Assignment 1

Name: Akshay Kumar Pandey

Registration Number: 21BCE10150

Campus: Bhopal

Branch: CSE Core

Assignment Description:

Take the top 5 OWAP vulnerabilities and explain any one cwe of that vulnerability.

1. Broken Access Control

This category focuses on vulnerabilities related to improper enforcement of access controls, leading to unauthorized access to sensitive resources or functionality. The Common Weakness Enumeration (CWE) is a list of software weaknesses and vulnerabilities, and it provides specific identifiers for various types of security issues. Here are some relevant CWEs related to the "Broken Access Control" category:

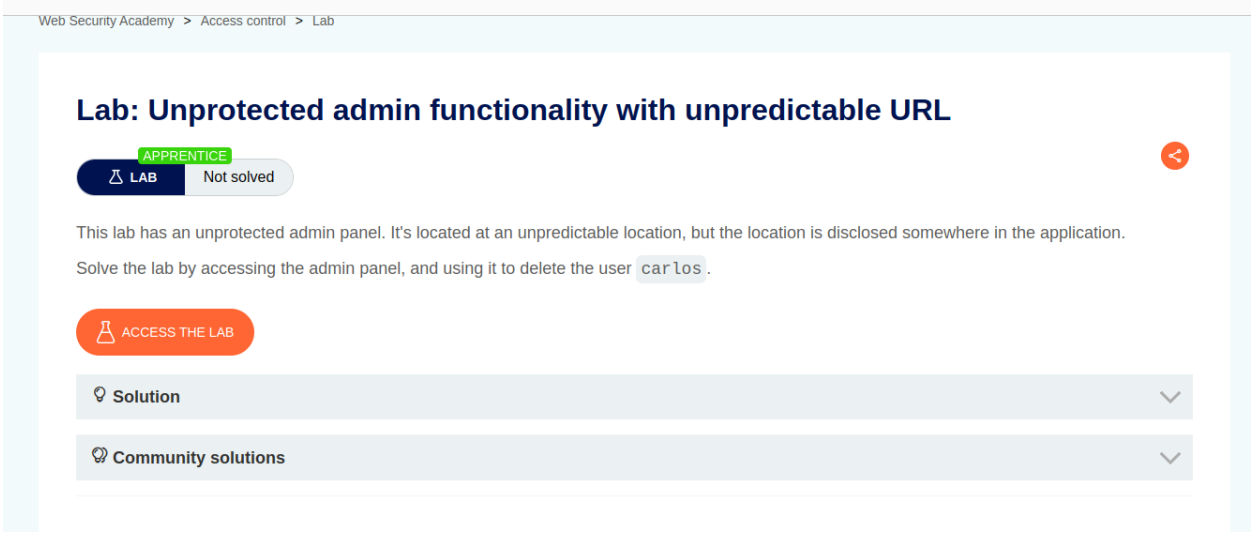
1. **CWE-284: Improper Access Control:** This weakness refers to situations where an attacker can access sensitive data or functionality without proper authorization. It encompasses various subcategories and variations of access control vulnerabilities.
2. **CWE-264:** Permissions, Privileges, and Access Controls
3. **CWE-285: Improper Authorization:** This weakness focuses on the failure to properly check whether a user has the necessary permissions to perform a particular action. It includes issues like missing authorization checks or not verifying user roles correctly.
4. **CWE-287: Improper Authentication:** While not strictly within the "Broken Access Control" category, this weakness relates to inadequate authentication mechanisms that can lead to unauthorized access.

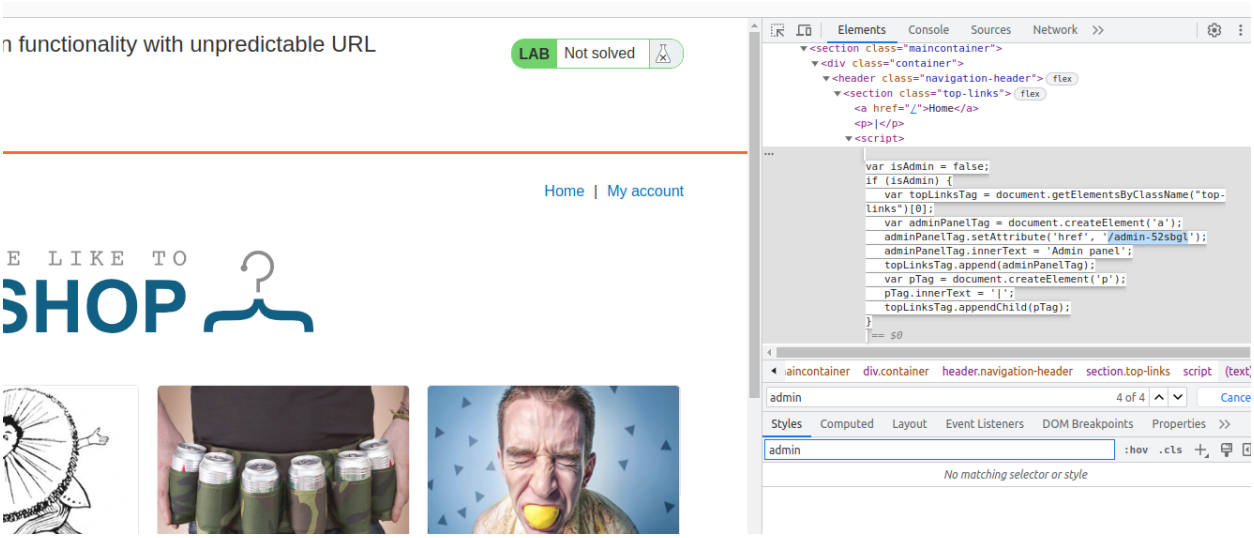
5. **CWE-639: Authorization Bypass Through User-Controlled Key:** This weakness occurs when an attacker can bypass authorization checks by providing or manipulating keys, tokens, or credentials that the application uses for access control.
6. **CWE-749: Exposed Dangerous Method or Function:** This weakness involves exposing sensitive methods or functions that can be called by unauthorized users, leading to improper access.
7. **CWE-862: Missing Authorization:** This weakness refers to scenarios where the application fails to perform authorization checks altogether, allowing unauthorized users to access restricted resources.
8. **CWE-863: Incorrect Authorization:** Similar to missing authorization, this weakness pertains to situations where the application performs authorization checks but does so incorrectly, leading to unauthorized access.
9. **CWE-864: Access to a Resource Using Incompatible Permissions or Privileges:** This weakness involves granting users access to resources with permissions or privileges that don't match their intended level of authorization.
10. **CWE-276: Incorrect Default Permissions:** This weakness is about setting insecure default permissions for resources, allowing unauthorized users to access them.
11. **CWE-284: Improper Enforcement of Message or Data Structure:** This weakness relates to cases where an attacker can manipulate messages or data structures to gain unauthorized access.

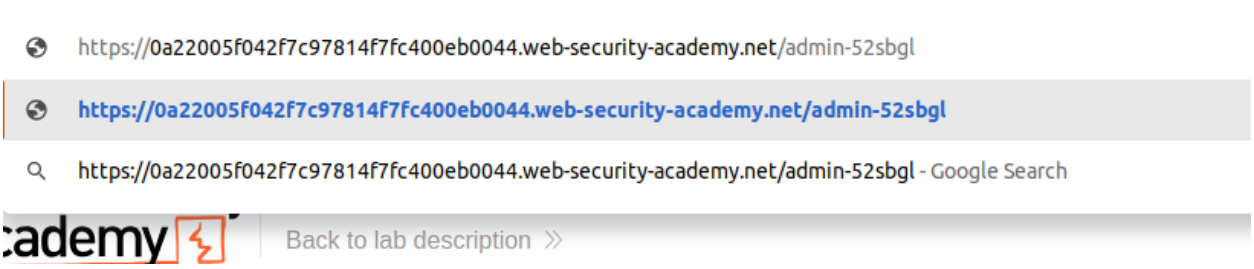
Business Impact: can harm businesses by enabling unauthorized users to access sensitive data, causing data breaches, financial loss, reputation damage, legal issues, operational disruptions, intellectual property theft, and increasing insider threat risks.

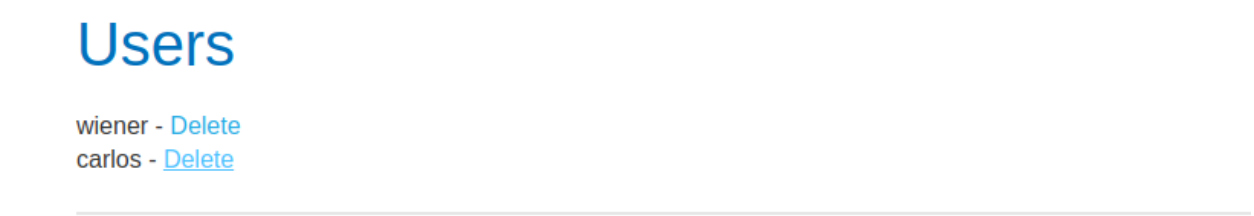
To demonstrate one of these CWEs, let's look at CWE-284: Improper Access Control. The product does not restrict or incorrectly restricts access to a resource from an unauthorized actor.

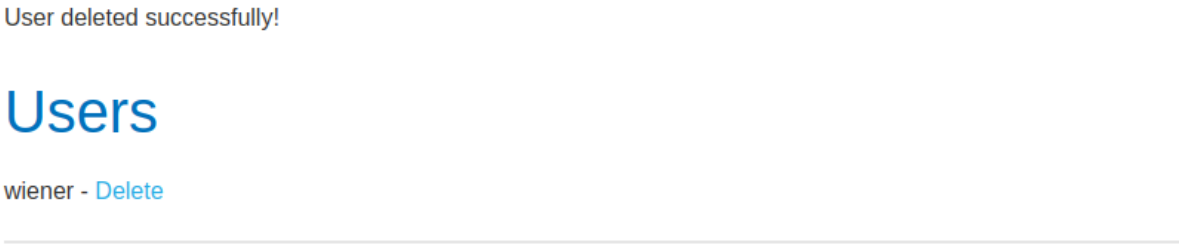
Here is a screenshot of a webpage that is vulnerable to CWE-284:


Web Security Academy > Access control > Lab
Lab: Unprotected admin functionality with unpredictable URL
APPRENTICE
LAB
Not solved
This lab has an unprotected admin panel. It's located at an unpredictable location, but the location is disclosed somewhere in the application.
Solve the lab by accessing the admin panel, and using it to delete the user carlos.
ACCESS THE LAB
Solution
Community solutions


n functionality with unpredictable URL
LAB
Not solved
Home | My account
E LIKE TO
SHOP
Elements
Console
Sources
Network
var isAdmin = false;
if (isAdmin) {
var topLinksTag = document.getElementsByClassName("top-links")[0];
var adminPanelTag = document.createElement('a');
adminPanelTag.setAttribute('href', '/admin-52sbgl');
adminPanelTag.innerText = 'Admin panel';
topLinksTag.append(adminPanelTag);
var pTag = document.createElement('p');
pTag.innerText = '|';
topLinksTag.appendChild(pTag);
}
admin
Styles
Computed
Layout
Event Listeners
DOM Breakpoints
Properties
No matching selector or style


https://0a22005f042f7c97814f7fc400eb0044.web-security-academy.net/admin-52sbgl
https://0a22005f042f7c97814f7fc400eb0044.web-security-academy.net/admin-52sbgl
https://0a22005f042f7c97814f7fc400eb0044.web-security-academy.net/admin-52sbgl - Google Search
Back to lab description


Users
wiener - Delete
carlos - Delete



Improper access controls can help attacker in altering or permanently deleting an existing user.

To fix this vulnerability, the website should implement security controls to prevent unauthorized users from logging in. These controls could include:

- Requiring users to enter a strong password
- Enabling two-factor authentication
- Restricting access to the website based on IP address
- Logging all login attempts

By implementing these security controls, the website can help to prevent attackers from exploiting CWE-264 and gaining unauthorized access to the data.

2. Cryptographic Failure

This category focuses on vulnerabilities related to weak or inadequate cryptographic practices that can lead to the compromise of sensitive data. The Common Weakness Enumeration (CWE) is a list of software weaknesses and vulnerabilities, and it provides specific identifiers for various types of security issues. Here are some relevant CWEs related to the "Cryptographic Failures" category:

1. **CWE-310: Cryptographic Issues:** This is a general category that encompasses various cryptographic weaknesses, including weak algorithms, insecure key

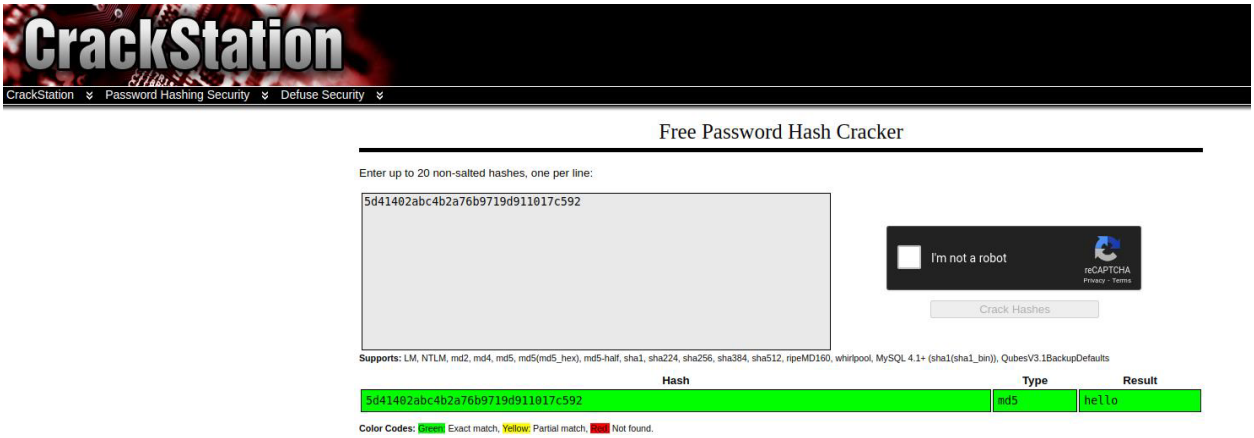
management, and improper usage of cryptographic functions.

2. **CWE-327: Use of a Broken or Risky Cryptographic Algorithm:** This weakness refers to using cryptographic algorithms that are known to be insecure or vulnerable to attacks.
3. **CWE-328: Reversible One-Way Hash:** This weakness involves using a reversible hash function (one that can be decrypted) instead of a one-way hash function, compromising the security of hashed data.
4. **CWE-329: Not Using a Random IV with CBC Mode:** This weakness pertains to using the Cipher Block Chaining (CBC) mode of encryption without a random Initialization Vector (IV), making the encryption vulnerable to certain attacks.
5. **CWE-331: Insufficient Entropy:** This weakness occurs when cryptographic keys, random values, or other sensitive data are generated with insufficient entropy, making them easier to guess or predict.
6. **CWE-332: Insufficient Entropy in PRNG:** Similar to insufficient entropy, this weakness involves using a pseudo-random number generator (PRNG) with insufficient entropy, resulting in predictable random values.
7. **CWE-335: PRNG Seed Error:** This weakness occurs when a pseudo-random number generator's (PRNG) initial seed is not generated securely, leading to predictable PRNG outputs.
8. **CWE-336: Same Seed in PRNG:** This weakness involves using the same initial seed for multiple instances of a pseudo-random number generator (PRNG), leading to predictable patterns in the generated values.
9. **CWE-337: Predictable Seed in PRNG:** This weakness occurs when the seed for a pseudo-random number generator (PRNG) can be easily predicted by an attacker, compromising the randomness of generated values.
10. **CWE-338: Use of Cryptographically Weak Pseudo-Random Number Generator (PRNG):** This weakness involves using a PRNG that is known to be cryptographically weak and predictable, leading to vulnerabilities in cryptographic operations.
11. **CWE-343: Non-Cryptographic Hash with Hardcoded Seed:** This weakness pertains to using non-cryptographic hash functions with hardcoded seeds, making them susceptible to collision attacks.

Business Impact: can seriously impact businesses by compromising data security, leading to potential breaches, loss of trust, regulatory penalties, and operational disruptions.

To demonstrate one of these CWEs, let's look at CWE-327: Use of a Broken or Risky Cryptographic Algorithm. The product uses a broken or risky cryptographic algorithm or protocol.

Here is a screenshot of a webpage that is vulnerable to CWE-327:



To fix this vulnerability, the website should use a stronger cryptographic algorithm to encrypt passwords. Some examples of strong cryptographic algorithms include SHA-256 and AES-256.

By using a stronger cryptographic algorithm, the website can help to protect sensitive data from being compromised by attackers.

3. Injection

This category focuses on vulnerabilities related to code injection attacks, where malicious input is inserted into a program or system and executed. These vulnerabilities can lead to data exposure, unauthorized access, and remote code execution. The Common Weakness Enumeration (CWE) is a list of software weaknesses and

vulnerabilities, and it provides specific identifiers for various types of security issues. Here are some relevant CWEs related to the "Injection" category:

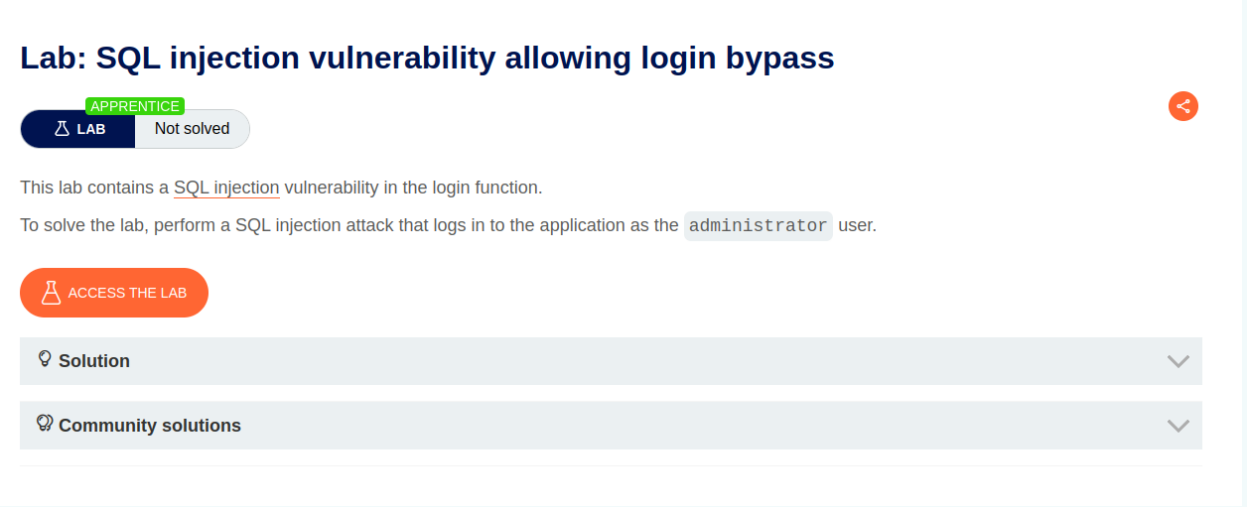
1. **CWE-89: SQL Injection:** This is a common injection vulnerability where an attacker can manipulate SQL queries by inserting malicious SQL code. This can lead to unauthorized access, data exposure, and potentially remote code execution.
2. **CWE-77: Improper Neutralization of Special Elements used in a Command ('Command Injection'):** This weakness involves injecting malicious commands into a system's command-line interface, potentially leading to unauthorized access and remote code execution.
3. **CWE-116: Improper Encoding or Escaping of Output:** While not a direct injection vulnerability, failing to properly encode or escape output data can lead to cross-site scripting (XSS) attacks, which involve injecting malicious scripts into web applications.
4. **CWE-78: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection'):** Similar to command injection, this involves injecting malicious commands into operating system commands, potentially leading to unauthorized access and remote code execution.
5. **CWE-22: Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal'):** This involves manipulating file paths to access files outside of the intended directory, leading to unauthorized access and data exposure.
6. **CWE-94: Improper Control of Generation of Code ('Code Injection'):** This involves injecting malicious code into a program, which can lead to remote code execution and unauthorized access.
7. **CWE-185: Incorrect Regular Expression:** This vulnerability occurs when a regular expression is not properly defined or constrained, leading to unexpected behavior and potential security issues.
8. **CWE-643: Improper Neutralization of Data within XPath Expressions ('XPath Injection'):** This is similar to SQL injection but involves manipulating XPath expressions to access unauthorized data.
9. **CWE-94: Improper Control of Generation of Code ('Code Injection') - Secondary:** This is a secondary entry for code injection weaknesses that pertain to languages and technologies other than C/C++.

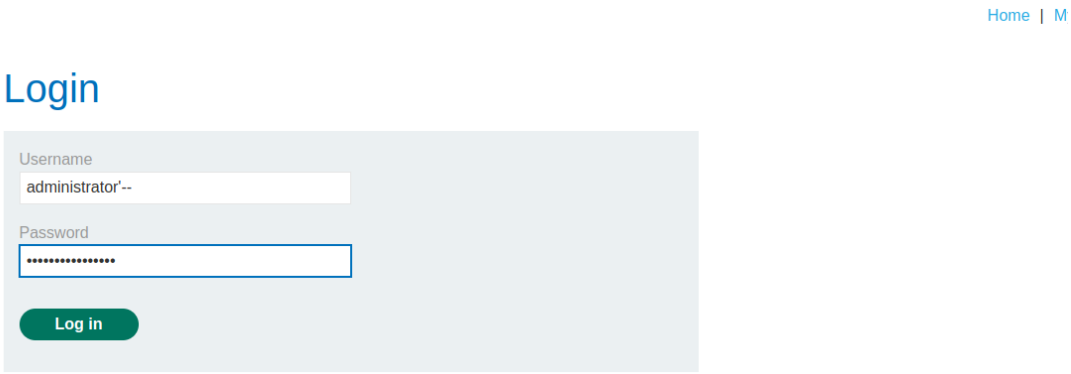
10. **CWE-113: Improper Neutralization of CRLF Sequences ('CRLF Injection'):** This involves injecting CRLF (Carriage Return Line Feed) characters into input to manipulate the behavior of systems that use these characters for formatting.
11. **CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting') (XSS):**This weakness involves injecting malicious scripts into web pages that
are then executed by other users. Cross-site scripting (XSS) attacks can lead to data theft, session hijacking, and unauthorized actions in the context of the victim's browser.

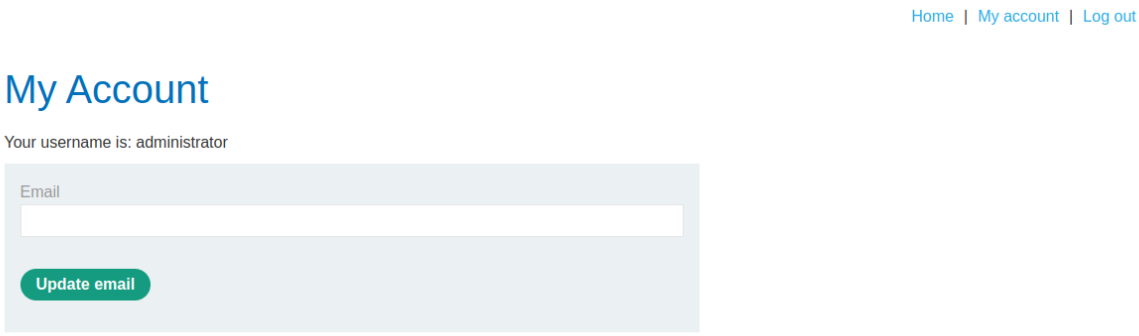
Business Impact: allows attackers to manipulate database queries through vulnerabilities in Hibernate, potentially leading to unauthorized data access, breaches, financial losses, legal issues, and reputation damage.

To demonstrate one of these CWEs, let's look at CWE-564: SQL Injection: Hibernate. Using Hibernate to execute a dynamic SQL statement built with user-controlled input can allow an attacker to modify the statement's meaning or to execute arbitrary SQL commands.

Here is a screenshot of a webpage that is vulnerable to CWE-564:







SQL Statements can be used to modify the table and gain access to users' data.

This webpage does not have any validation in place to prevent malicious code from being entered into the page. This means that an attacker could easily enter malicious code into the page and execute it on the website's server.

To fix this vulnerability, the website should validate the input that is entered into the search bar. This could be done by using a regular expression to check for malicious characters.

By validating the input, the website can help to prevent attackers from exploiting CWE-79 and executing malicious code on the server.

4. Insecure Design

This category focuses on vulnerabilities that arise from poor design decisions, leading to systemic weaknesses that are difficult to address solely through coding practices. The Common Weakness Enumeration (CWE) is a list of software weaknesses and vulnerabilities, and it provides specific identifiers for various types of security issues. Here are some relevant CWEs related to the "Insecure Design" category:

1. **CWE-285: Improper Authorization:** This weakness involves failing to properly check whether a user has the necessary permissions to perform a particular action. It can stem from a design that doesn't account for proper authorization checks.
2. **CWE-798: Use of Hard-coded Credentials:** This weakness occurs when sensitive credentials, such as passwords or API keys, are hard-coded into the application's design, making them accessible to attackers.
3. **CWE-306: Missing Authentication for Critical Function:** This weakness involves missing authentication checks for critical functions or resources, potentially allowing unauthorized access.
4. **CWE-862: Missing Authorization:** This weakness pertains to scenarios where the application fails to perform authorization checks altogether, allowing unauthorized users to access restricted resources.
5. **CWE-863: Incorrect Authorization:** Similar to missing authorization, this weakness involves situations where the application performs authorization checks but does so incorrectly, leading to unauthorized access.
6. **CWE-927: Use of Implicit Permissions for UI Elements:** This weakness occurs when UI elements (buttons, links, etc.) are designed with implicit permissions that allow unauthorized users to access sensitive functionality.
7. **CWE-657: Violation of Secure Design Principles:** This weakness involves design decisions that violate established secure design principles, such as the principle of least privilege or separation of concerns.
8. **CWE-693: Protection Mechanism Failure:** This weakness occurs when protection mechanisms, such as authentication or authorization, fail due to improper design or implementation.
9. **CWE-895: Failure to Encapsulate Behavior:** This weakness involves failing to encapsulate sensitive behavior or logic within appropriate components or layers of

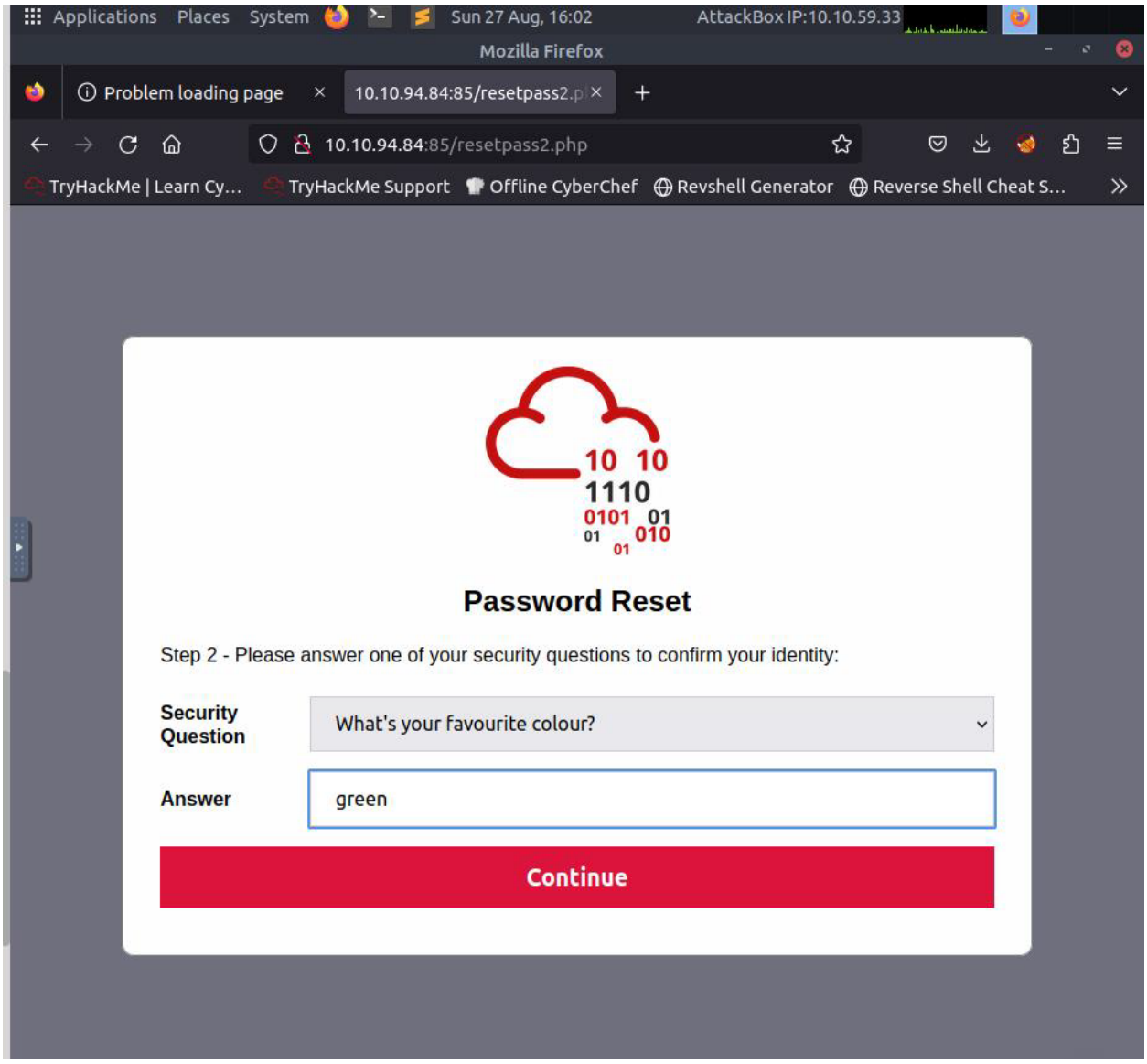
the application, leading to vulnerabilities.

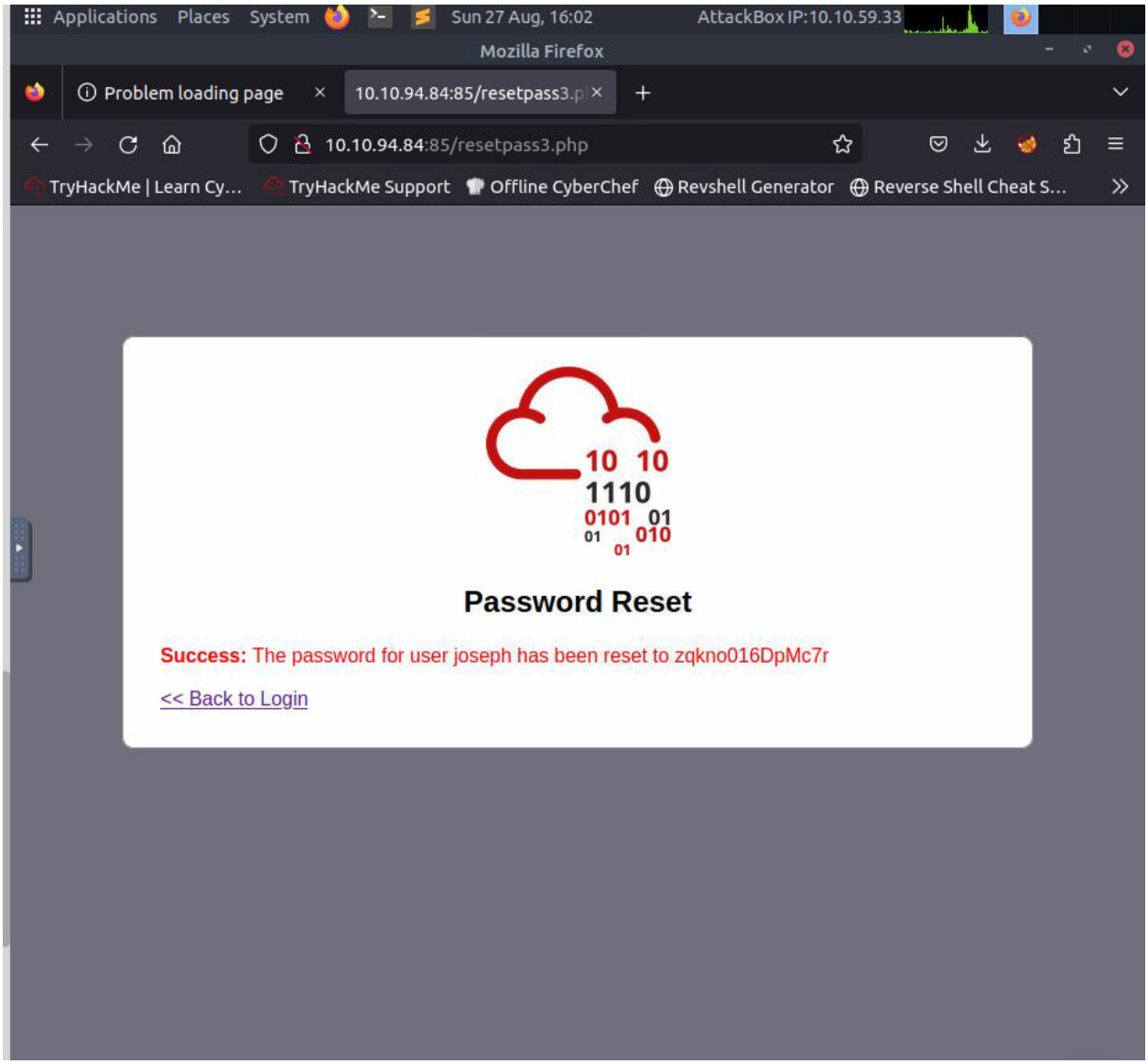
10. **CWE-915: Improperly Controlled Modification of Dynamically-Determined Object Attributes:** This weakness pertains to design flaws that allow attackers to modify attributes of objects in unexpected ways, leading to unauthorized access or manipulation.
11. **CWE-1027: Insufficient Logging:** While not strictly within the "Insecure Design" category, insufficient or inadequate logging design can lead to issues like insufficient monitoring, delayed detection of attacks, and difficulties in forensic analysis.

Business Impact: has substantial business implications due to its potential to result in vulnerabilities and weak security structures. This can lead to data breaches, financial losses, reputational damage, and regulatory penalties.

To demonstrate one of these CWEs, let's look at CWE-657: Violation of Secure Design Principles. The product violates well-established principles for secure design.

Here is a screenshot of a webpage that is vulnerable to CWE-657:


Applications Places System
Sun 27 Aug, 16:02
AttackBox IP:10.10.59.33
Mozilla Firefox
Problem loading page
10.10.94.84:85/resetpass2.p
10.10.94.84:85/resetpass2.php
TryHackMe | Learn Cy...
TryHackMe Support
Offline CyberChef
Revshell Generator
Reverse Shell Cheat S...
Password Reset
Step 2 - Please answer one of your security questions to confirm your identity:
Security Question
What's your favourite colour?
Answer
green
Continue



Violation of security design principles includes poor security designing. The chances of an attacker stealing passwords is higher when poor security related questions are set.

To fix this vulnerability, the website should store passwords in a secure manner, such as using a hashing algorithm.

By storing passwords securely, the website can help to prevent attackers from exploiting CWE-657 and gaining unauthorized access to the website.

5. Security Misconfiguration

This category focuses on vulnerabilities that arise from insecure configurations of applications, APIs, frameworks, and other components. The Common Weakness Enumeration (CWE) is a list of software weaknesses and vulnerabilities, and it provides specific identifiers for various types of security issues. Here are some relevant CWEs related to the "Security Misconfiguration" category:

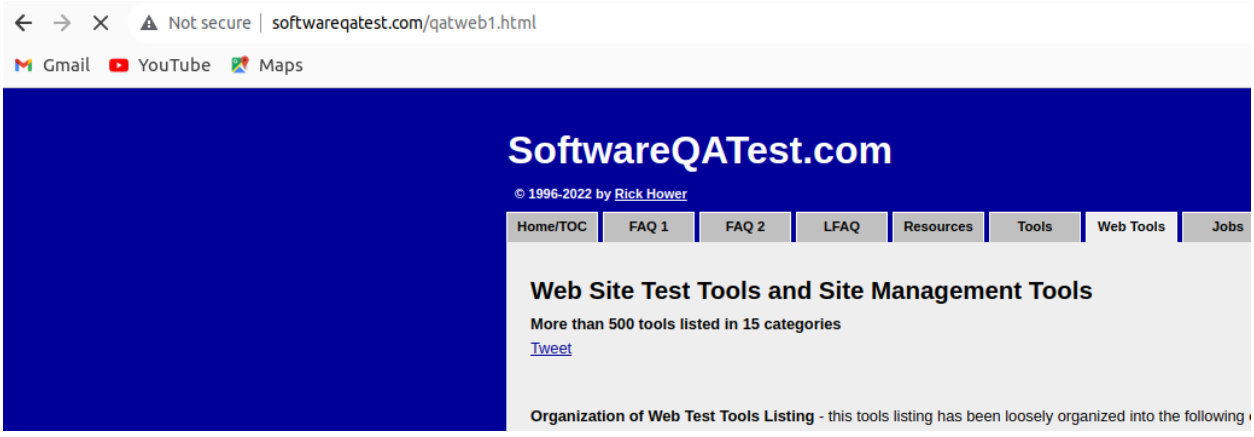
1. **CWE-15: External Control of System or Configuration Setting:** This weakness occurs when an application allows attackers to manipulate system settings or configurations that can lead to security vulnerabilities.
2. **CWE-20: Improper Input Validation:** While not directly under the "Security Misconfiguration" category, inadequate input validation can result in misconfigurations and security issues.
3. **CWE-76: Insecure Deployment:** This weakness pertains to deploying software or systems with insecure configurations, making them vulnerable to attacks.
4. **CWE-116: Improper Encoding or Escaping of Output:** Similar to input validation, improper encoding or escaping of output can lead to security misconfigurations that enable attacks like cross-site scripting (XSS).
5. **CWE-287: Improper Authentication:** Incorrectly configured authentication mechanisms can lead to unauthorized access and security misconfigurations.
6. **CWE-326: Inadequate Encryption Strength:** Weak encryption configurations can result in inadequate protection of sensitive data.
7. **CWE-345: Insufficient Verification of Data Authenticity:** Misconfigured verification of data authenticity can lead to security issues such as man-in-the-middle attacks.
8. **CWE-347: Improper Verification of Cryptographic Signature:** This weakness involves improper verification of cryptographic signatures, leading to security misconfigurations and potential data breaches.
9. **CWE-352: Cross-Site Request Forgery (CSRF):** CSRF attacks can be facilitated by insecure configurations that allow attackers to trick users into performing actions unintentionally.

10. **CWE-427: Uncontrolled Search Path Element:** Insecure search paths can lead to the loading of unintended or malicious code.
11. **CWE-428: Unquoted Search Path or Element:** Similar to uncontrolled search paths, unquoted search paths can lead to misconfigurations and unintended code execution.
12. **CWE-444: Inconsistent Interpretation of HTTP Requests ('HTTP Request Smuggling'):** This weakness involves misconfigurations that lead to inconsistent interpretation of HTTP requests, potentially enabling attacks.
13. **CWE-798: Use of Hard-coded Credentials:** This weakness occurs when sensitive credentials, such as passwords or API keys, are hard-coded into the application's configuration.

Business Impact: can harm businesses by exposing confidential data during transmission, leading to data breaches, compromised customer trust, regulatory violations, and potential legal consequences.

To demonstrate one of these CWEs, let's look at CWE-319: Cleartext Transmission of Sensitive Information. The product transmits sensitive or security-critical data in cleartext in a communication channel that can be sniffed by unauthorized actors.

Here is a screenshot of a webpage that is vulnerable to CWE-319:



Websites with http allows attackers to steal sensitive information.

In the absence of an SSL certificate, all your communications are not encrypted at all. Meaning attackers have access to all the data.

Sensitive Information must only be channelled through HTTPS communicatons.