# ASSIGNMENT-1

**Name :** Bhumireddy Thanmaye

**Reg No:** 21BCI0013

**Course:** AI for Cyber Security With IBM Qradar (AI for Web Security)

**Branch:** Computer Science and Engineering with specialization in Information Security

**Assignment Objective:**

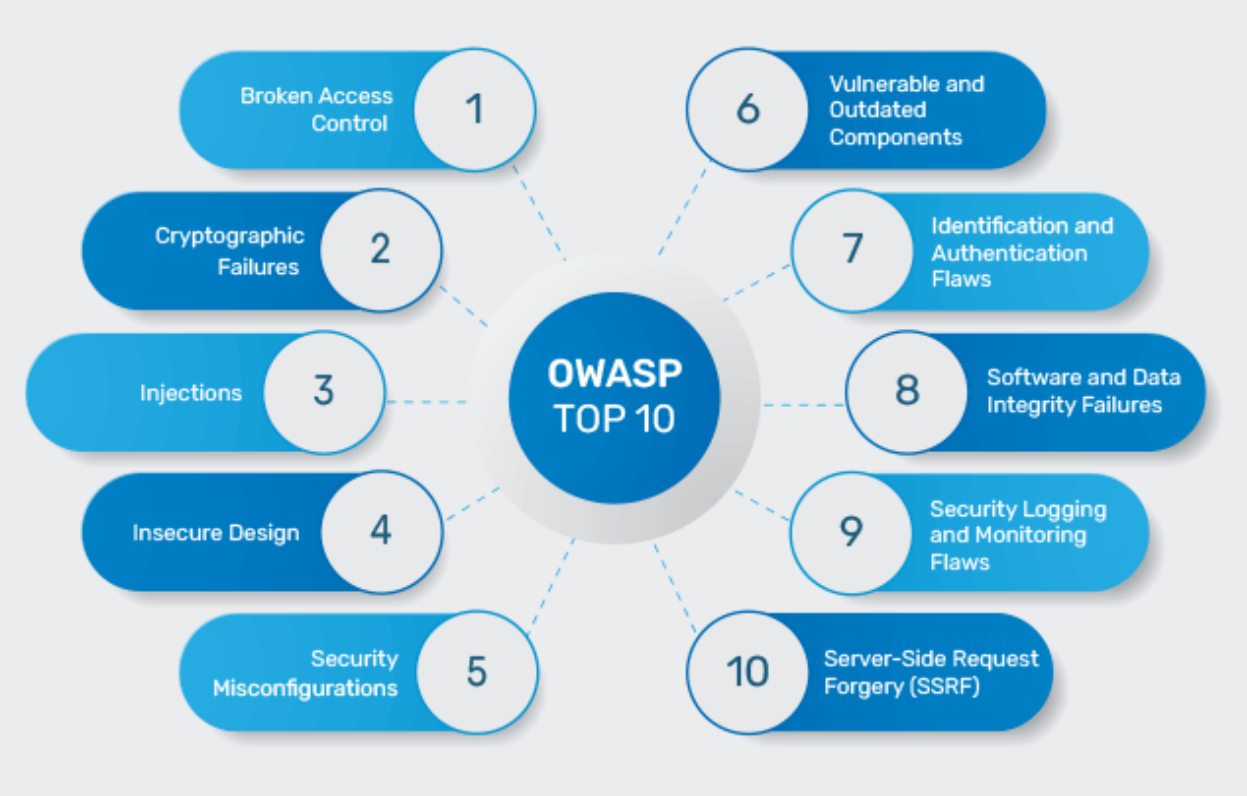
OWASP TOP 10 VULNERABILITIES(TAKE ANY 5) PERFORM IN ANY WEBSITE AND SHOW PROOF OF CONTACT

**What is OWASP**

- OWASP - Open Worldwide Application Security Project
- An online community that produces freely-available articles, methodologies, documentation, tools, and technologies in the field of web application security. The OWASP provides free and open resources. It is led by a non-profit called The OWASP Foundation.

**OWASP Top Ten**

The OWASP Top 10 is a standard awareness document for developers and web application security. It represents a broad consensus about the most critical security risks to web applications.




## 1. Broken Access Control:

Refers to the failure of a system to properly enforce restrictions on what authenticated users are allowed to do. This vulnerability can result from flaws in authentication, session management, and authorization mechanisms, potentially granting unauthorized users access to sensitive data and functionality.

**Exploitation:**

Exploiting broken access control can involve unauthorized users gaining elevated privileges, accessing restricted resources, or modifying data beyond their authorized scope. Attackers might manipulate URL parameters, bypass authentication mechanisms, or abuse insufficiently protected API endpoints to achieve their objectives.

**Mitigation:**

To mitigate this vulnerability, developers should implement proper authentication and authorization mechanisms, such as role-based access control (RBAC) and attribute-based access control (ABAC). Regular security audits, role validation, and thorough testing of access controls can help identify and rectify potential weaknesses**.**

**Business Impact:**

The business impact of broken access control can be severe, including unauthorized access to sensitive customer data, intellectual property theft, regulatory fines, and reputational damage. Breaches resulting from this vulnerability can lead to legal liabilities and loss of customer trust, resulting in significant financial losses.

**List of Mapped CWEs:**

CWE-22 Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')

CWE-23 Relative Path Traversal

CWE-35 Path Traversal: '.../...//'

CWE-59 Improper Link Resolution Before File Access ('Link Following')

CWE-200 Exposure of Sensitive Information to an Unauthorized Actor

CWE-201 Exposure of Sensitive Information Through Sent Data

CWE-219 Storage of File with Sensitive Data Under Web Root

CWE-264 Permissions, Privileges, and Access Controls (should no longer be used)

CWE-275 Permission Issues

CWE-276 Incorrect Default Permissions

CWE-284 Improper Access Control

CWE-284: Improper Access Control

**Description:**The product does not restrict or incorrectly restricts access to a resource from an unauthorized actor.

**Business Impact:**Access control involves the use of several protection mechanisms such as:Authentication (proving the identity of an actor),Authorization (ensuring that a given actor can access a resource), and Accountability (tracking of activities that were performed).If suppose a unauthorized actor came to see the resources if any of the above mechanism is not applied or otherwise fails ,then they are giving chance to the attackers.This can leads to compromise the security of the product by gaining privileges, reading sensitive information, executing commands, evading detection.

CWE-285 Improper Authorization

CWE-352 Cross-Site Request Forgery (CSRF)

CWE-359 Exposure of Private Personal Information to an Unauthorized Actor

CWE-377 Insecure Temporary File

CWE-402 Transmission of Private Resources into a New Sphere ('Resource Leak')

CWE-425 Direct Request ('Forced Browsing')

CWE-441 Unintended Proxy or Intermediary ('Confused Deputy')

CWE-497 Exposure of Sensitive System Information to an Unauthorized Control Sphere

CWE-538 Insertion of Sensitive Information into Externally-Accessible File or Directory

CWE-540 Inclusion of Sensitive Information in Source Code

CWE-548 Exposure of Information Through Directory Listing

CWE-552 Files or Directories Accessible to External Parties

CWE-566 Authorization Bypass Through User-Controlled SQL Primary Key

CWE-601 URL Redirection to Untrusted Site ('Open Redirect')

CWE-639 Authorization Bypass Through User-Controlled Key

CWE-651 Exposure of WSDL File Containing Sensitive Information

CWE-668 Exposure of Resource to Wrong Sphere

CWE-706 Use of Incorrectly-Resolved Name or Reference

CWE-862 Missing Authorization

CWE-863 Incorrect Authorization

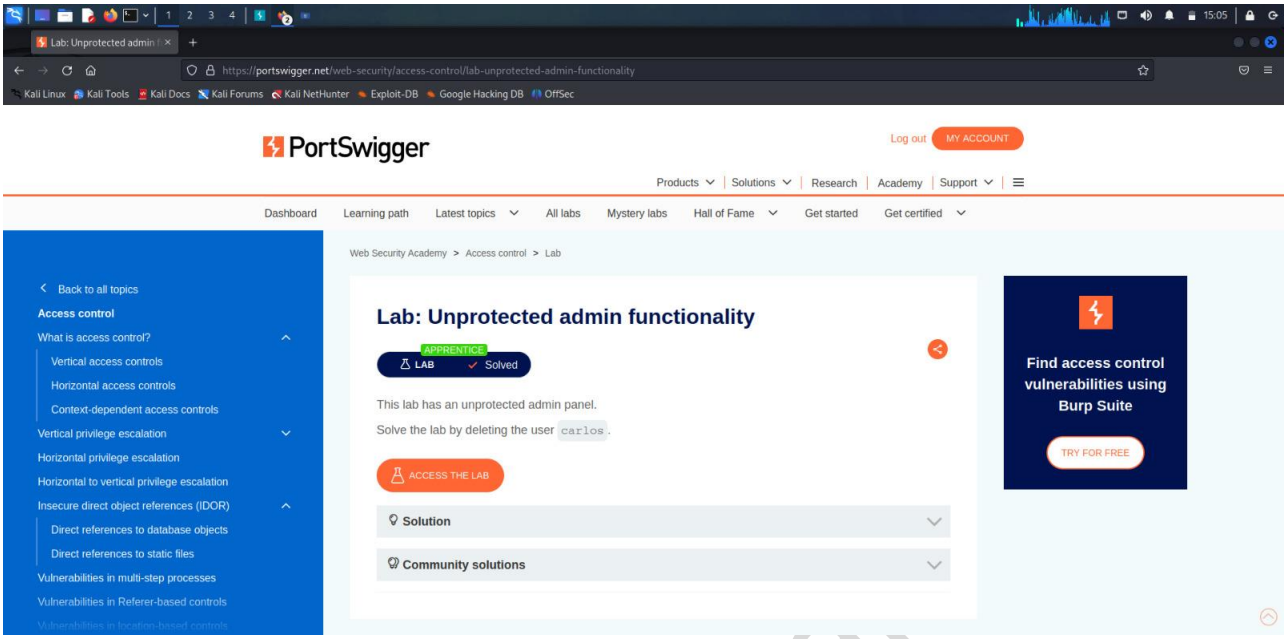
CWE-913 Improper Control of Dynamically-Managed Code Resources

CWE-922 Insecure Storage of Sensitive Information

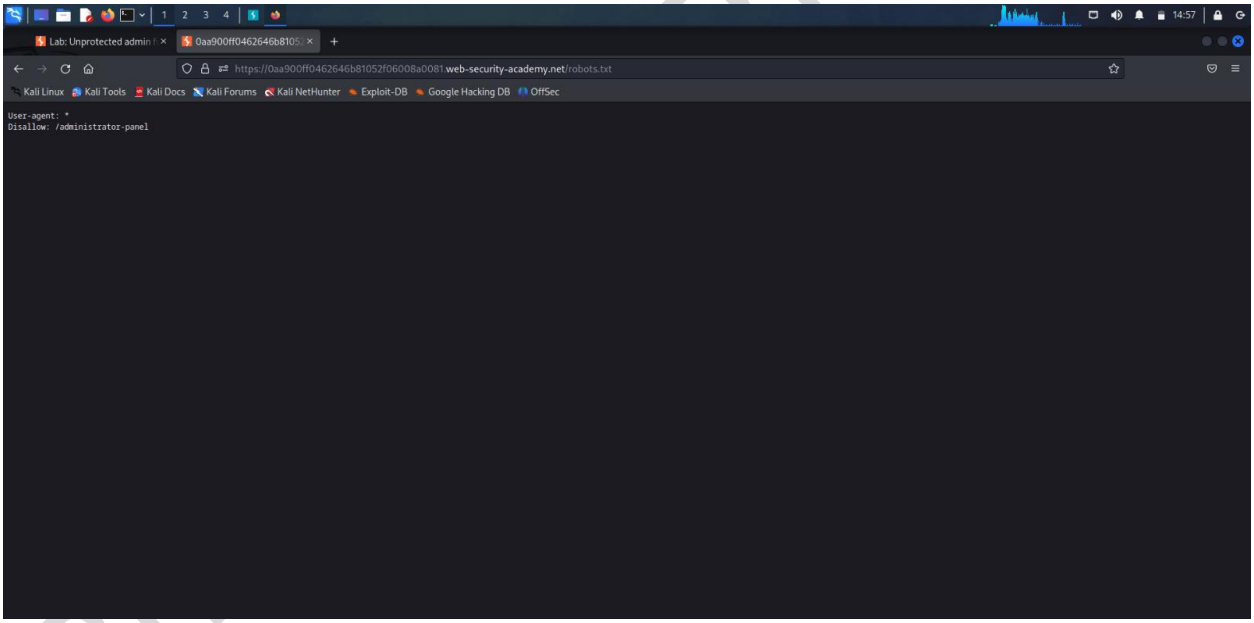
CWE-1275 Sensitive Cookie with Improper SameSite Attribute

**Practical Approach:**

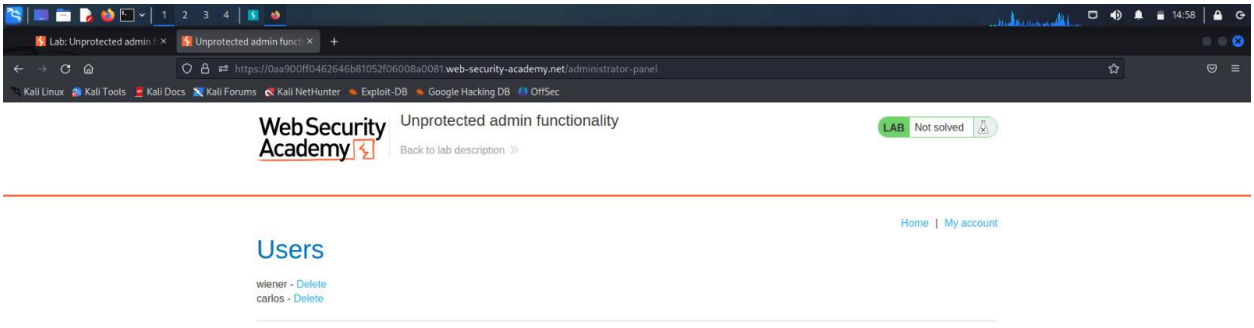
After reading the description,click on access the lab



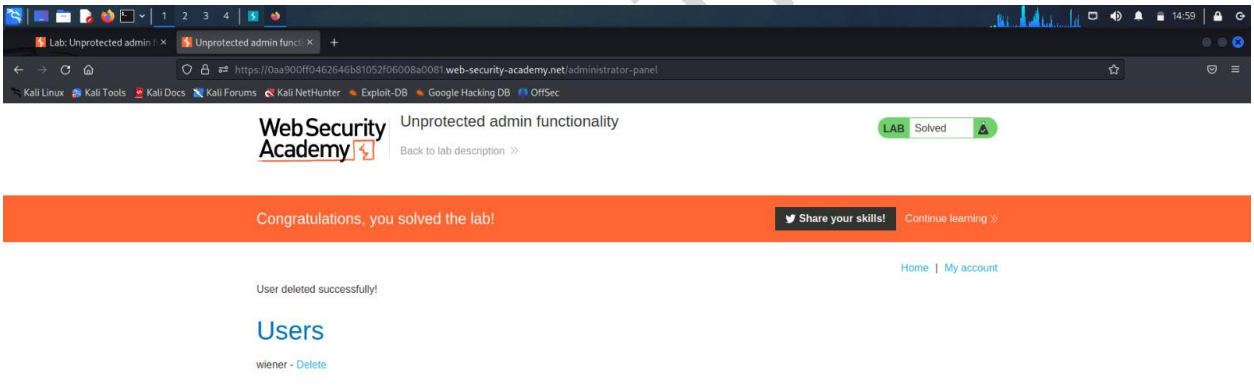

Use /robots.txt in the lab access url




In the url replace /robots.txt to /administrator-panel




As per the description of scenario delete carlos user from the administrator-panel




Therefore,we deleted the carlos user account from administrator panel due to unprotected admin functionality

## 2. Cryptographic Failures:

Cryptographic Failures involve the improper implementation or use of cryptographic mechanisms, rendering sensitive data susceptible to unauthorized access. These failures can encompass weak key management, flawed encryption algorithms, or insecure random number generation.

**Exploitation:**

Attackers can exploit cryptographic failures to decrypt intercepted data, perform man-in-the-middle attacks, or recover sensitive information from improperly protected storage. Weakened encryption may also allow unauthorized tampering with data during transit.

**Mitigation:**

Mitigating cryptographic failures requires using industry-standard encryption algorithms, strong key management practices, and secure random number generators. Regular audits of cryptographic implementations and adherence to established cryptographic guidelines are essential to ensure robust protection.

**Business Impact:**

Cryptographic failures can lead to severe breaches of confidential data, loss of competitive advantage, and compromised business relationships. If customer data or trade secrets are compromised, legal repercussions, compliance violations, and tarnished brand reputation can result.

**List of Mapped CWEs:**

CWE-261 Weak Encoding for Password

**Description:**Obscuring a password with a trivial encoding does not protect the password.

**Business Impact:** Basically passwords are some of the most commonly used authentication mechanisms. To authenticate to a system/application, a user provides a password, which is compared to a value stored on the server. If the two values match, then the user is granted access. Otherwise, access is denied.Password encoding can go wrong in some ways like when the passwords are stored in plain text,using simple encoding algorithms like Base64 which is reversible that means we can decode the original value and use of weak hash algorithms that can be exploited by brute force attacks .This can lead to sensitive data exposure or system compromise.

CWE-296 Improper Following of a Certificate's Chain of Trust

CWE-310 Cryptographic Issues

CWE-319 Cleartext Transmission of Sensitive Information

CWE-321 Use of Hard-coded Cryptographic Key

CWE-322 Key Exchange without Entity Authentication

CWE-323 Reusing a Nonce, Key Pair in Encryption

CWE-324 Use of a Key Past its Expiration Date

CWE-325 Missing Required Cryptographic Step

CWE-326 Inadequate Encryption Strength

CWE-327 Use of a Broken or Risky Cryptographic Algorithm

CWE-328 Reversible One-Way Hash

CWE-329 Not Using a Random IV with CBC Mode

CWE-330 Use of Insufficiently Random Values

CWE-331 Insufficient Entropy

CWE-335 Incorrect Usage of Seeds in Pseudo-Random Number Generator(PRNG)

CWE-336 Same Seed in Pseudo-Random Number Generator (PRNG)

CWE-337 Predictable Seed in Pseudo-Random Number Generator (PRNG)

CWE-338 Use of Cryptographically Weak Pseudo-Random Number Generator(PRNG)

CWE-340 Generation of Predictable Numbers or Identifiers

CWE-347 Improper Verification of Cryptographic Signature

CWE-523 Unprotected Transport of Credentials

CWE-720 OWASP Top Ten 2007 Category A9 - Insecure Communications

CWE-757 Selection of Less-Secure Algorithm During Negotiation('Algorithm Downgrade')

CWE-759 Use of a One-Way Hash without a Salt

CWE-760 Use of a One-Way Hash with a Predictable Salt

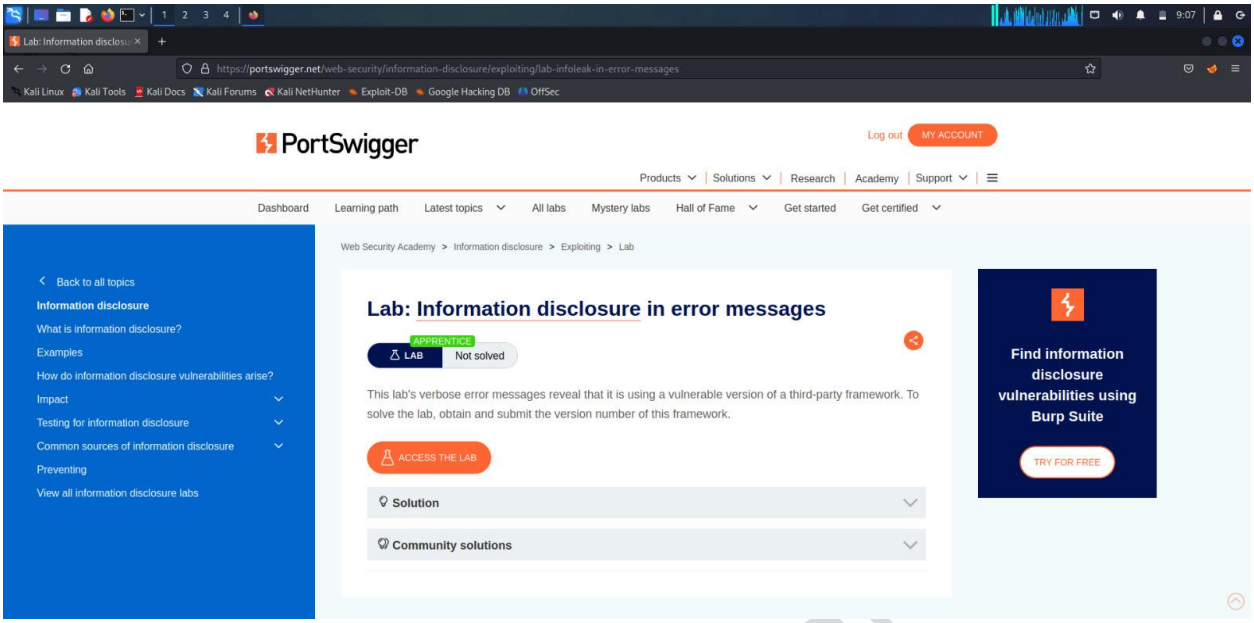
CWE-780 Use of RSA Algorithm without OAEP

CWE-818 Insufficient Transport Layer Protection

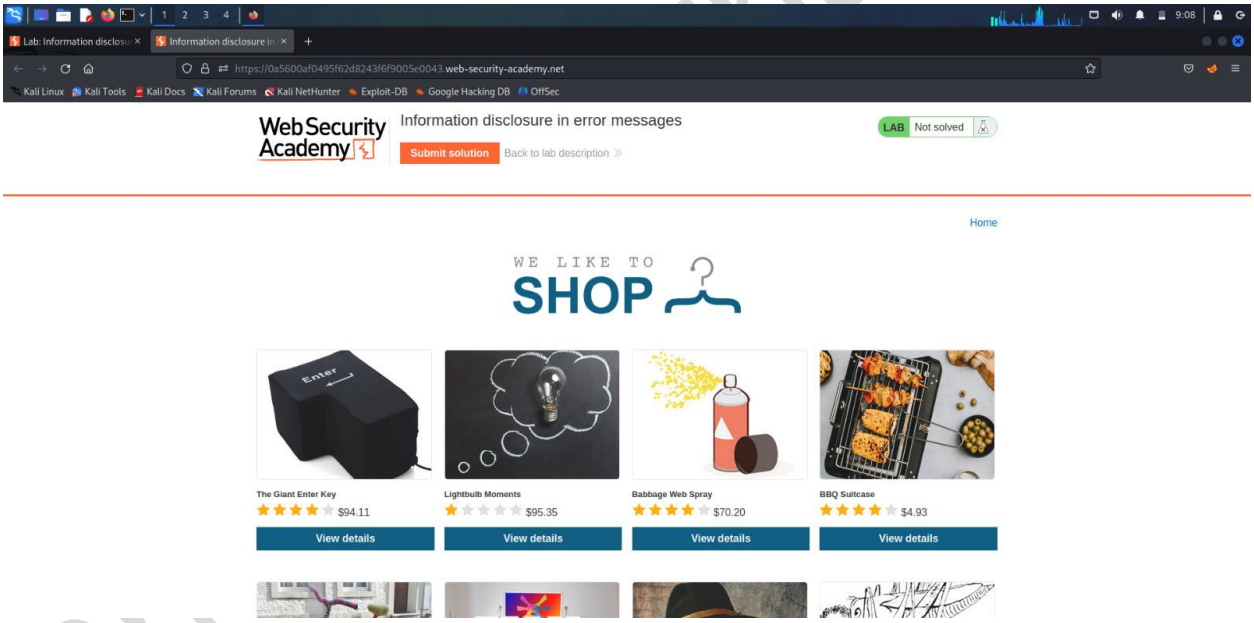
CWE-916 Use of Password Hash With Insufficient Computational Effort

**Practical Approach:**

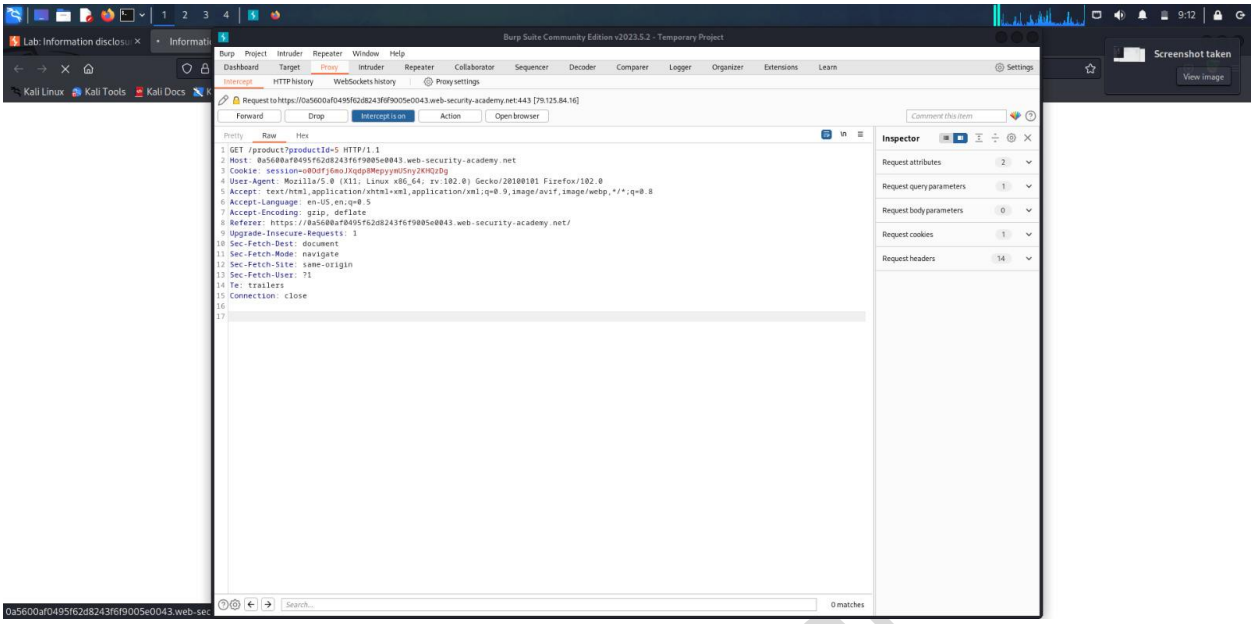
Go to port swigger website and take the information disclosure in error message lab

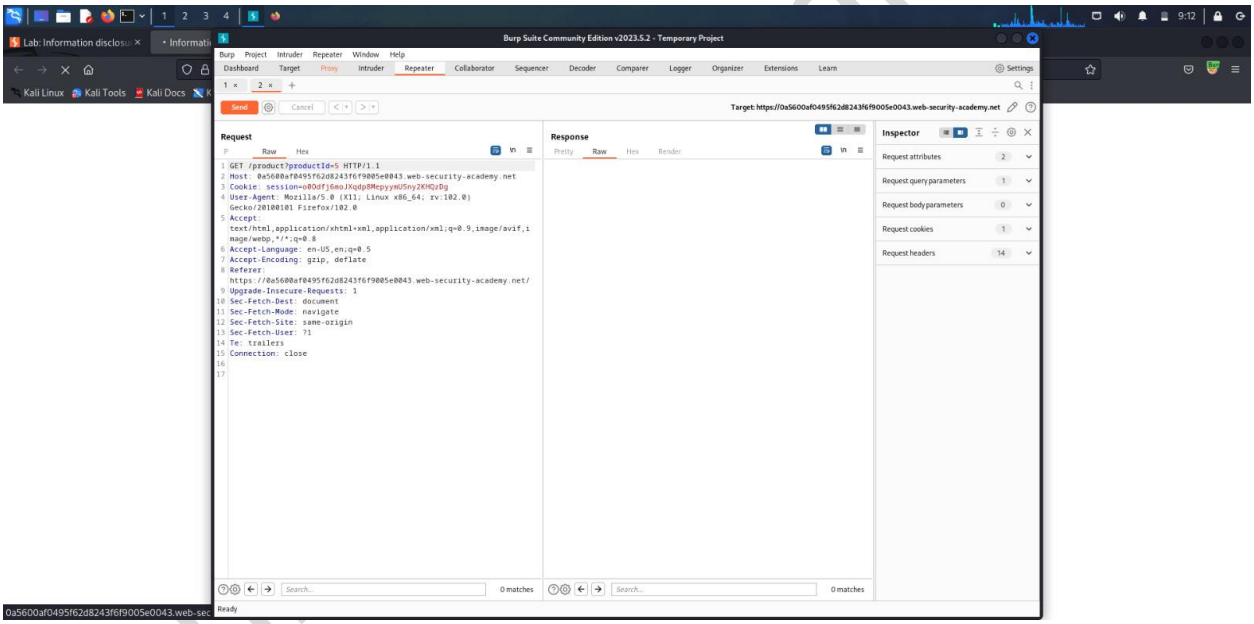



Now click on access the lab,select any one of them




Change foxy proxy to firefox settings to burp,open burp suite go to proxy on the intercept and right click send to repeater

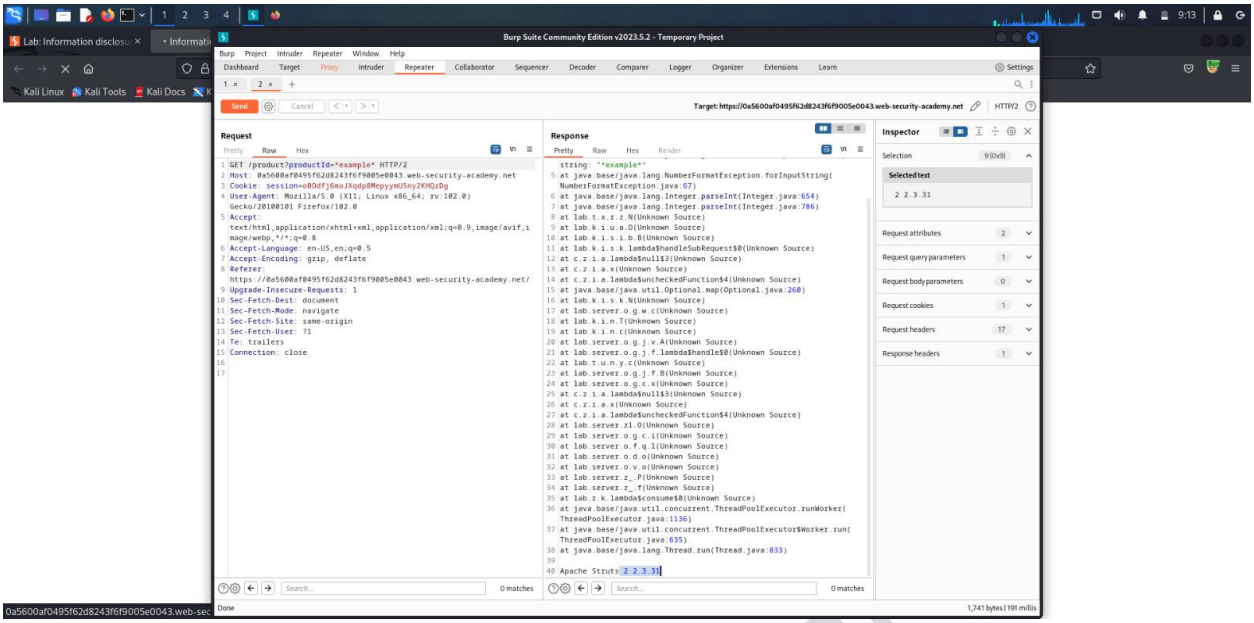



Change the product id from 5 to like *example* for suppose and click on request send

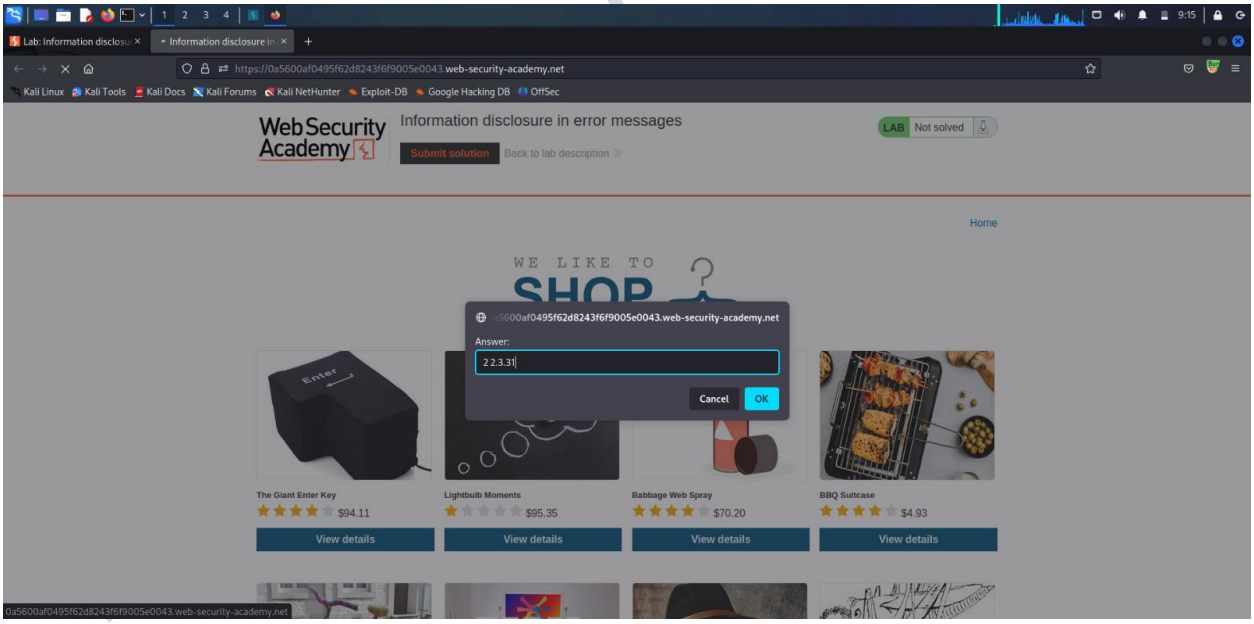
Now we can some apache status in Response to the given request

Copy the apache status

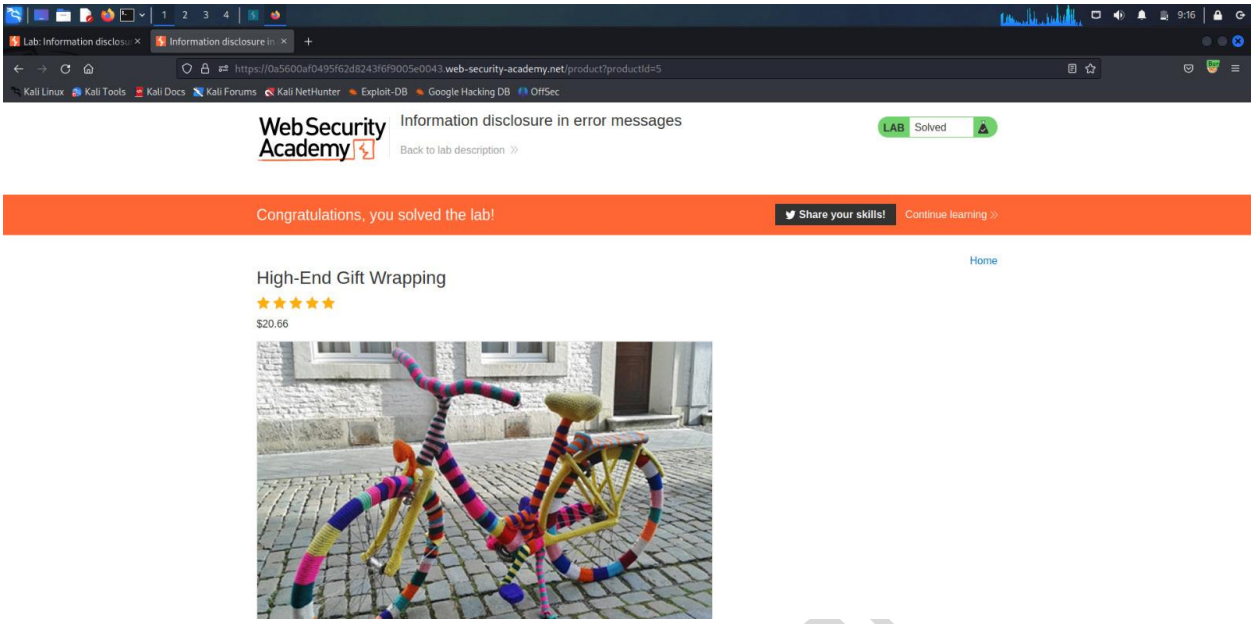



Before going to portswigger website off the intercept

Go to portswigger click on submit solution paste the apache status there and submit




This lab using apache framework version 2 2.3.31 by submitting the version in the solution the lab can be solved






## 3. Injections:

Injection vulnerabilities arise when untrusted data is improperly handled and executed within an application's runtime. Common types include SQL Injection, where malicious SQL queries are injected, and Command Injection, where malicious commands are executed on a system.

**Exploitation:**

Attackers exploit injection vulnerabilities by inserting malicious code into input fields or data streams. For example, SQL Injection attacks can manipulate database queries to reveal sensitive information, modify data, or even gain administrative access to systems.

**Mitigation:**

Mitigating injection vulnerabilities involves using parameterized queries, input validation, and output encoding. Properly sanitizing and validating

user inputs before processing them can significantly reduce the risk of injection attacks.

**Business Impact:**

Injection attacks can lead to data breaches, loss of data integrity, service downtime, and potential legal liabilities. Compromised customer data can erode trust and trigger regulatory penalties, damaging a company's financial standing and reputation.

**List of Mapped CWEs:**

CWE-20 Improper Input Validation

**Description:**The product receives input or data, but it does not validate or incorrectly validates that the input has the properties that are required to process the data safely and correctly.

**Business Impact:** The failure to properly validate or sanitize input data from untrusted sources before using it in a software system leads to parts of the system receiving unintended input, which may result in altered control flow, arbitrary control of a resource, or arbitrary code execution. This can lead to a range of security vulnerabilities which include security breaches, unauthorized access, disruptions to application functionality, privacy violations by exposing user information, legal and regulatory penalties due to mishandling of data, reputational damage, and resource consumption through attacks like Denial of Service (DoS) .

CWE-74 Improper Neutralization of Special Elements in Output Used by a Downstream Component ('Injection')

CWE-75 Failure to Sanitize Special Elements into a Different Plane (Special Element Injection)

CWE-77 Improper Neutralization of Special Elements used in a Command ('Command Injection')

CWE-78 Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')

CWE-79 Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')

CWE-80 Improper Neutralization of Script-Related HTML Tags in a Web Page (Basic XSS)

CWE-83 Improper Neutralization of Script in Attributes in a Web Page

CWE-87 Improper Neutralization of Alternate XSS Syntax

CWE-88 Improper Neutralization of Argument Delimiters in a Command ('Argument Injection')

CWE-89 Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')

CWE-90 Improper Neutralization of Special Elements used in an LDAP Query ('LDAP Injection')

CWE-91 XML Injection (aka Blind XPath Injection)

CWE-93 Improper Neutralization of CRLF Sequences ('CRLF Injection')

CWE-94 Improper Control of Generation of Code ('Code Injection')

CWE-95 Improper Neutralization of Directives in Dynamically Evaluated Code ('Eval Injection')

CWE-96 Improper Neutralization of Directives in Statically Saved Code ('Static Code Injection')

CWE-97 Improper Neutralization of Server-Side Includes (SSI) Within a Web Page

CWE-98 Improper Control of Filename for Include/Require Statement in PHP Program ('PHP Remote File Inclusion')

CWE-99 Improper Control of Resource Identifiers ('Resource Injection')

CWE-100 Deprecated: Was catch-all for input validation issues

CWE-113 Improper Neutralization of CRLF Sequences in HTTP Headers ('HTTP Response Splitting')

CWE-116 Improper Encoding or Escaping of Output

CWE-138 Improper Neutralization of Special Elements

CWE-184 Incomplete List of Disallowed Inputs

CWE-470 Use of Externally-Controlled Input to Select Classes or Code ('Unsafe Reflection')

CWE-471 Modification of Assumed-Immutable Data (MAID)

CWE-564 SQL Injection: Hibernate

CWE-610 Externally Controlled Reference to a Resource in Another Sphere

CWE-643 Improper Neutralization of Data within XPath Expressions ('XPath Injection')

CWE-644 Improper Neutralization of HTTP Headers for Scripting Syntax

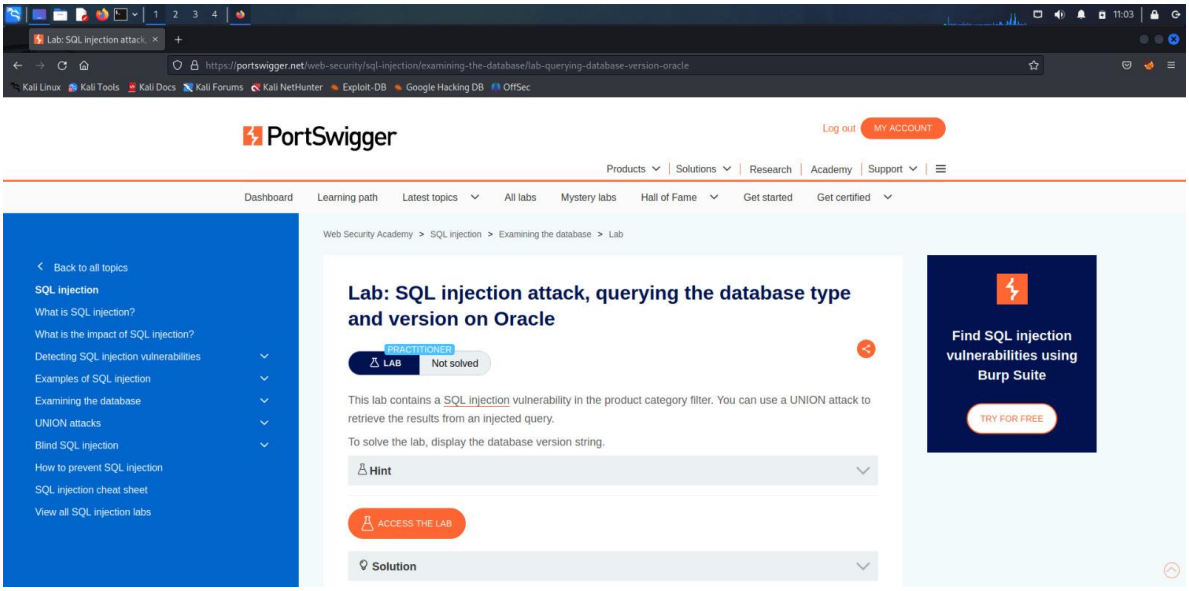
CWE-652 Improper Neutralization of Data within XQuery Expressions ('XQuery Injection')

CWE-917 Improper Neutralization of Special Elements used in an Expression Language Statement ('Expression Language Injection')

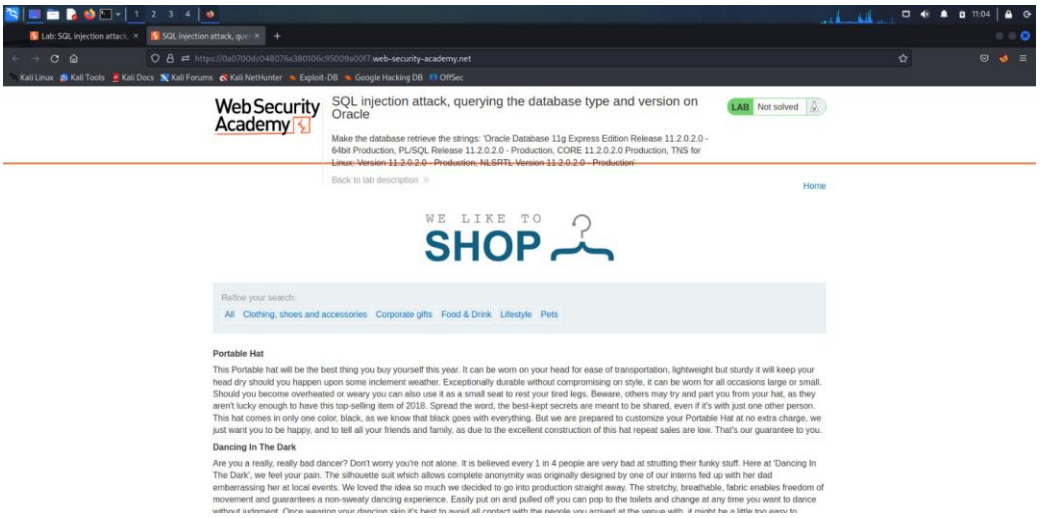
## Practical Approach:

It's also almost similar approach to cryptographic failure how we did

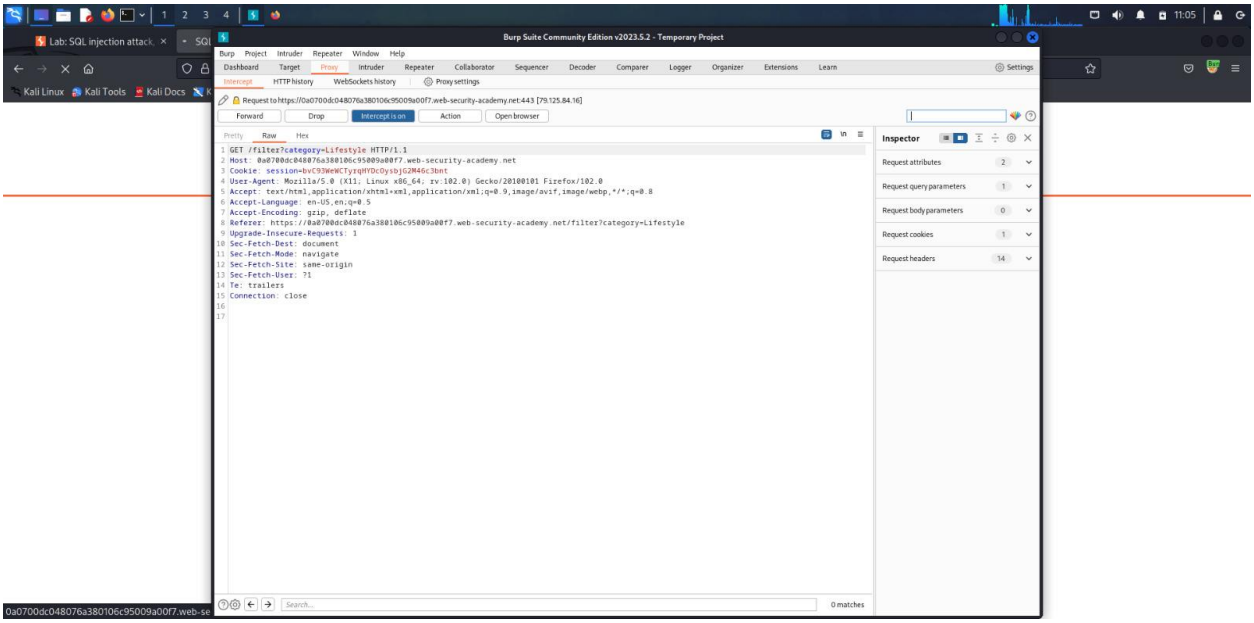
Go to portswigger choose one of the lab under sql injection



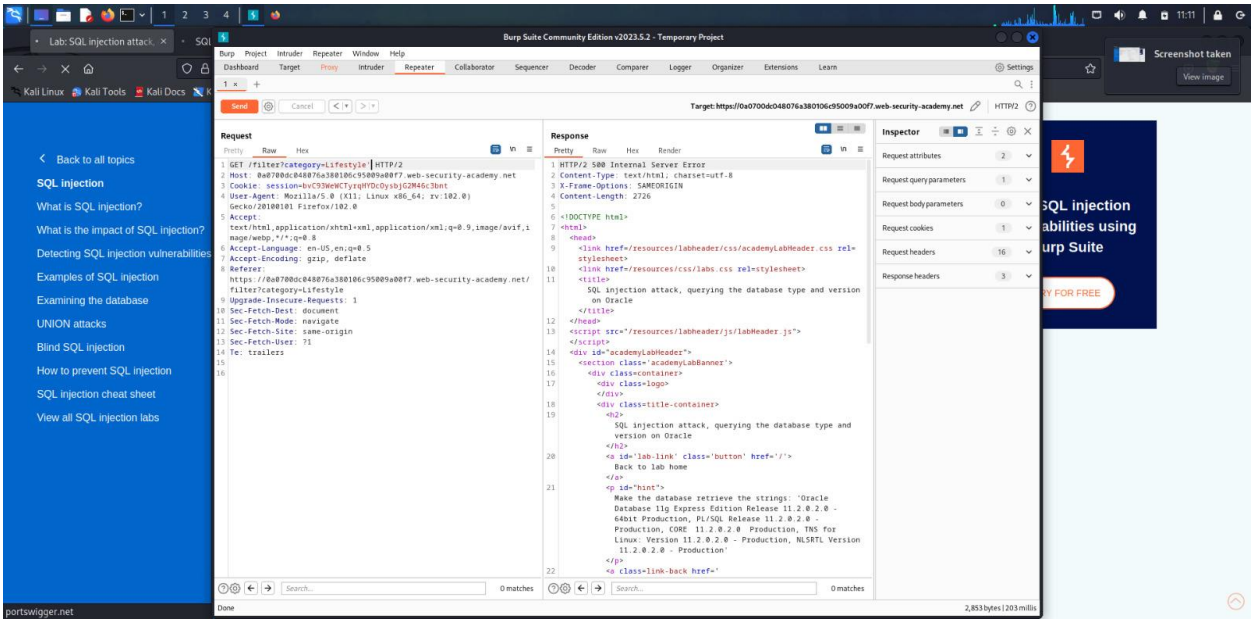

Access the lab go to any of the search in details i have chosen lifestyle here

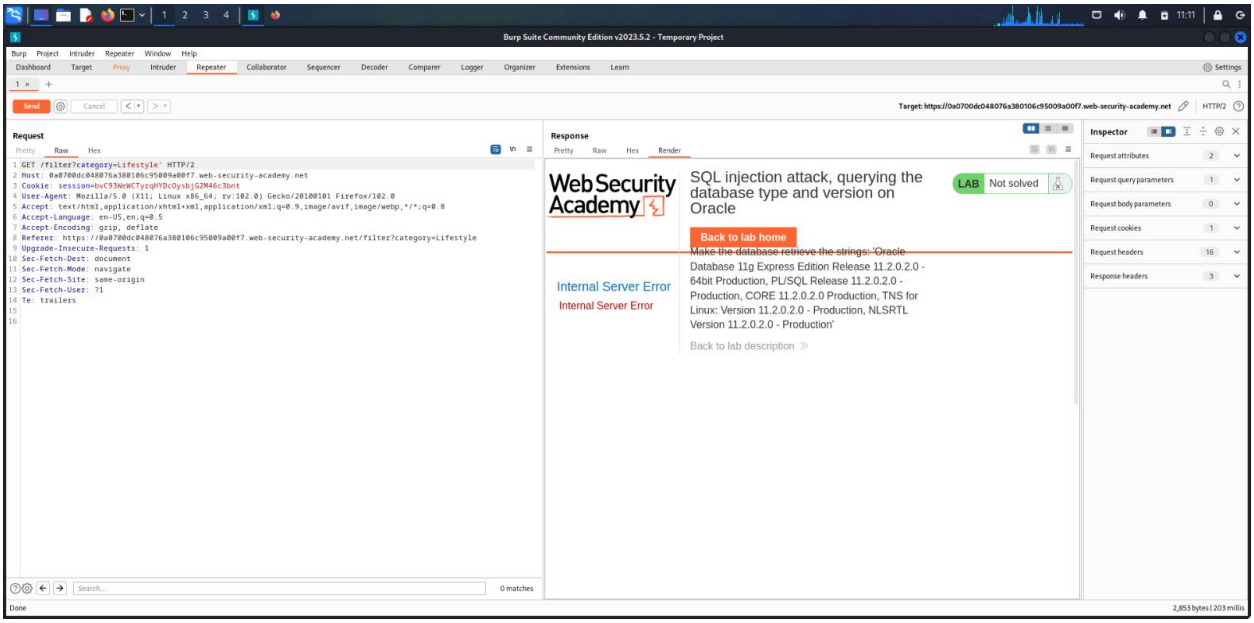



Using foxyproxy extension give burp settings and on the intercept on burp suite



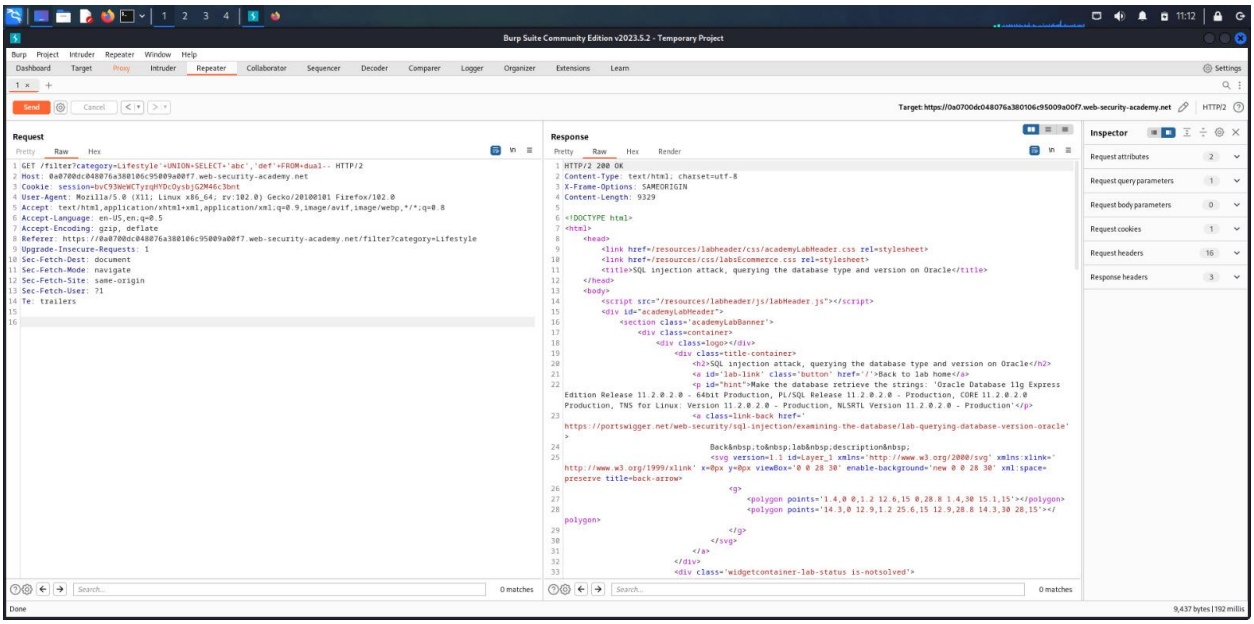
Right click on it and select send to repeater.Now go to repeater and try to append with some like ‘ single quotation at the end of the string and send a request.





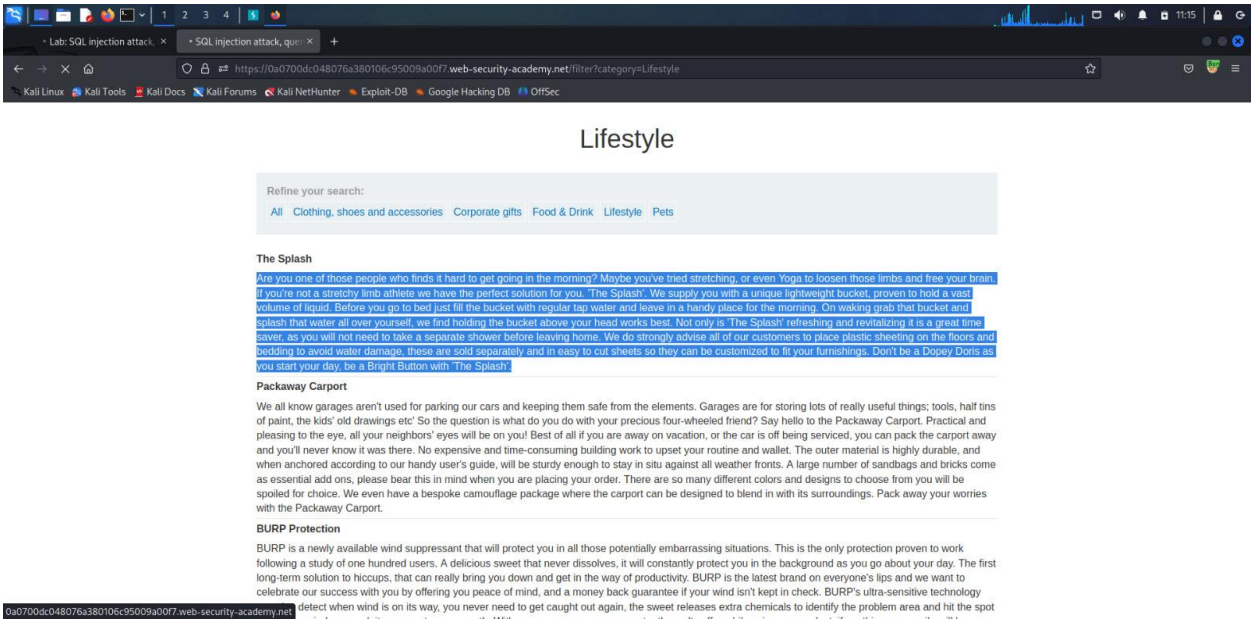

Here by doing that we got an internal server error that means we are using some wrong syntax or something

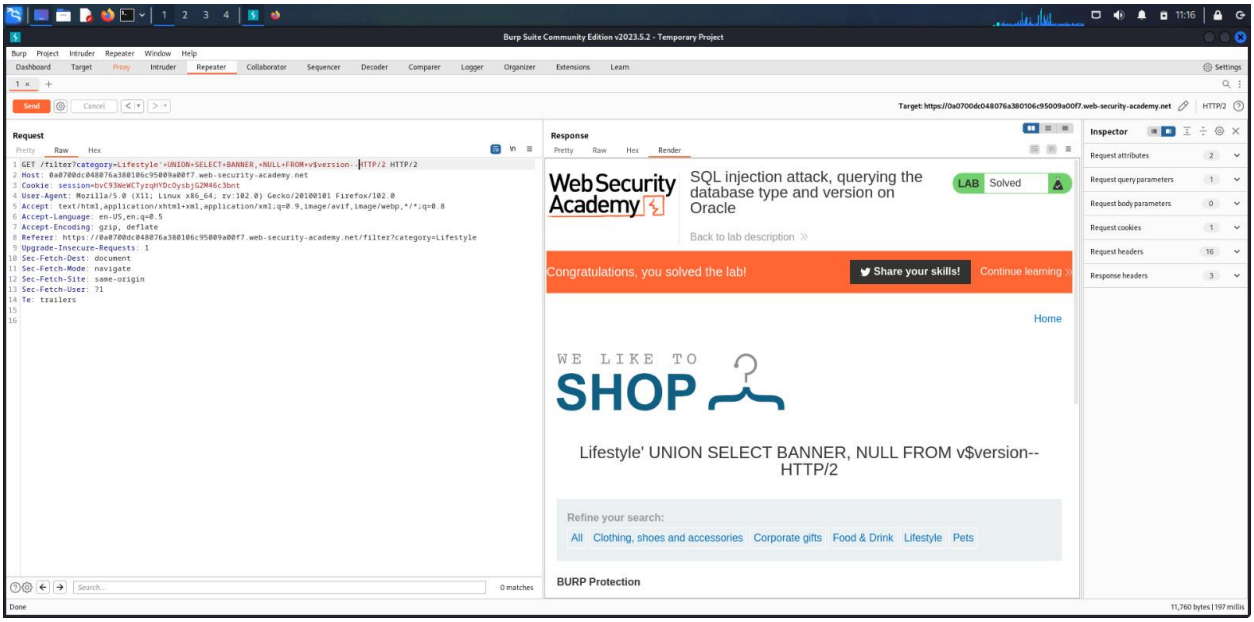
To determine the number of columns that are being returned by the query and which columns contain text data using a payload in the category parameter '+UNION+SELECT+'abc','def'+FROM+dual–



By sending the request it returning the 200 ok that means this is a successful sql injection attack

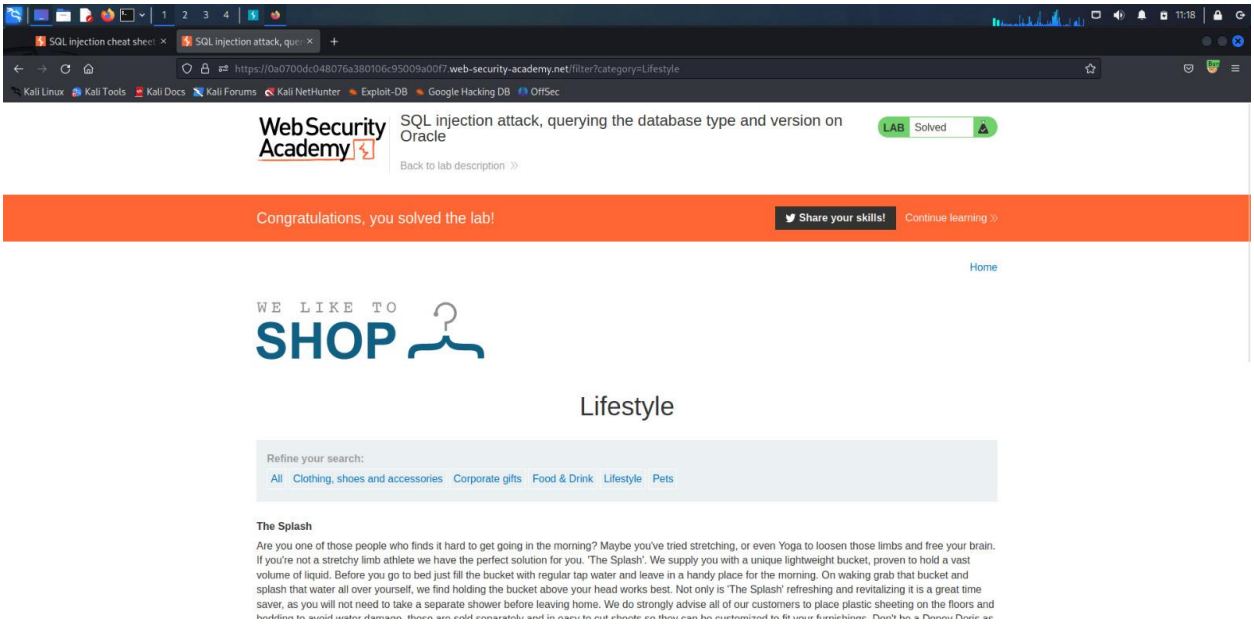
Select some text on lifestyle ,use another payload to retrieve the database version '+UNION+SELECT+BANNER,+NULL+FROM+v$version– and send request




It gave their version.So,the lab is solved

Now go to proxy and paste the payload then forward and off the intercept.Go to port swigger it completely solved






## 4. Insecure Design:

Insecure Design vulnerabilities stem from poor architectural choices and a lack of security considerations during the design phase. These vulnerabilities can result in systemic weaknesses that are difficult to rectify later in the development cycle.

**Exploitation:**

Attackers can exploit insecure design by taking advantage of structural weaknesses in the application or system architecture. These vulnerabilities might allow unauthorized access, data leakage, or unintended functionality.

**Mitigation:**

Preventing insecure design vulnerabilities requires a comprehensive security assessment during the design phase. Adhering to secure design principles, following established coding practices, and conducting thorough security reviews can help identify and rectify design flaws.

**Business Impact:**

Insecure design vulnerabilities can lead to long-term security maintenance challenges, frequent breaches, and costly re-architecting efforts. Addressing these vulnerabilities post-development can be financially burdensome and damage a company's ability to deliver secure products.

**List of Mapped CWEs:**

CWE-73 External Control of File Name or Path

CWE-183 Permissive List of Allowed Inputs

CWE-209 Generation of Error Message Containing Sensitive Information

CWE-213 Exposure of Sensitive Information Due to Incompatible Policies

CWE-235 Improper Handling of Extra Parameters

CWE-256 Unprotected Storage of Credentials

CWE-257 Storing Passwords in a Recoverable Format

CWE-266 Incorrect Privilege Assignment

CWE-269 Improper Privilege Management

CWE-280 Improper Handling of Insufficient Permissions or Privileges

CWE-311 Missing Encryption of Sensitive Data

CWE-312 Cleartext Storage of Sensitive Information

CWE-313 Cleartext Storage in a File or on Disk

CWE-316 Cleartext Storage of Sensitive Information in Memory

CWE-419 Unprotected Primary Channel

CWE-430 Deployment of Wrong Handler

CWE-434 Unrestricted Upload of File with Dangerous Type

CWE-444 Inconsistent Interpretation of HTTP Requests ('HTTP Request Smuggling')

CWE-451 User Interface (UI) Misrepresentation of Critical Information

CWE-472 External Control of Assumed-Immutable Web Parameter

CWE-501 Trust Boundary Violation

CWE-522 Insufficiently Protected Credentials

CWE-525 Use of Web Browser Cache Containing Sensitive Information

CWE-539 Use of Persistent Cookies Containing Sensitive Information

CWE-579 J2EE Bad Practices: Non-serializable Object Stored in Session

CWE-598 Use of GET Request Method With Sensitive Query Strings

CWE-602 Client-Side Enforcement of Server-Side Security

**Description:**The product is composed of a server that relies on the client to implement a mechanism that is intended to protect the server.
**Business Impact:**When the server relies on protection mechanisms placed on the client side, an attacker can modify the client-side behavior to bypass the protection

mechanisms resulting in potentially unexpected interactions between the client and server**.** When critical security decisions are made on the client side, data integrity can be compromised. If a messaging app relies solely on the client to determine message recipients, attackers can manipulate the client to send messages to unintended recipients, causing miscommunication or privacy breaches.This can lead to loss of data integrity,data breaches and reduced security control.

CWE-642 External Control of Critical State Data

CWE-646 Reliance on File Name or Extension of Externally-Supplied File

CWE-650 Trusting HTTP Permission Methods on the Server Side

CWE-653 Insufficient Compartmentalization

CWE-656 Reliance on Security Through Obscurity

CWE-657 Violation of Secure Design Principles

CWE-799 Improper Control of Interaction Frequency

CWE-807 Reliance on Untrusted Inputs in a Security Decision

CWE-840 Business Logic Errors

CWE-841 Improper Enforcement of Behavioral Workflow

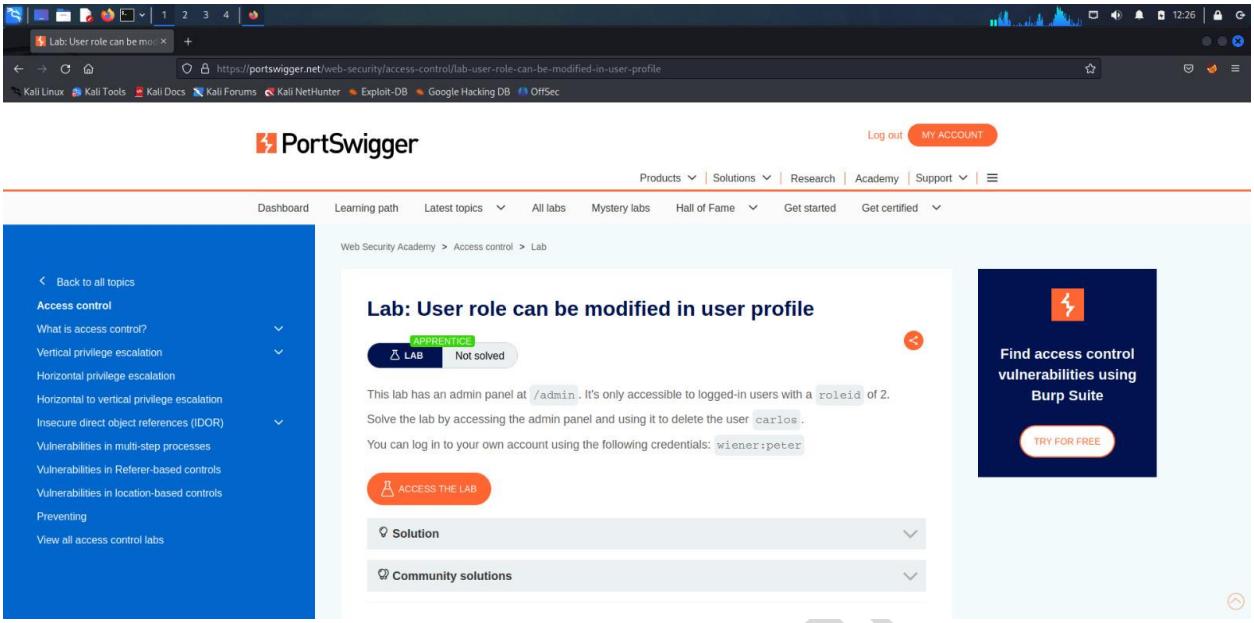
CWE-927 Use of Implicit Intent for Sensitive Communication

CWE-1021 Improper Restriction of Rendered UI Layers or Frames

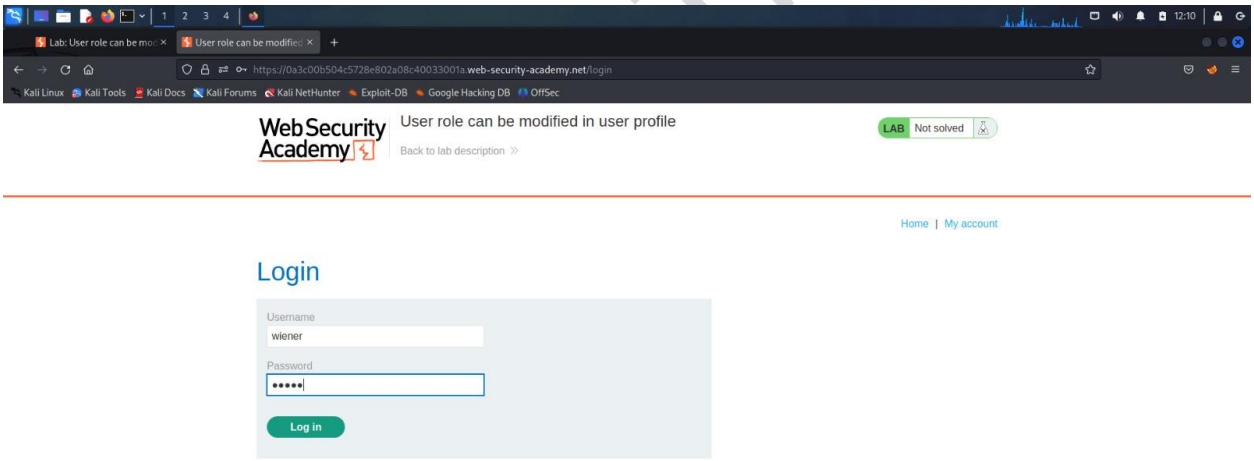
CWE-1173 Improper Use of Validation Framework

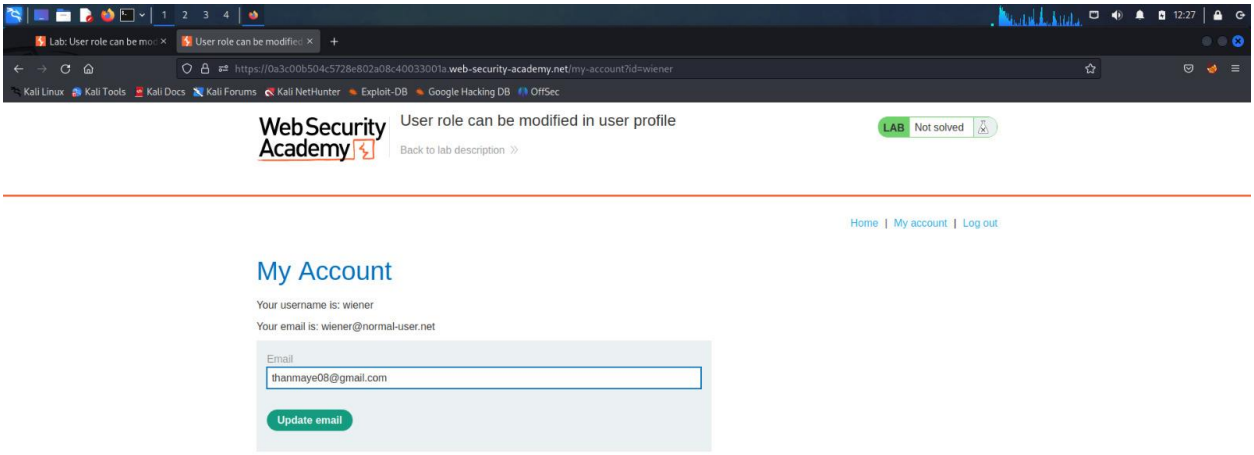
## Practical Approach:

Go to the portswigger website and select a lab.

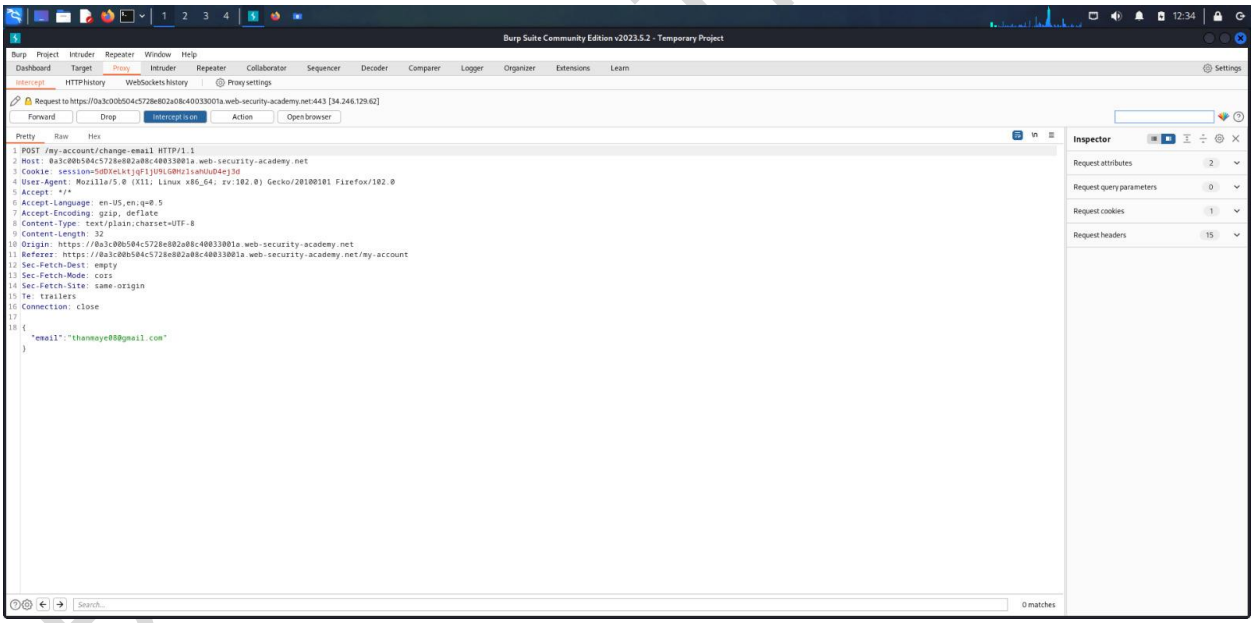



Click on access the lab ,go to my account type the credentials given in the lab description update your email id in place of their original email id

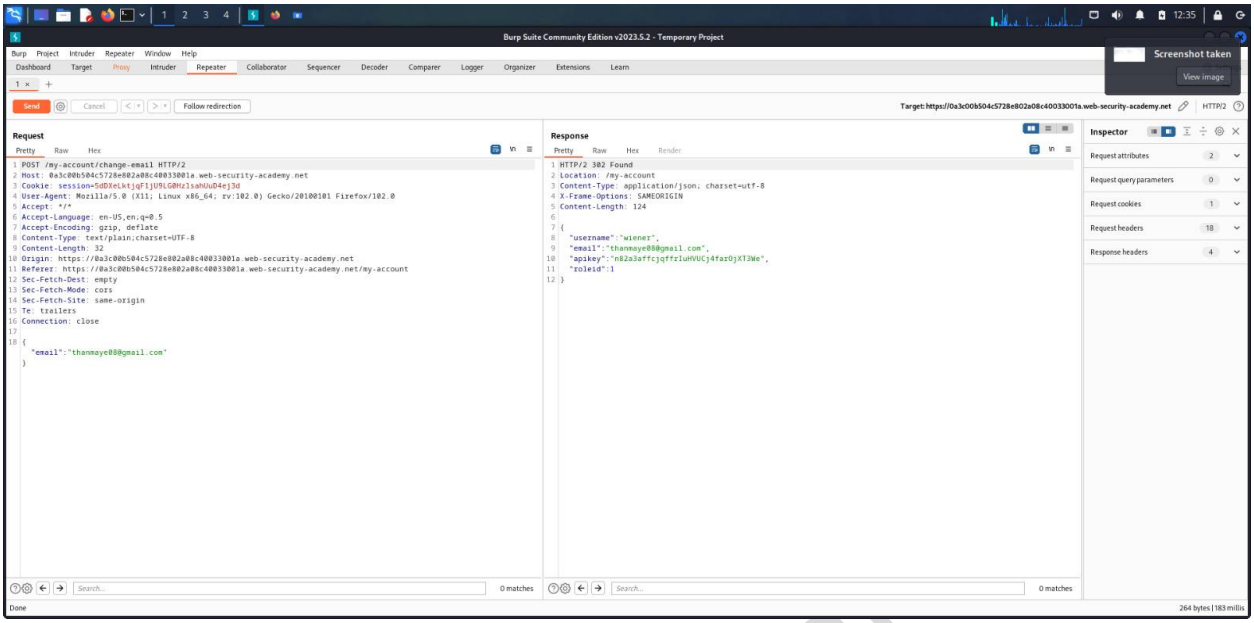




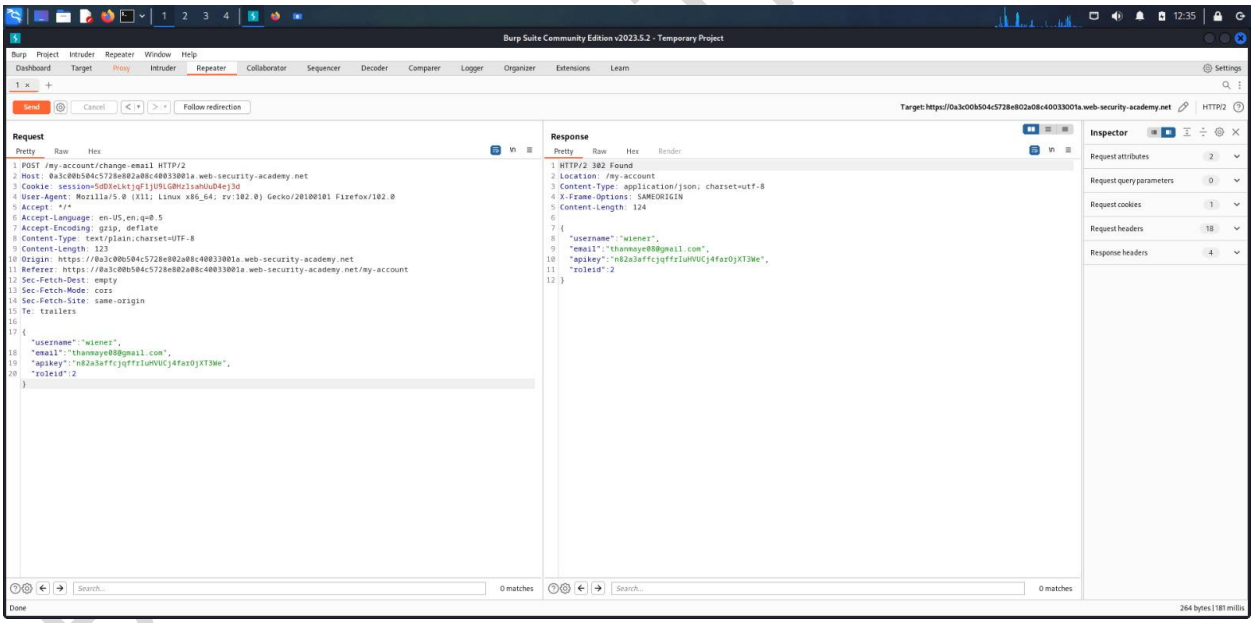

Using foxy proxy extension with burp open burp suite and on the intercept

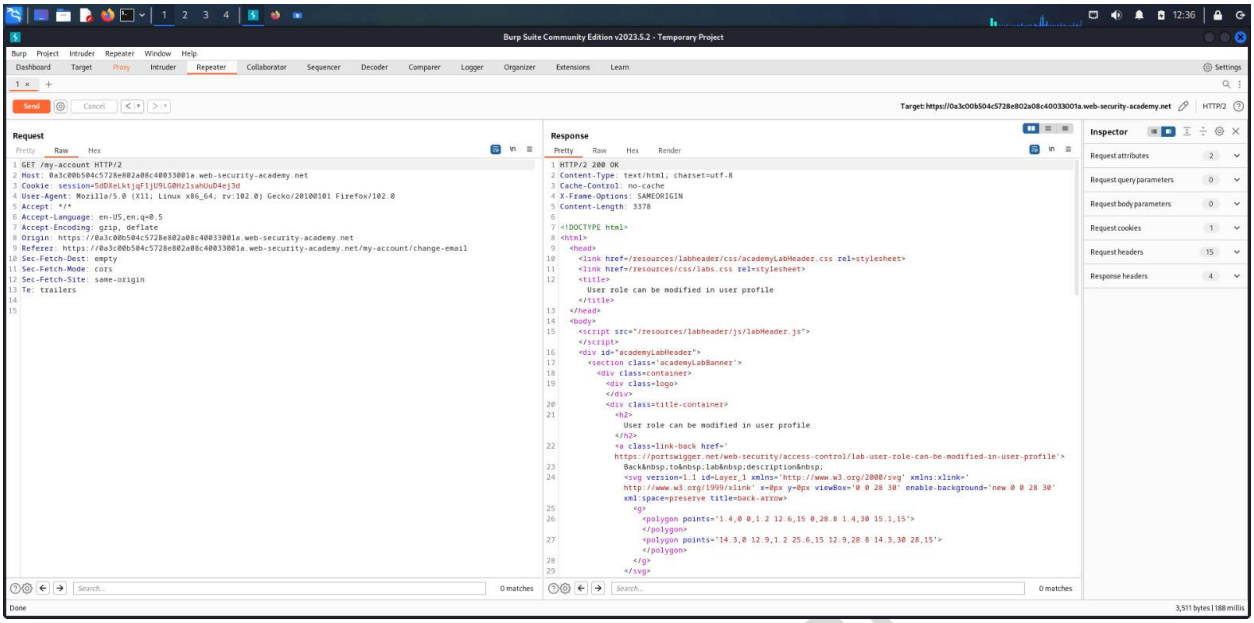



Right click on the repeater ,go to repeater and send a request

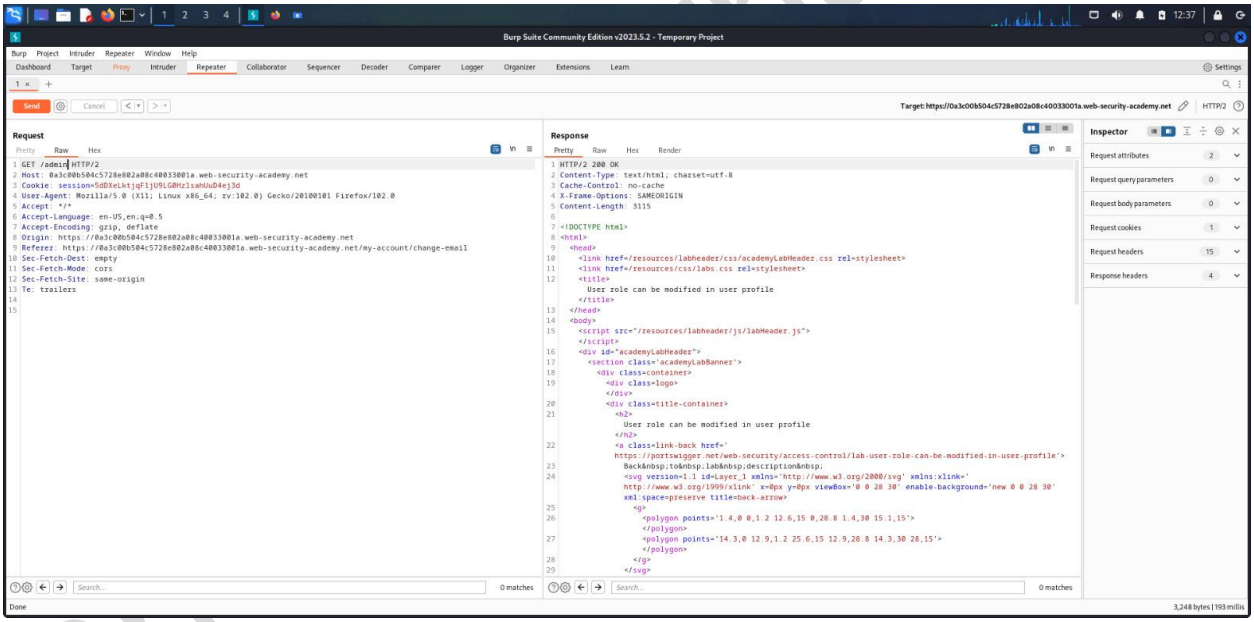



Now copy the given credentials in the response and replace it in the request,send the request and follow redirection

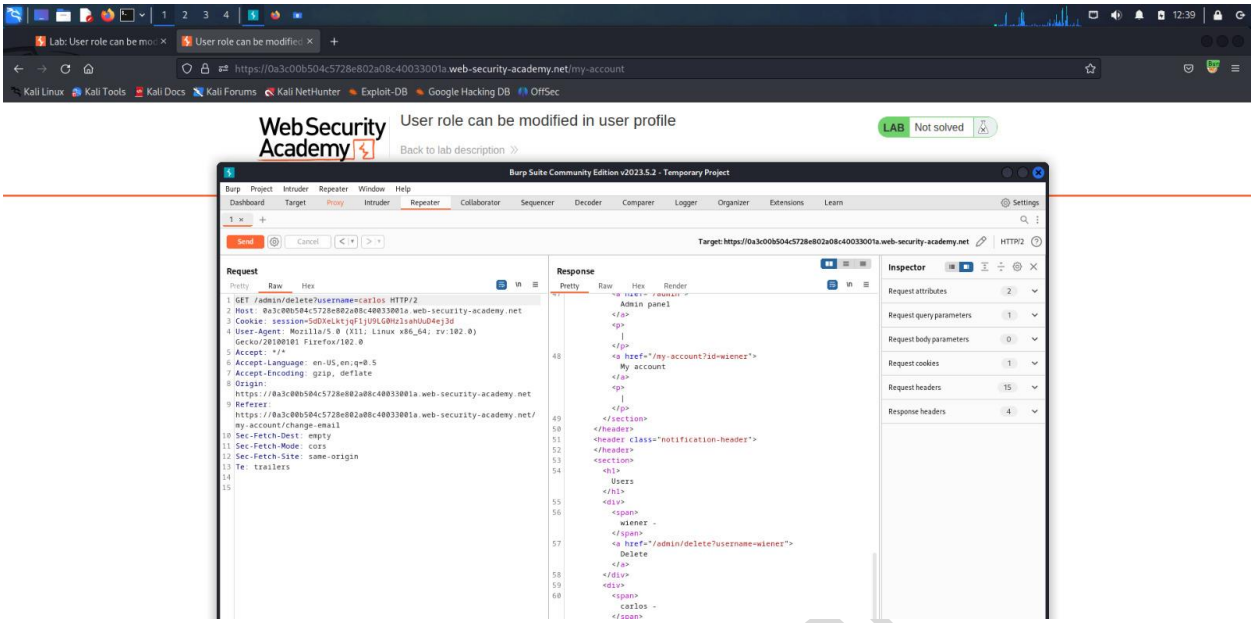




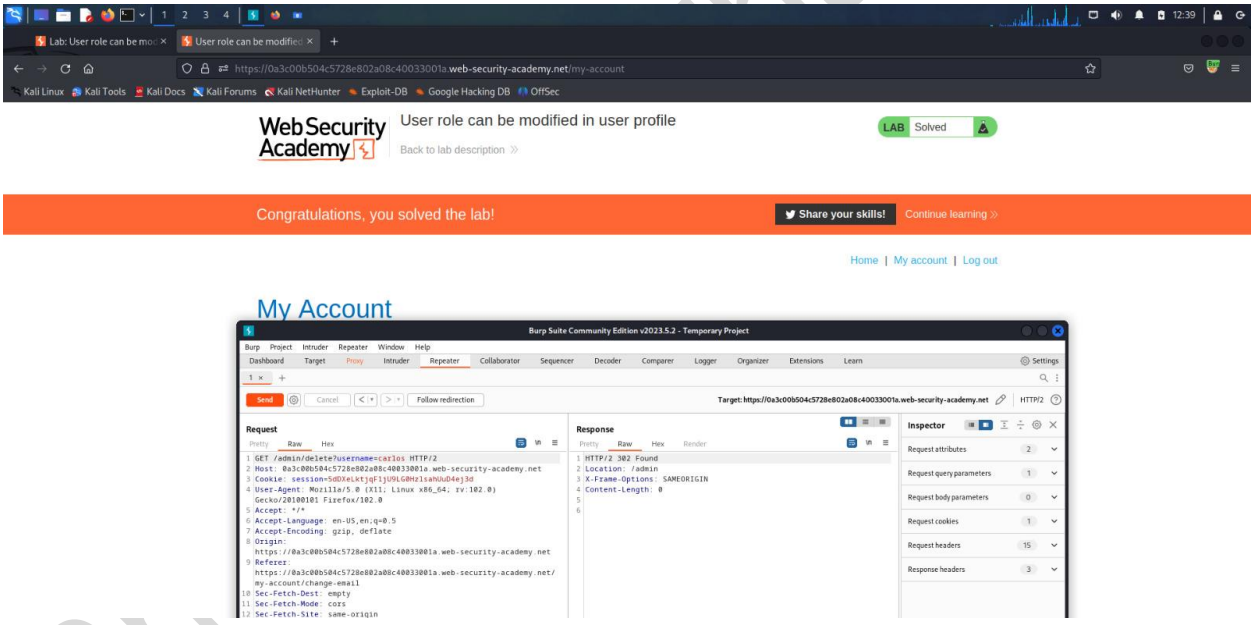

In the request replace /my-account to /admin




Now delete the carlos account using admin panel




Account got deleted



## 5. Security Misconfigurations:

Security misconfigurations arise when applications, servers, or databases are improperly configured, leaving them susceptible to attacks due to default settings, unnecessary services, or excessive permissions.

**Exploitation:**

Attackers exploit security misconfigurations to gain unauthorized access, exfiltrate data, deface websites, or launch Distributed Denial of Service (DDoS) attacks.

**Mitigation:** Regularly audit and assess application configurations, minimize unnecessary services and open ports, apply the principle of least privilege, and follow security hardening guidelines for servers and databases.

**Business Impact:** Security misconfigurations can lead to data breaches, service disruptions, financial loss, compliance violations, and damage to the organization's reputation.

**List of Mapped CWEs:**

CWE-2 7PK - Environment

CWE-11 ASP.NET Misconfiguration: Creating Debug Binary

CWE-13 ASP.NET Misconfiguration: Password in Configuration File

CWE-15 External Control of System or Configuration Setting

CWE-16 Configuration

CWE-260 Password in Configuration File

CWE-315 Cleartext Storage of Sensitive Information in a Cookie

CWE-520 .NET Misconfiguration: Use of Impersonation

CWE-526 Exposure of Sensitive Information Through Environmental Variables

CWE-537 Java Runtime Error Message Containing Sensitive Information

CWE-541 Inclusion of Sensitive Information in an Include File

CWE-547 Use of Hard-coded, Security-relevant Constants

**Description:**The product uses hard-coded constants instead of symbolic names for security-critical values, which increases the likelihood of mistakes during code maintenance or security policy change.
**Business Impact:**Relying on hard-coded values for security has some limitations on flexibility, complicating updates, increasing vulnerability to attacks, and potentially exposing sensitive data.

CWE-611 Improper Restriction of XML External Entity Reference

CWE-614 Sensitive Cookie in HTTPS Session Without 'Secure' Attribute

CWE-756 Missing Custom Error Page

CWE-776 Improper Restriction of Recursive Entity References in DTDs ('XML Entity Expansion')

CWE-942 Permissive Cross-domain Policy with Untrusted Domains

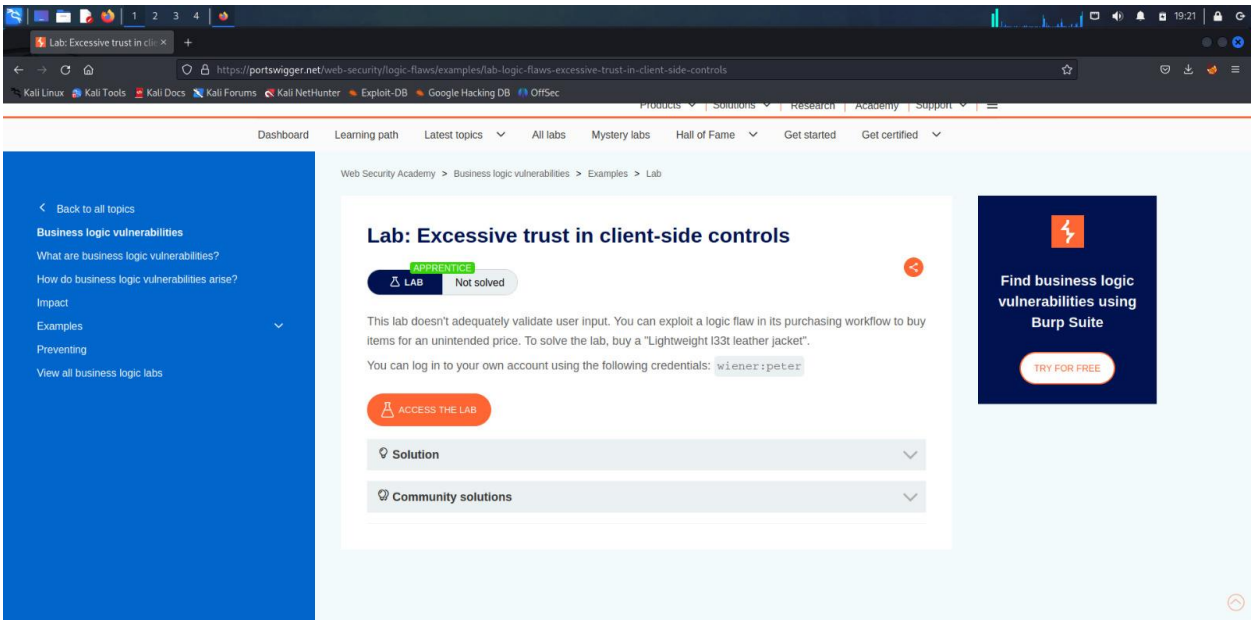
CWE-1004 Sensitive Cookie Without 'HttpOnly' Flag

CWE-1032 OWASP Top Ten 2017 Category A6 - Security Misconfiguration

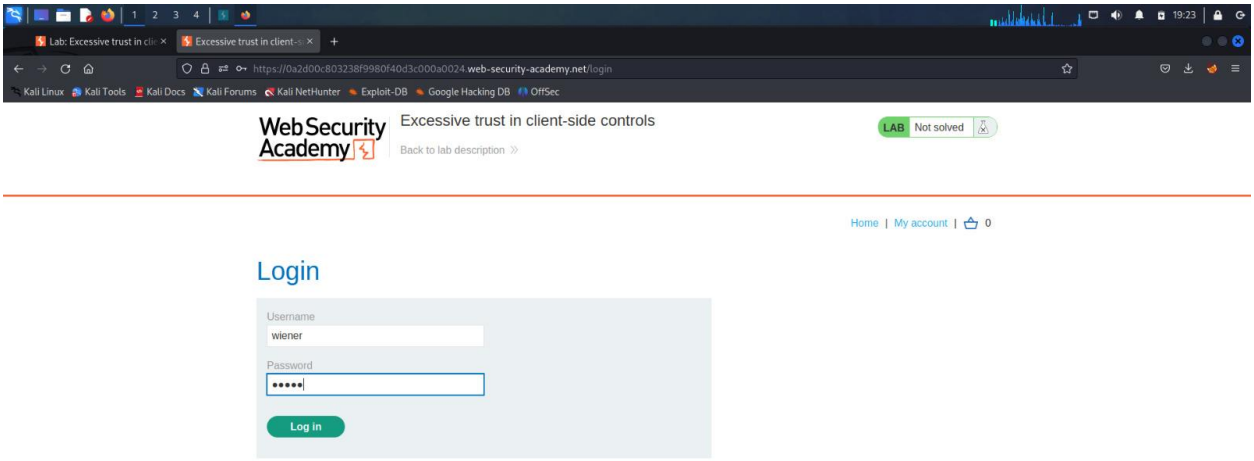
CWE-1174 ASP.NET Misconfiguration: Improper Model Validation

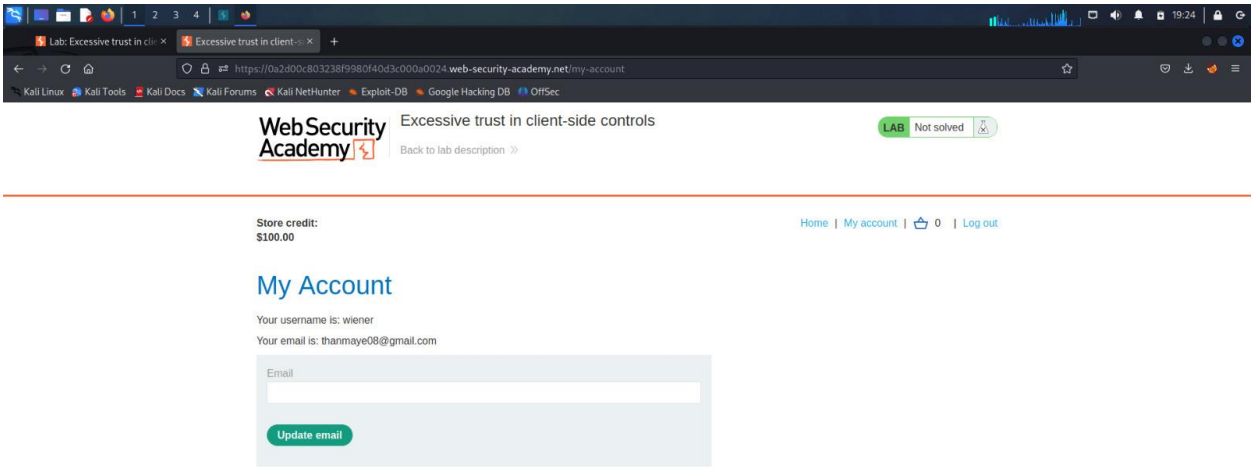
## Practical Approach:

Go to portswigger website and take which is suitable under security misconfiguration

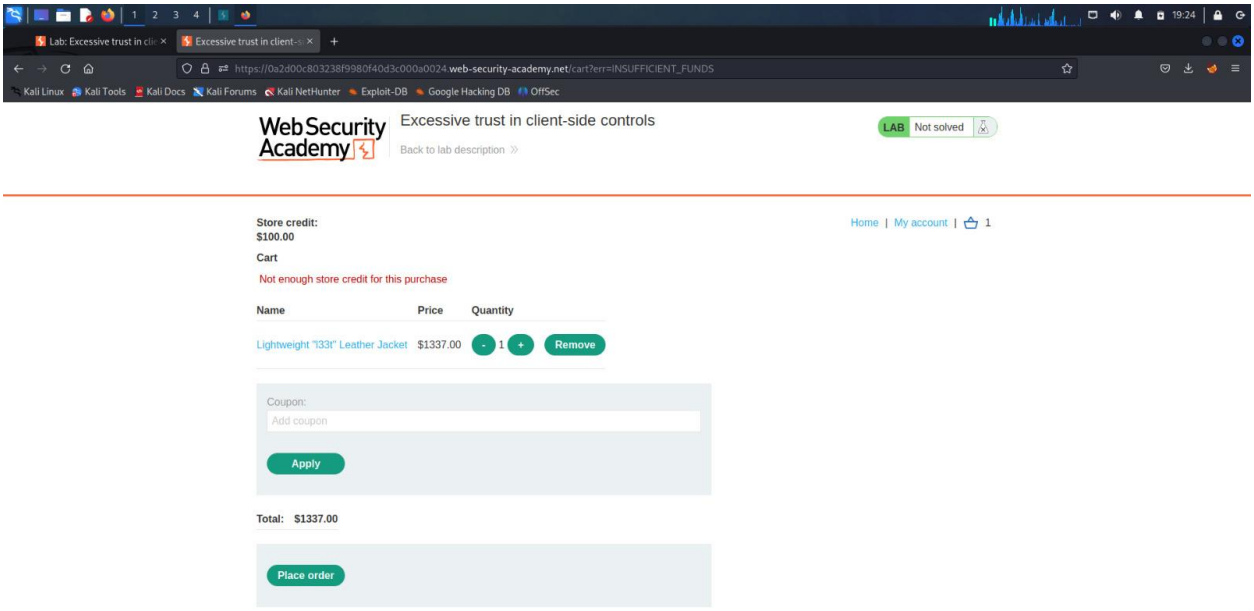



Access the lab and go to my account,give the credentials mentioned in the description of the lab and update your email id

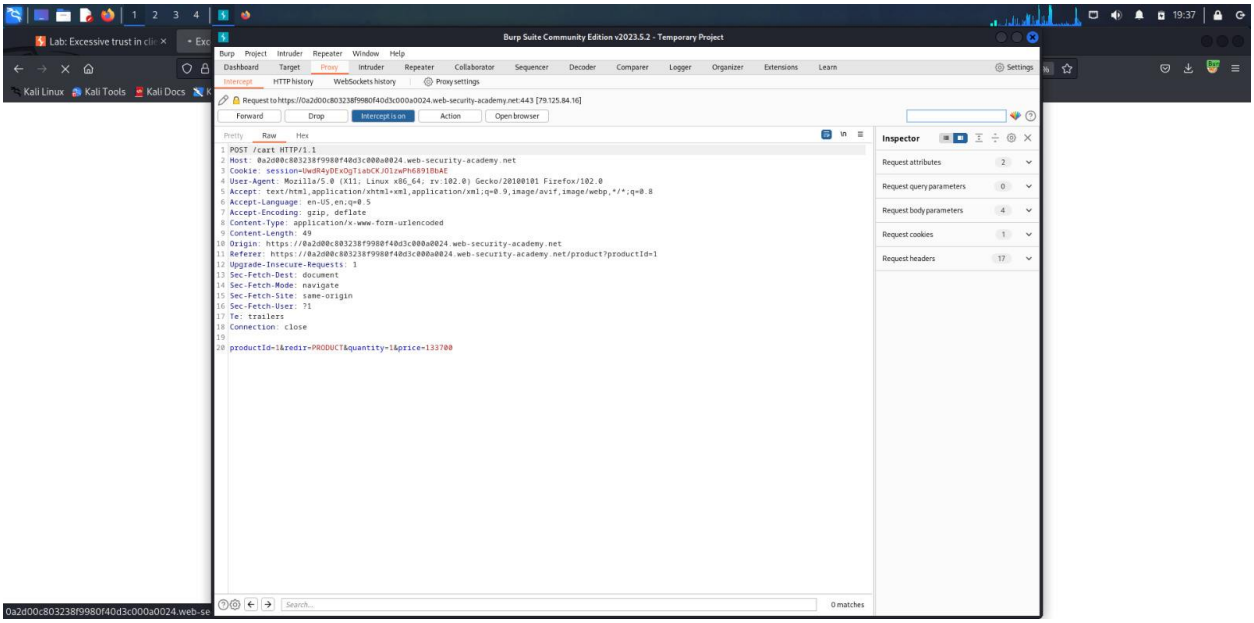





Now add leather jacket to your cart and try to purchase it will not happens due to insufficient balance

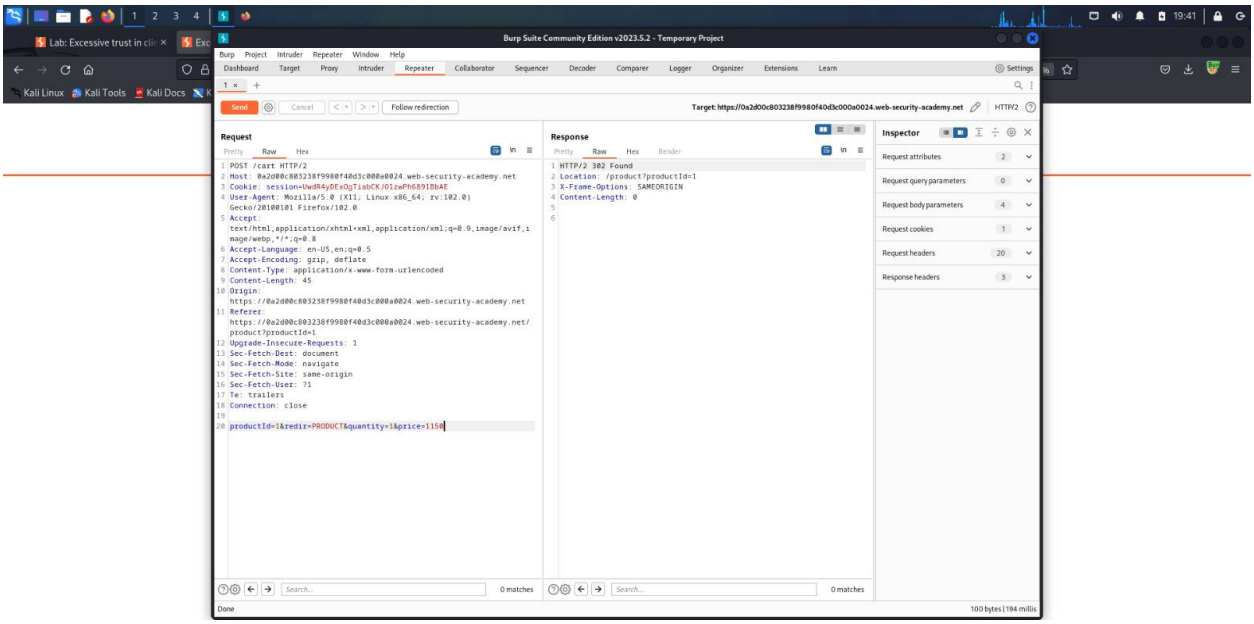


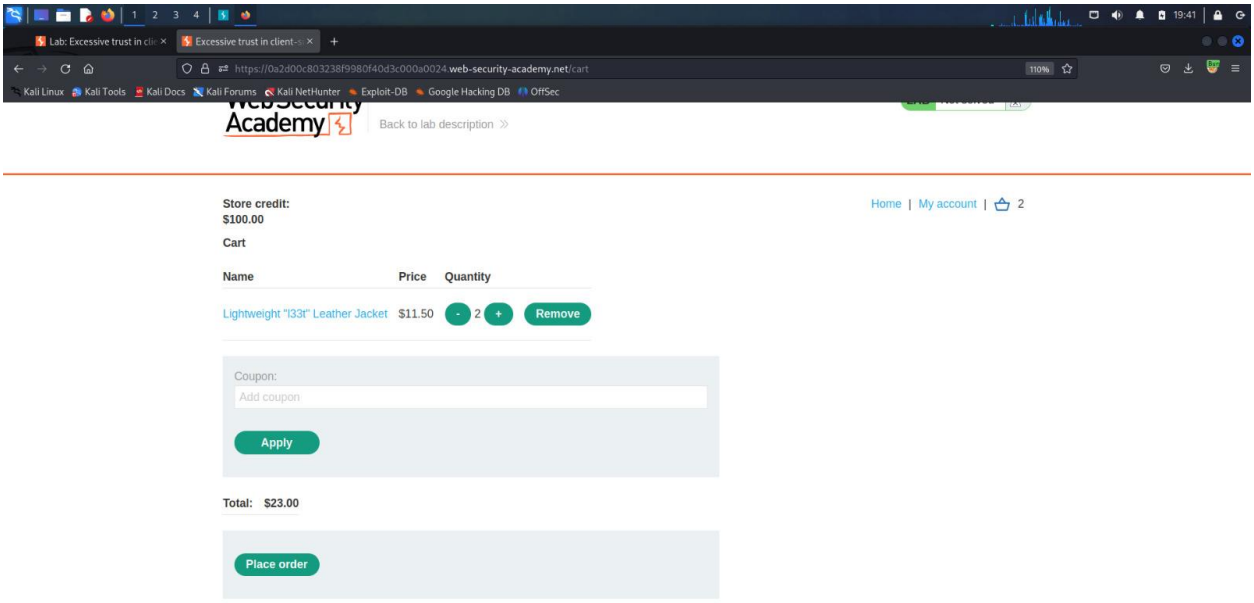

Now on proxy burp in the foxy proxy on the intercept,add to cart

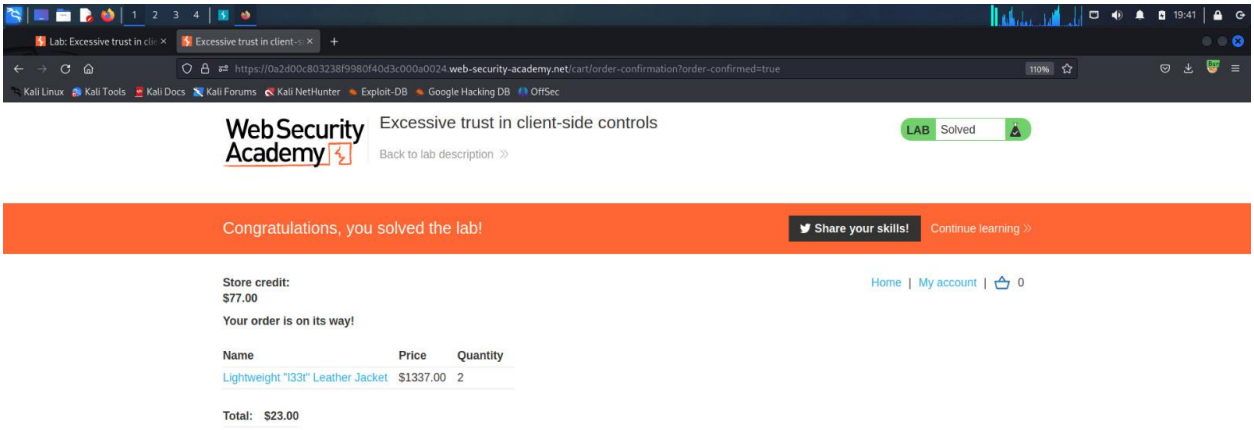



Now send to repeater and off the intercept

Change the value in the request and send a request .Now go to website cart and check the value




Lab: Excessive trust in cli
Excessive trust in client-
https://0a2d00c803238f9980f40d3c000a0024.web-security-academy.net/cart
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
Back to lab description
Store credit:
$100.00
Home | My account | 2
Cart
Name
Price
Quantity
Lightweight "l33t" Leather Jacket
$11.50
2
Remove
Coupon:
Add coupon
Apply
Total: $23.00
Place order


Lab: Excessive trust in cli
Excessive trust in client-
https://0a2d00c803238f9980f40d3c000a0024.web-security-academy.net/cart/order-confirmation?order-confirmed=true
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
Web Security Academy
Excessive trust in client-side controls
Back to lab description
LAB Solved
Congratulations, you solved the lab!
Share your skills!
Continue learning
Store credit:
$77.00
Home | My account | 0
Your order is on its way!
Name
Price
Quantity
Lightweight "l33t" Leather Jacket
$1337.00
2
Total: $23.00