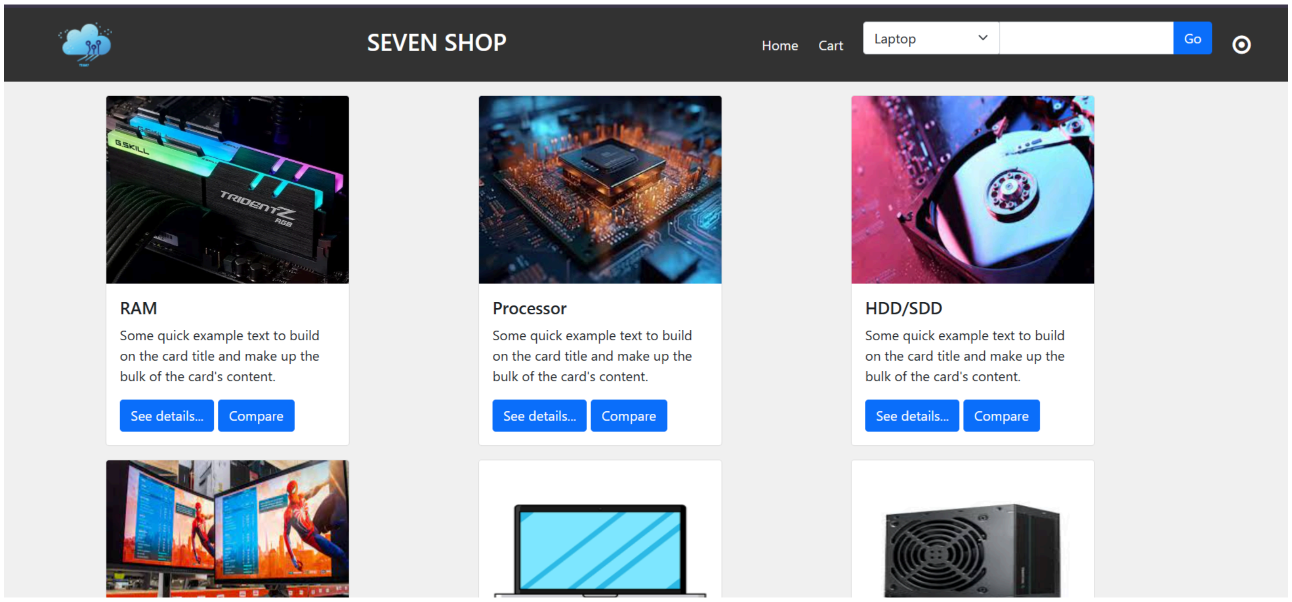
Here in the login page, we simply typed ' OR 1=1 -- in the username field. This input manipulated the SQL query to always return true for the authentication check. When the query was processed after we clicked the login button, the application executed the manipulated query, bypassed authentication, and granted us access to the restricted area without requiring valid credentials.

Attack Walkthrough



Username: ' OR '1'='1' --

Password: •••••••••••••••


SEVEN SHOP
Home
Cart
Laptop
Go
RAM
Some quick example text to build on the card title and make up the bulk of the card's content.
See details...
Compare
Processor
Some quick example text to build on the card title and make up the bulk of the card's content.
See details...
Compare
HDD/SDD
Some quick example text to build on the card title and make up the bulk of the card's content.
See details...
Compare

Prevention

Secure Password Hashing:

Password hashing should be done with strong hash algorithms such as bcrypt or Argon2. These algorithms are slow and have incorporated a certain amount of protection against such attacks as brute force ones.

Salting:

Before hashing passwords include a unique salt for each of them to prevent two similar passwords from having the same hash.

2. Data Tampering Attack

Data tampering refers to a cyber act of interference with data either while it is in transit or in storage. This undermines the credibility of data. The system may be used for illegitimate performances or data tampering, or for unauthorized control or modification of the data.

Working Principle

An attacker snoops data under transmission between two systems, changes it and then forwards the data to the intended destination. For instance, altering the value in a financial transaction or altering some data in a file while transferring. The attacker is able to read stored data and manipulate the contents of the data by modifying the content of files such as configuration files, modifying records in a database or injecting scripts in files.

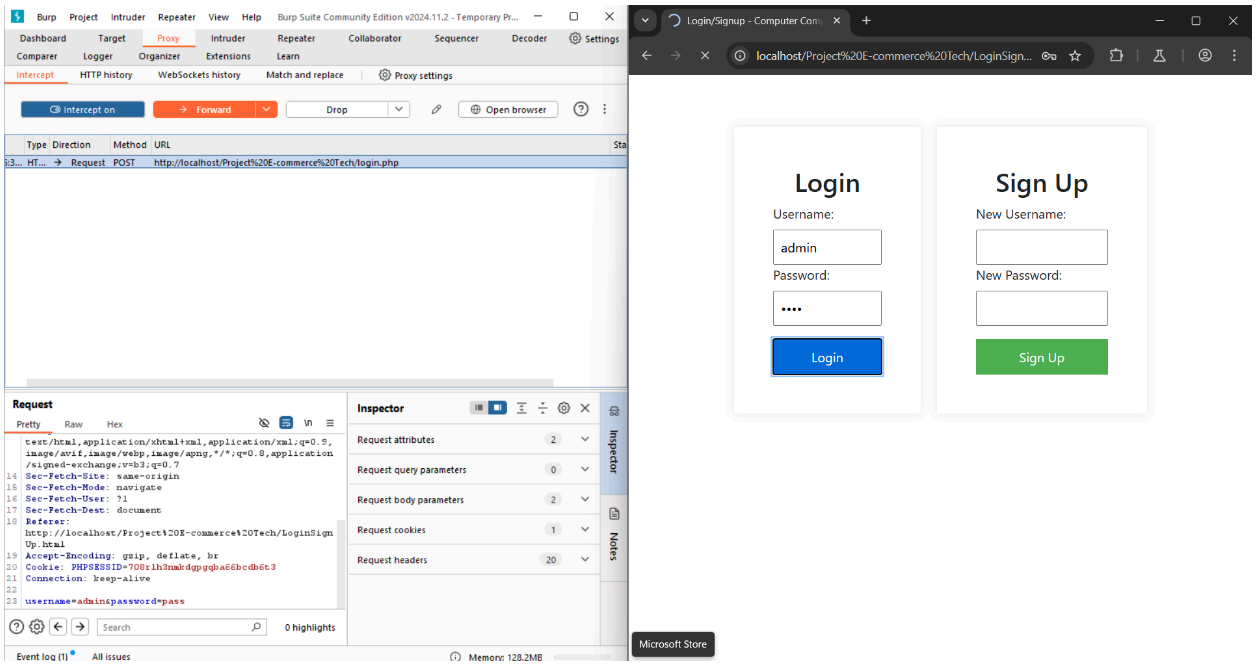
Attack Methodology

Here in the login page, we intercepted the HTTP POST request containing the parameters login=admin and password=pass using a tool like Burp Suite. This allowed us to manipulate the data being sent to the server. We tampered with the login parameter, changing its value from admin to other_user to test unauthorized access to another account. The manipulated request became:

```
username=other_user&password=1234
```

When the server processed this tampered request, it allowed access to the account of other_user, demonstrating that the application lacked proper authentication checks and was vulnerable to data tampering. This attack confirmed that sensitive parameters like login were not validated or protected against manipulation.

Attack Walkthrough

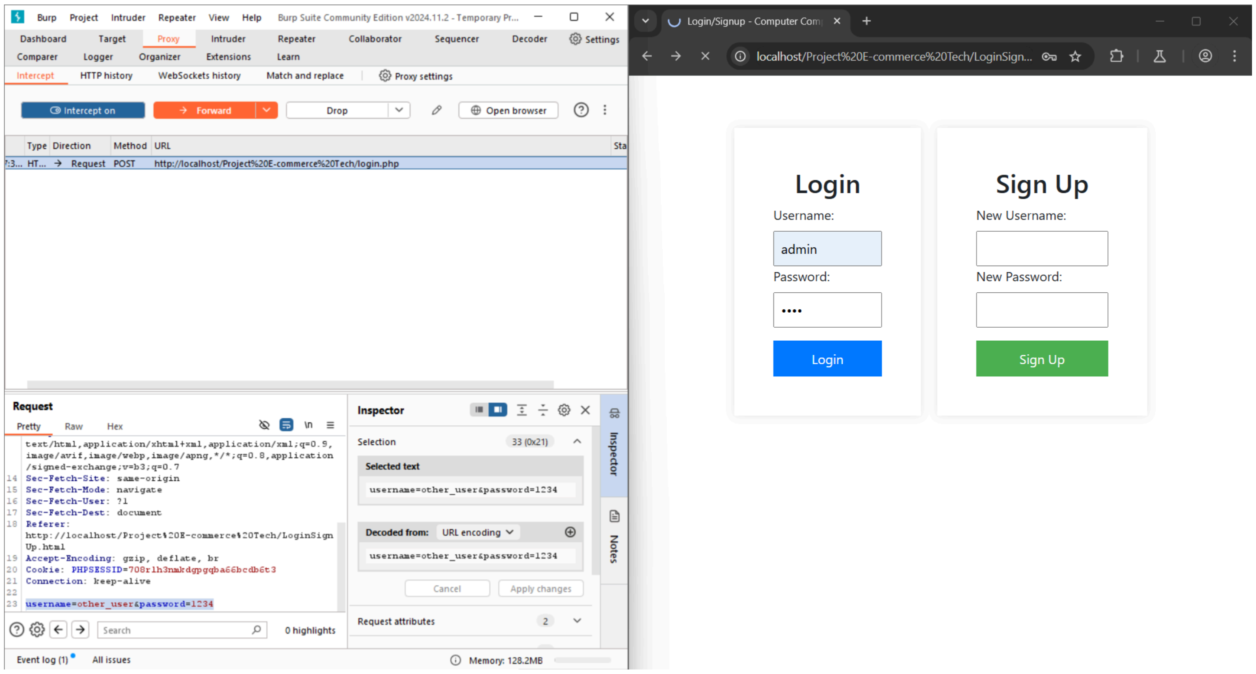


```
POST http://localhost/Project%20E-commerce%20Tech/login.php
```

```
text/html,application/xhtml+xml,application/xml;q=0.9,
image/avif,image/webp,image/apng,*/*;q=0.8,application
/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer:
http://localhost/Project%20E-commerce%20Tech/LoginSign
Up.html
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=708r1h3nmkdgpgqba66bcdb6t3
Connection: keep-alive

username=admin&password=pass
```

Username: admin

Password: ••••



```
http://localhost/Project%20E-commerce%20Tech/login.php

text/html,application/xhtml+xml,application/xml;q=0.9,
image/avif,image/webp,image/apng,*/*;q=0.8,application
/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer:
http://localhost/Project%20E-commerce%20Tech/LoginSign
Up.html
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=708r1h3nmkdgpgqba66bcdb6t3
Connection: keep-alive

username=other_user&password=1234
```

Selected text: username=other_user&password=1234

Decoded from: URL encoding: username=other_user&password=1234

Prevention

Encrypt Data in Transit:

Encrypt your data before sending through protocols such as TLS or HTTPS. Integrate encryption with another security layer which is the integrity checks including MAC.

Encrypt Data at Rest:

Ensure data kept in storage is well protected with sound formats of encryption such as the Advanced Encryption Standard AES256. Encrypt keys differently and independently from the stored data.

Implement File Integrity Monitoring (FIM):

Some measures include how to track changes even on files or databases that one is not supposed to modify. Raise notifications to the administrators once there are changes that have been made.

3. Replay Attack

A replay attack occurs when an attacker captures valid data messages and relays them to perform some actions he is not authorized to do. For example, using an authentication token to gain access to a system.

Working Principle

An attacker captures valid data packets transmitted between two systems, stores them and retransmits the data to the intended destination at a later time without modifying its content. For instance, replaying an intercepted login request or a financial transaction to perform unauthorized actions. The attacker does not necessarily need to understand the content of the intercepted data but exploits the system's inability to distinguish between original and replayed requests. By resending authentication tokens, transaction IDs or encrypted messages. The attacker can impersonate users, repeat transactions or gain unauthorized access to restricted areas, causing financial loss or breaching system integrity.

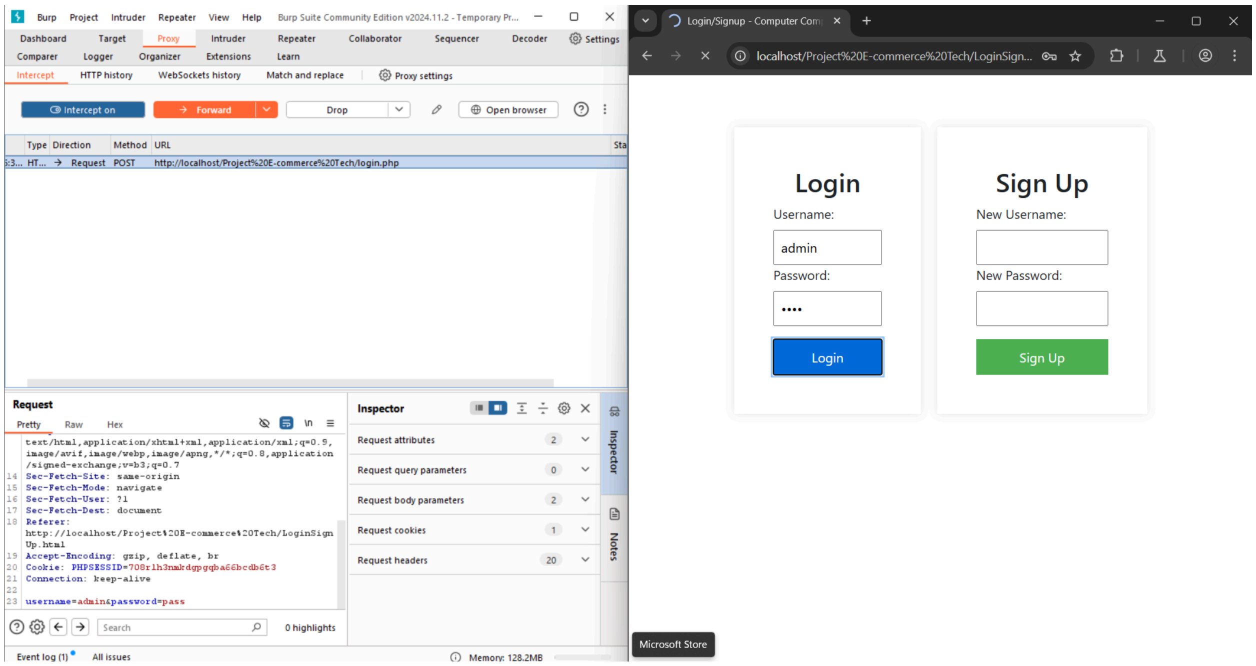
Attack Methodology

Here in the login page, we intercepted the HTTP POST request containing the parameters login=admin and password=admin using a tool like Burp Suite. We captured this valid authentication request which was sent to the server during a successful login attempt. Instead of modifying the intercepted data, we replayed the exact same request to the server multiple times by resending it through the intercepting tool. The replayed request became:

```
login=admin&password=admin&security_level=0&form=submit
```

When the server processed this replayed request, it granted access each time without verifying whether the request had already been used or expired. This demonstrated that the application was vulnerable to replay attacks as it failed to implement measures like session validation, token expiration or request uniqueness to prevent duplicate access attempts. This confirmed that intercepted data could be reused maliciously to bypass authentication mechanisms.

Attack Walkthrough



```
text/html,application/xhtml+xml,application/xml;q=0.9,
image/avif,image/webp,image/apng,*/*;q=0.8,application
/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer:
http://localhost/Project%20E-commerce%20Tech/LoginSign
Up.html
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=708r1h3nmkdgpgqba66bcdb6t3
Connection: keep-alive

username=admin&password=pass
```

Prevention

Unique Initialization Vector (IV):

IV should be used only once, for the same plaintext, to have different cipher text values.

Replay Detection:

One must employ nonces, timestamps or sequence numbers to help avoid recognizing or admitting old duplicate or duplicate requests.

4. Brute Force Attack

A brute force attack performs an organized try of every possible key, password or credential with the intent to gain unauthorized access to a system. It is basically a trial-and-error process to crack the login credentials.

Working Principle

An attacker uses automated tools like Burp Suite's Intruder feature to perform a brute force attack by systematically trying different combinations of usernames and passwords. By utilizing pre-built wordlists from resources like SecLists, the attacker targets the login page of the application. The attack begins by intercepting a valid HTTP POST request containing parameters like username and password. The attacker sets the username and password fields as payload positions in Burp Suite Intruder and loads a wordlist to iterate through multiple combinations. When a request matches valid credentials, the server responds with a success message, granting access to the attacker. This method exploits weak or predictable credentials, demonstrating the application's vulnerability to brute force attacks and lack of proper security measures such as account lockout policies or CAPTCHA.

Attack Methodology

In the brute force attack, we intercepted the HTTP POST request containing the parameters username=admin and password=password123 using Burp Suite. This allowed us to manipulate the data being sent to the server. We then configure Burp Suite’s Intruder to automate the attack by testing different combinations of usernames and passwords from a pre-built wordlist such as SecLists. We set the username and password fields as payload positions and loaded the wordlist to generate various combinations.

The request initially looked like: username=admin&password=admin&Login=Submit

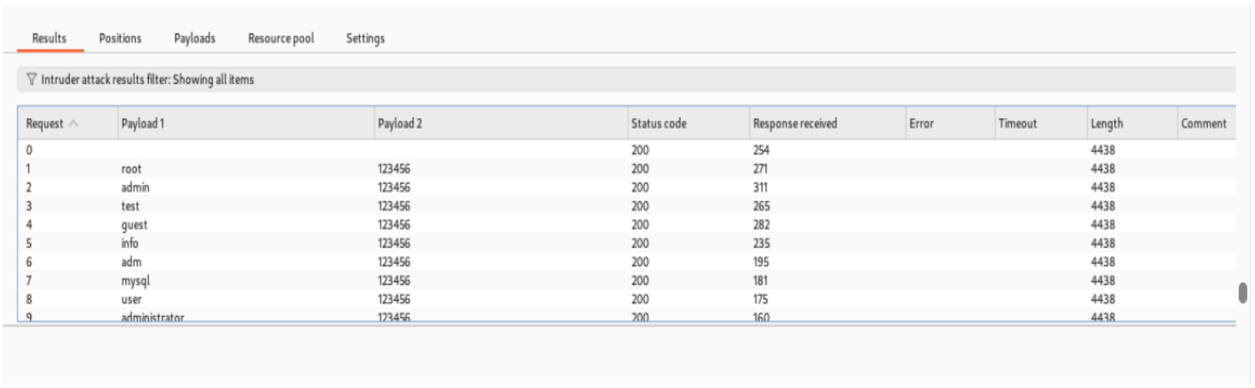
Burp Suite's Intruder then iterated through multiple combinations, testing requests such as: username=admin&password=admin123 username=admin&password=admin2021

After several attempts, we found that the combination username=admin&password=admin resulted in a successful login. This demonstrated that the application was vulnerable to brute force attacks due to weak or predictable credentials and it lacked proper protections like account lockout policies or CAPTCHA. This attack confirmed that the application was not resilient to automated credential testing, allowing unauthorized access once the correct combination was discovered.

Here is the source of seclist that we used for the brute force attack:

https://github.com/danielmiessler/SecLists

Attack Walkthrough



Intruder attack results filter: Showing all items

Request	Payload 1	Payload 2	Status code	Response received	Error	Timeout	Length	Comment
0			200	254			4438	
1	root	123456	200	271			4438	
2	admin	123456	200	311			4438	
3	test	123456	200	265			4438	
4	guest	123456	200	282			4438	
5	info	123456	200	235			4438	
6	adm	123456	200	195			4438	
7	mysql	123456	200	181			4438	
8	user	123456	200	175			4438	
9	administrator	123456	200	160			4438	

Prevention

Strong Encryption:

AES-256 should be used due to the deterrent of key guessing through mathematical computation.

Rate Limiting:

There has to be measures put in place that allow for a certain number of attempts one is allowed to log in and if they exceed that, then they should make the account inactive for a certain amount of time.

5. Data Disclosure Attack

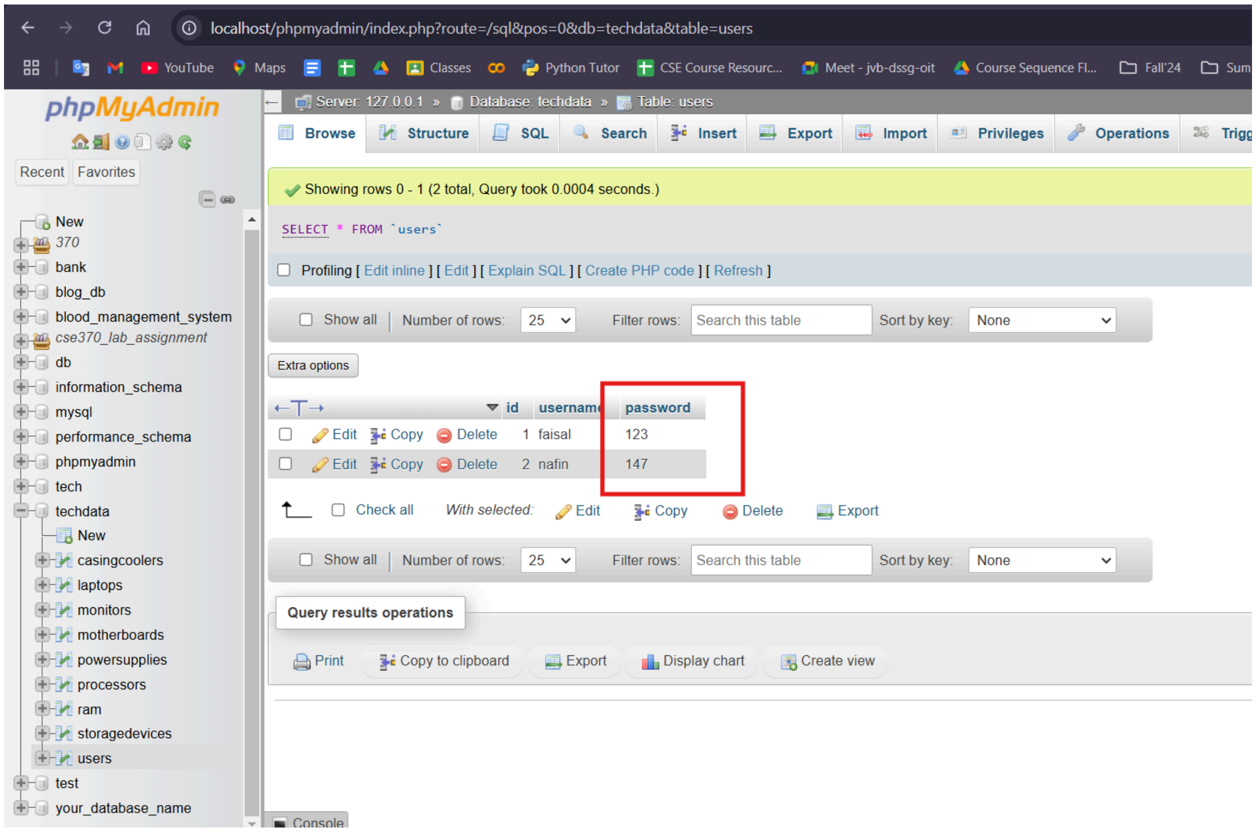
When sensitive information is sent or maintained without encryption or appropriate access controls, it may become vulnerable to interception by attackers, allowing them to gain unauthorized access to private data, manipulate it, or use it maliciously. This can lead to severe consequences including data breaches, identity theft and loss of user trust.

Working Principle

An attacker performs a Data Disclosure Attack by exploiting vulnerabilities in an application where sensitive data is improperly exposed to unauthorized users. The attack begins by interacting with a feature and intercepting the server's response using tools like Burp Suite. Upon analyzing the intercepted response, the attacker identifies sensitive information such as personal details, login credentials or financial data that is not properly secured. The attacker then accesses this data without any restrictions or encryption, highlighting the application's failure to implement proper access controls or data protection mechanisms. This exposes the application to unauthorized data disclosure, potentially leading to identity theft or other privacy breaches.

Attack Methodology

In the case of a Data Disclosure Attack, we found that the application did not restrict the access of sensitive data to unauthorized users. Interacting with any feature from the website with the help of a tool like Burp Suite, we intercept the response from the server, which includes sensitive data in the form of usernames, email addresses or even financial information not properly secured or encrypted. After having captured the response of the application, it was noticed that there was no encryption or access control applied, thereby revealing the sensitive information. This will show that the applications are vulnerable to disclosure of the information whose unauthorized access cannot be restricted owing to a lack of controls in the application.


localhost/phpmyadmin/index.php?route=/sql&pos=0&db=techdata&table=users
phpMyAdmin
Server: 127.0.0.1 » Database: techdata » Table: users
Browse Structure SQL Search Insert Export Import Privileges Operations
Recent Favorites
New
370
bank
blog_db
blood_management_system
cse370_lab_assignment
db
information_schema
mysql
performance_schema
phpmyadmin
tech
techdata
New
casingcoolers
laptops
monitors
motherboards
powersupplies
processors
ram
storagedevices
users
test
your_database_name
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None
Extra options
Check all With selected: Edit Copy Delete Export
Query results operations
Print Copy to clipboard Export Display chart Create view
Console

Showing rows 0 - 1 (2 total, Query took 0.0004 seconds.)

```
SELECT * FROM `users`
```

	id	username	password
Edit Copy Delete	1	faisal	123
Edit Copy Delete	2	nafin	147

Prevention

Secure Key Storage: Keep keys in safe places, like cloud key management systems or hardware security modules (HSM).

Access Controls: Key access to authorized people or processes by implementing access controls

Encryption of Keys: Use password-based encryption or a master key to encrypt keys.

Key Rotation: Regularly rotate and update keys to limit exposure if compromised.

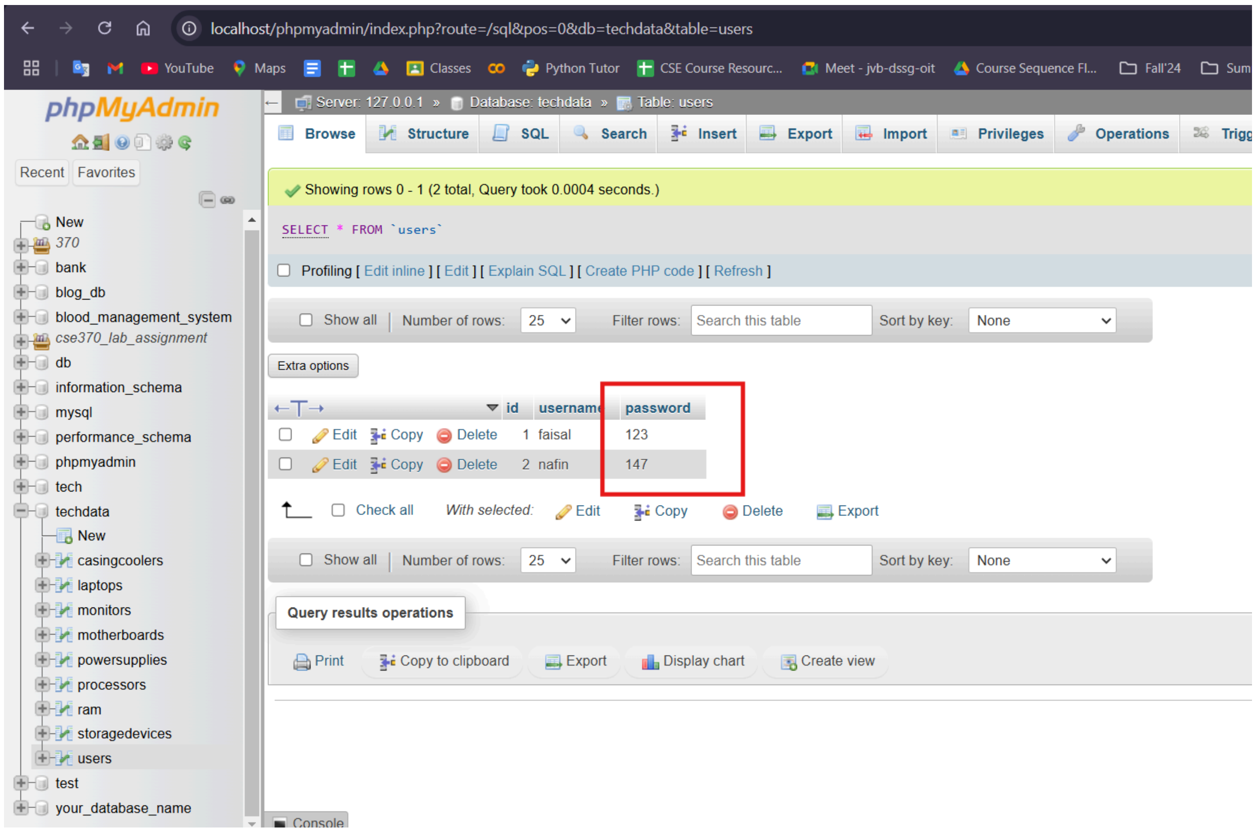
6. Insider Attack

When trusted employees abuse their access to systems to steal information, harm the company or enable outside attacks. This is known as an insider attack.

Working principle

An Insider Attack occurs when an individual with authorized access to a system such as an employee or contractor intentionally or unintentionally exploits their privileges to harm the organization or its users. The attack begins when the insider abuses their access rights to obtain sensitive information, alter data or disrupt normal operations. This can involve stealing proprietary data, misusing access to confidential records or bypassing security measures to cause damage. Since the insider already has legitimate access, traditional security systems may not detect the attack making it difficult to prevent or mitigate. The attack highlights the risk posed by trusted individuals within an organization, demonstrating the need for stringent monitoring, access controls and employee training to protect sensitive information from internal threats.

Attack Walkthrough



localhost/phpmyadmin/index.php?route=/sql&pos=0&db=techdata&table=users

Server: 127.0.0.1 » Database: techdata » Table: users

Showing rows 0 - 1 (2 total, Query took 0.0004 seconds.)

```
SELECT * FROM `users`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

	id	username	password
Edit Copy Delete	1	faisal	123
Edit Copy Delete	2	nafin	147

Prevention

Encrypt Sensitive Data: Make sure that important information is encrypted so that even insiders cannot access it without the right decryption keys.

Access Least Privilege: Restrict access privileges to just those required for a user's role.

Monitor Logs: Monitor and analyze logs for unusual activity or illegal access attempts.

Audit Permissions: Verify role-based access controls (RBAC) and user permissions on a regular basis.

Security Awareness: Conduct insider risk training to educate employees about ethical responsibilities.

7. IDOR Attack

When an application grants direct access to objects (such as files, database entries, or resources) based on user-provided input without first verifying user authorization, it is known as an IDOR access control vulnerability.

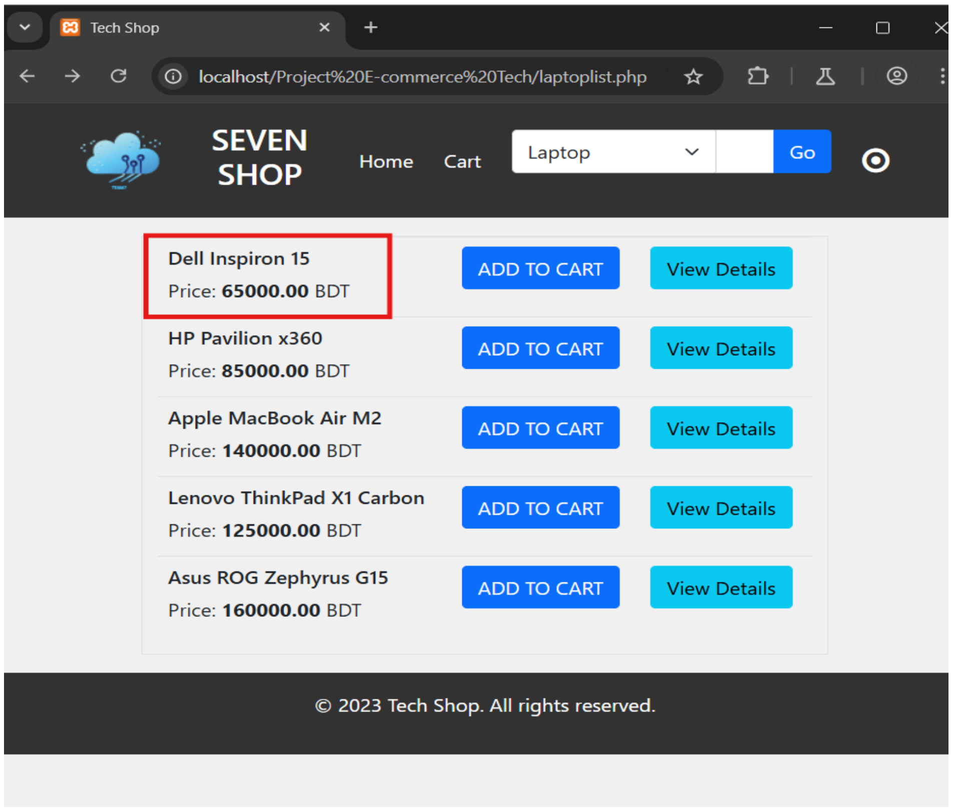
Working principle

An IDOR (Insecure Direct Object Reference) Attack occurs when an application fails to properly validate user authorization, allowing attackers to access or manipulate resources they are not authorized to. The attack begins when the attacker intercepts a request using a tool like Burp Suite such as viewing the details of a product with a specific ID. By manipulating the intercepted request, the attacker changes the product ID to another value such as altering the cost of a product or accessing another user’s sensitive information. If the server processes the manipulated request without validating permissions, the attacker successfully exploits the vulnerability. This highlights the risks of insufficient access controls and the need for proper authorization checks to prevent unauthorized access or modifications.

Attack Methodology

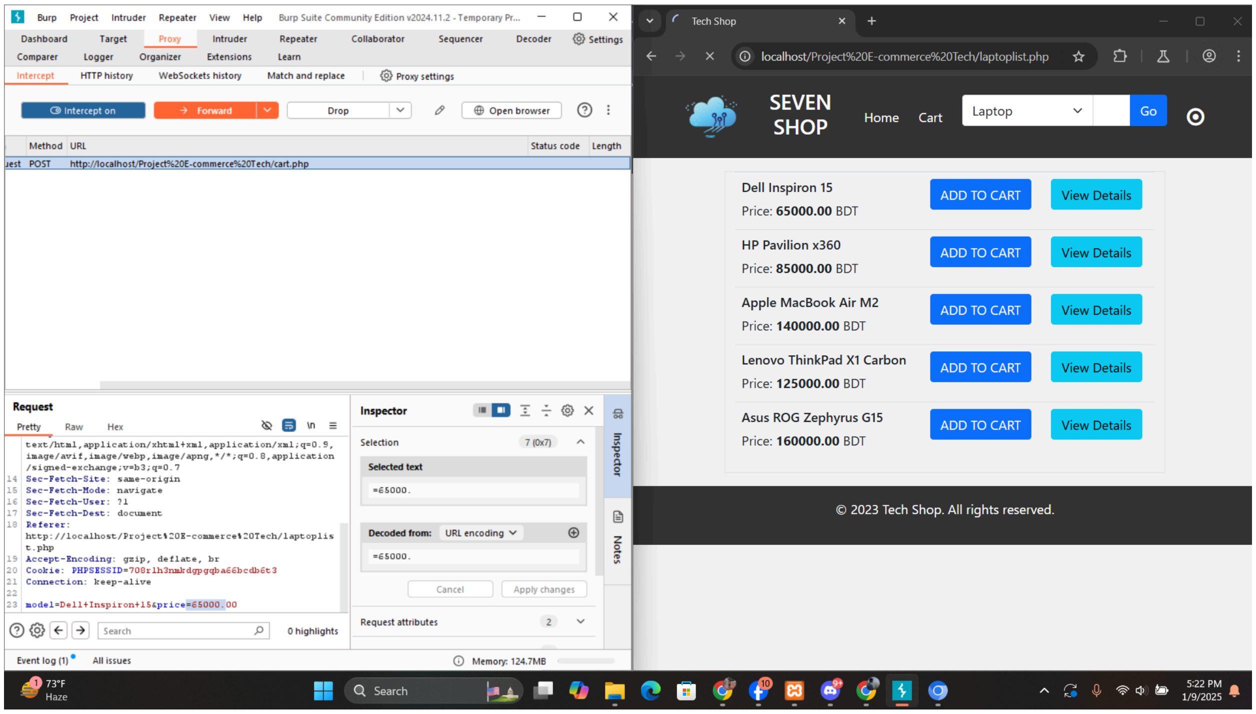
In the case of an IDOR (Insecure Direct Object Reference) Attack, we identified that the application failed to validate user authorization when accessing or modifying resources. Using a tool like Burp Suite, we intercepted a request to update the price of an electronic product on the website. The intercepted request contained parameters such as product_id and price. By modifying the price parameter to a lower value and forwarding the request to the server, we observed that the change was successfully reflected on the main website without any validation. This demonstrated that the application did not enforce proper access controls or verify the legitimacy of the request, making it vulnerable to unauthorized manipulation of sensitive data like product prices.

Attack Walkthrough



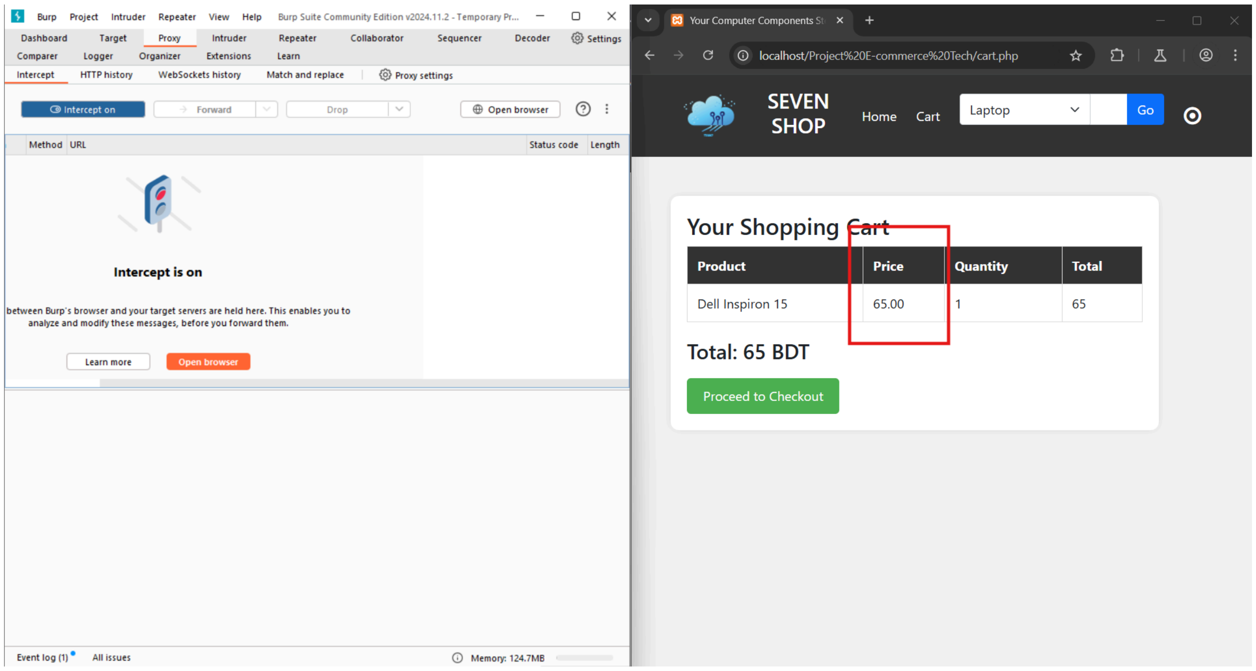
Dell Inspiron 15 Price: **65000.00** BDT	ADD TO CART	View Details
HP Pavilion x360 Price: **85000.00** BDT	ADD TO CART	View Details
Apple MacBook Air M2 Price: **140000.00** BDT	ADD TO CART	View Details
Lenovo ThinkPad X1 Carbon Price: **125000.00** BDT	ADD TO CART	View Details
Asus ROG Zephyrus G15 Price: **160000.00** BDT	ADD TO CART	View Details

© 2023 Tech Shop. All rights reserved.



```
POST   http://localhost/Project%20E-commerce%20Tech/cart.php
```

```
text/html,application/xhtml+xml,application/xml;q=0.9,
image/avif,image/webp,image/apng,*/*;q=0.8,application
/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer:
http://localhost/Project%20E-commerce%20Tech/laptoplis
t.php
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=708rlh3nmkdgpgqba66bcdb6t3
Connection: keep-alive

model=Dell+Inspiron+15&price=65000.00
```


Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Settings
Comparer Logger Organizer Extensions Learn
Intercept HTTP history WebSockets history Match and replace Proxy settings
Intercept on Forward Drop Open browser
Method URL Status code Length
Intercept is on
Learn more Open browser
Event log (1) All issues Memory: 124.7MB
localhost/Project%20E-commerce%20Tech/cart.php
SEVEN SHOP Home Cart Laptop Go
Proceed to Checkout

Intercept is on

between Burp's browser and your target servers are held here. This enables you to analyze and modify these messages, before you forward them.

Your Shopping Cart

Product	Price	Quantity	Total
Dell Inspiron 15	65.00	1	65

Total: 65 BDT

Prevention

Implement Access Controls: Before allowing access to objects, confirm user authorization at every layer.

Use Indirect References: Use secure mappings such as hashed IDs or surrogate identifiers instead of exposing sensitive IDs.

Least Privilege: Limit access to just the data and resources needed for the user's role.

Security Testing: Use vulnerability scanners to detect IDOR during penetration tests.

Audit Logs: Maintain detailed logs for access to sensitive resources and monitor for anomalies.

Failure Attacks

Despite the successful attacks mentioned above, several attack attempts were unsuccessful due to the constraints of the local PHP environment. Limitations such as the inability to send malicious requests, inject content or exploit specific functionalities prevented these attacks. Pointing out the potential weaknesses in real world scenarios while emphasizing the challenges of conducting tests in a controlled environment.

1. Man in the middle Attack

MITM arises when an attacker directly interposes and alters communications between two parties in their blindfold. The attacker is also able to listen, capture and even manipulate data in real time without being noticed.

Working Principle

The attacker authenticates themselves in between the sender and the receiver and can use methods such as ARP poisoning. All communication is handled by the attacker by reading, altering or rerouting the information passed.

Attack Methodology

To perform a Man-in-the-Middle (MITM) attack, we began by installing essential tools like Ettercap and Wireshark on the attacker's machine and setting up the environment using XAMPP for the localhost application. We proceeded with network scanning using commands like ifconfig and arp-scan to identify active devices and retrieve their IP addresses. Next, we configured ARP spoofing to redirect traffic between the target device and the router, aiming to position the attacker in the communication channel. Finally, we attempted to intercept and analyze traffic using Wireshark to verify successful data interception. However, the attack failed as the IP address did not display correctly during network scanning, preventing further progression

2. Key Disclosure Attack

In a key disclosure attack, an attacker who obtains encryption keys without authorization can use them to decrypt sensitive data or pretend to be a genuine user.

Working Principle

A Key Disclosure attack aims to exploit vulnerabilities to extract encryption keys stored insecurely in databases, memory or configuration files. Attackers typically use techniques like SQL injection or memory scraping to access these keys. Once the encryption keys are obtained, the attacker can decrypt sensitive data, compromising the system's security.

Attack Methodology

To perform a key Disclosure attack, we initiated it by analyzing the database for potential vulnerabilities using SQL query injection techniques to identify sensitive encryption keys. The goal was to extract keys stored insecurely in the database or memory. We checked for misconfigurations and accessed tables containing sensitive data. However, upon importing data from the database, it was evident that no encryption was implemented and all the data were stored in plaintext. As a result, the attack could not proceed since the absence of encryption itself represents a serious security breach, leaving no encryption keys to disclose or exploit.

3. Tampered Content Attack

In this attack, submitted content (such as files or media) is altered to contain harmful code or data. An attacker might, for instance, upload files that include encrypted material that has been altered or metadata that has been manipulated.

Working Principle

In a tampered content attack, text, images or scripts are altered or other illegal content is injected into a web application. Attackers usually inject malicious material through file upload features, input fields, or unprotected content management systems. When manipulated content is introduced, the security and integrity of the system may be in jeopardy because it may result in defacement, false information or even the execution of dangerous programs.

Attack Methodology

We started with the review of the content management features of the application for any vulnerabilities that could be used to perform a Tampered Content attack. This could involve adding malicious scripts or modifying product information to modify or inject illegal content. We attempted to introduce tampered data such as additional product information or an unapproved image by exploiting input fields or upload capabilities. However, the program didn't support any uploading kind of content or adding or changing information on the product. The restriction imposed prevented the tampering process and thus unsuccessful assault since no capability supported any kind of content change.

4. CSRF Attack

CSRF is a type of attack that tricks a user into performing an unwanted action (e.g., transferring funds, changing email) on a web application where they're authenticated. It exploits the trust that a website has in the user's browser.

Working Principle

A victim is authenticated in a trusted web application. The attacker embeds a malicious link in an email or on a third-party website, or transmits it to the victim. The malicious request (such as a form submission) is sent to the trusted application using the victim's cookies or session which the browser automatically includes if the victim clicks on the link.

Attack Methodology

We started by trying to take advantage of the application's state-changing requests such as form submissions or URL-based activities that could change user data in order to execute a Cross-Site Request Forgery (CSRF) attack. The goal was to create a malicious image or link that would cause these requests unknowingly when clicked by a vulnerable user. We designed a website controlled by the attacker that would forward these queries to the localhost PHP application. Because the project was stored locally, we could not deliver malicious links or images to a user through the website. Also, since the localhost environment could not allow any cross-origin queries, the attack failed.

5. XSS Attack

One kind of security vulnerability that allows an attacker to insert malicious scripts into websites that other users are seeing is called Cross-Site Scripting (XSS). Because these scripts are usually run within the victim's browser, they may provide hackers the ability to take control of user sessions, steal confidential information, or carry out actions on the user's behalf.

Working Principle

The attacker injects malicious code (typically JavaScript) into a vulnerable website.

The malicious script runs in the victim's browser when they visit the page or use certain features.

The script may act on behalf of the victim or steal session tokens, cookies, or private user information.

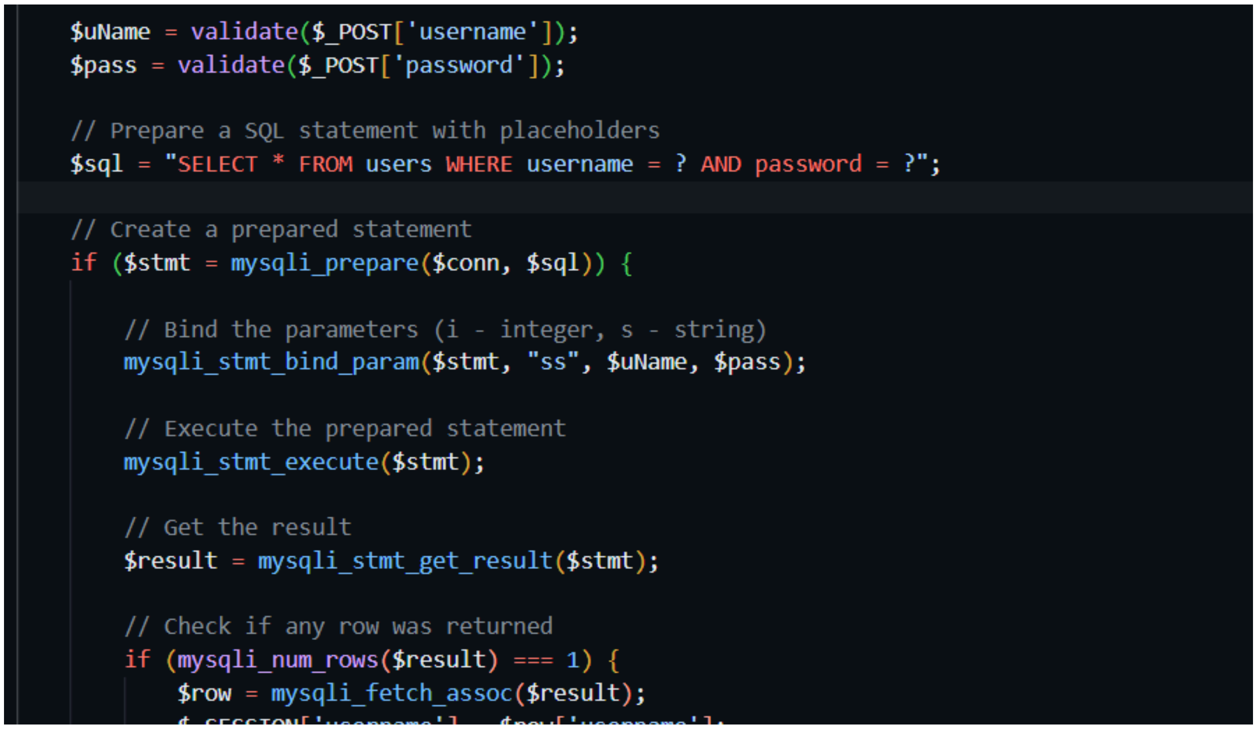
Attack Methodology

In order to perform a Cross-Site Scripting (XSS) attack, we tried to inject malicious scripts in some input fields of an application or in the URLs which would execute while being viewed by a victim client. We craft the payload that triggers a malicious action. Let's say, stealing session cookies or redirecting to another website by taking advantage of unsanitized inputs which could render scripts on the page. But due to the project hosted on a PHP-based localhost, the possibility to send malicious scripts or payloads via links or other content to the user was impossible. Therefore, due to this environment and cross origin not being able to support this, it will cause an XSS attack to fail.

Security Implementation (Additional Work):

How we prevented SQLi:

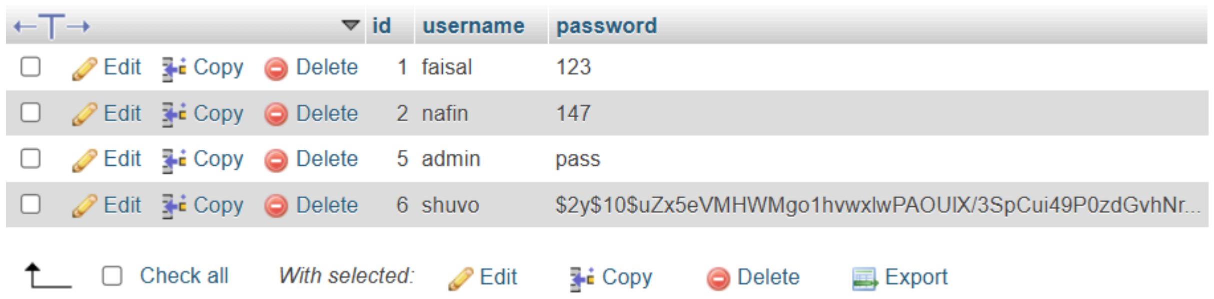
To prevent SQL injection, we need to **properly sanitize and parameterize our SQL queries**



```php
    $uName = validate($_POST['username']);
    $pass = validate($_POST['password']);

    // Prepare a SQL statement with placeholders
    $sql = "SELECT * FROM users WHERE username = ? AND password = ?";

    // Create a prepared statement
    if ($stmt = mysqli_prepare($conn, $sql)) {

        // Bind the parameters (i - integer, s - string)
        mysqli_stmt_bind_param($stmt, "ss", $uName, $pass);

        // Execute the prepared statement
        mysqli_stmt_execute($stmt);

        // Get the result
        $result = mysqli_stmt_get_result($stmt);

        // Check if any row was returned
        if (mysqli_num_rows($result) === 1) {
            $row = mysqli_fetch_assoc($result);
            $_SESSION['username'] = $row['username'];
```

The query is now prepared using placeholders (?) for the values ($uName and $pass), ensuring the values are treated as data, not SQL code. The mysqli_stmt_bind_param() function binds the user inputs to the placeholders. In this case, both are strings. Here s stands for string. The prepared statement is executed using mysqli_stmt_execute(), and the result is fetched using mysqli_stmt_get_result().

How we created password hashing:



id	username	password
1	faisal	123
2	nafin	147
5	admin	pass
6	shuvo	$2y$10$uZx5eVMHWMgo1hvwxlwPAOUIX/3SpCui49P0zdGvhNr...

Instead of storing the password in plain text, we hash it using **password_hash()** with the PASSWORD_BCRYPT algorithm. This ensures the password is stored securely. **password_hash()** function automatically handles salting and hashing of the password, making it much more secure. **PASSWORD_BCRYPT** is a strong algorithm that is widely accepted and suitable for password hashing.

Conclusion

This penetration testing work successfully identified several critical vulnerabilities within the localhost PHP application, shedding light on potential risks in real-world attacks and their potential impact on the application's security and integrity. While strict input validation, secure authentication and robust access control policies are fundamental to securing applications. The testing process also revealed the need for better protection against data breaches such as encryption for sensitive data. Although some attack attempts were unsuccessful due to the limitations of the local PHP environment such as the inability to send malicious requests or inject content, the failures themselves underscored important constraints of the setup. These outcomes emphasize the importance of incorporating preventive measures for vulnerabilities even in controlled testing environments. The proposed remediation steps are essential for handling these weaknesses effectively ensuring that the application becomes much more secure and resilient. Proactively identifying and mitigating vulnerabilities not only strengthens system security but also builds trust among users, helping to safeguard against an evolving landscape of cyber threats.