PRANAVASRI RM

ASSIGNMENT – 1

AI with Cyber Security

What is OWASP top 10 category?

The OWASP Top Ten is a widely recognized and periodically updated list of the ten most critical security risks that affect web applications. It is curated by the Open Web Application Security Project (OWASP), a nonprofit organization focused on improving software security. The OWASP Top Ten provides guidance to developers, security professionals, and organizations on identifying and mitigating the most significant security vulnerabilities commonly found in web applications.

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. The list is intended to raise awareness about these common security risks, facilitate better understanding, and promote the adoption of secure coding practices. The OWASP Top Ten helps organizations prioritize their efforts to secure their web applications effectively and reduce the likelihood of vulnerabilities that could lead to breaches or other security incidents.

What are the OWASP top 10?

- A01:2021-Broken Access Control moves up from the fifth position; 94% of applications were tested for some form of broken access control. The 34 Common Weakness Enumerations (CWEs) mapped to Broken Access Control had more occurrences in applications than any other category.
- A02:2021-Cryptographic Failures shifts up one position to #2, previously known as Sensitive Data Exposure, which was broad symptom rather than a root cause. The renewed focus here is on failures related to cryptography which often leads to sensitive data exposure or system compromise.
- A03:2021-Injection slides down to the third position. 94% of the applications were tested for some form of injection, and the 33 CWEs mapped into this category have the second most occurrences in applications. Cross-site Scripting is now part of this category in this edition.
- A04:2021-Insecure Design is a new category for 2021, with a focus on risks related to design flaws. If we genuinely want to "move left" as an industry, it calls for more use of threat modeling, secure design patterns and principles, and reference architectures.
- A05:2021-Security Misconfiguration moves up from #6 in the previous edition; 90% of applications were tested for some form of misconfiguration. With more shifts into highly configurable software, it's not surprising to see this category move up. The former category for XML External Entities (XXE) is now part of this category.

- A06:2021-Vulnerable and Outdated Components was previously titled Using Components with Known Vulnerabilities and is #2 in the Top 10 community survey, but also had enough data to make the Top 10 via data analysis. This category moves up from #9 in 2017 and is a known issue that we struggle to test and assess risk. It is the only category not to have any Common Vulnerability and Exposures (CVEs) mapped to the included CWEs, so a default exploit and impact weights of 5.0 are factored into their scores.
- A07:2021-Identification and Authentication Failures was previously Broken Authentication and is sliding down from the second position, and now includes CWEs that are more related to identification failures. This category is still an integral part of the Top 10, but the increased availability of standardized frameworks seems to be helping.
- A08:2021-Software and Data Integrity Failures is a new category for 2021, focusing on making assumptions related to software updates, critical data, and CI/CD pipelines without verifying integrity. One of the highest weighted impacts from Common Vulnerability and Exposures/Common Vulnerability Scoring System (CVE/CVSS) data mapped to the 10 CWEs in this category. Insecure Deserialization from 2017 is now a part of this larger category.
- A09:2021-Security Logging and Monitoring Failures was previously Insufficient Logging & Monitoring and is added from the industry survey (#3), moving up from #10 previously. This category is expanded to include more types of failures, is challenging to test for, and isn't well represented in the CVE/CVSS data. However, failures in this category can directly impact visibility, incident alerting, and forensics.
- A10:2021-Server-Side Request Forgery is added from the Top 10 community survey (#1). The data shows a relatively low incidence rate with above average testing coverage, along with above-average ratings for Exploit and Impact potential. This category represents the scenario where the security community members are telling us this is important, even though it's not illustrated in the data at this time.

What is CWE?

CWE stands for "Common Weakness Enumeration." It is a community-developed list of software and hardware weaknesses and vulnerabilities. CWE is maintained by the MITRE Corporation and serves as a standardized way to describe and categorize security weaknesses in software and systems.

CWE provides a comprehensive and well-organized taxonomy of common weaknesses that can exist in software code, design, architecture, and other aspects of technology systems. Each CWE entry includes a unique identifier, a description of the weakness, potential consequences, examples, mitigations, and references to related resources.

By using CWE, developers, security professionals, and organizations can better understand the types of vulnerabilities that can exist in their software and systems. It aids in improving secure coding practices, vulnerability assessments, and overall software security. It's often used in conjunction with other security resources like the OWASP Top Ten to help address and mitigate known software vulnerabilities.

1. Broken Access Control

OWASP CATEGORY: A01 2021 Broken Access Control

CWE ENTRIES ASSOCIATED:

- CWE-284: Improper Access Control (Authorization)
- CWE-285: Improper Authorization
- CWE-862: Missing Authorization

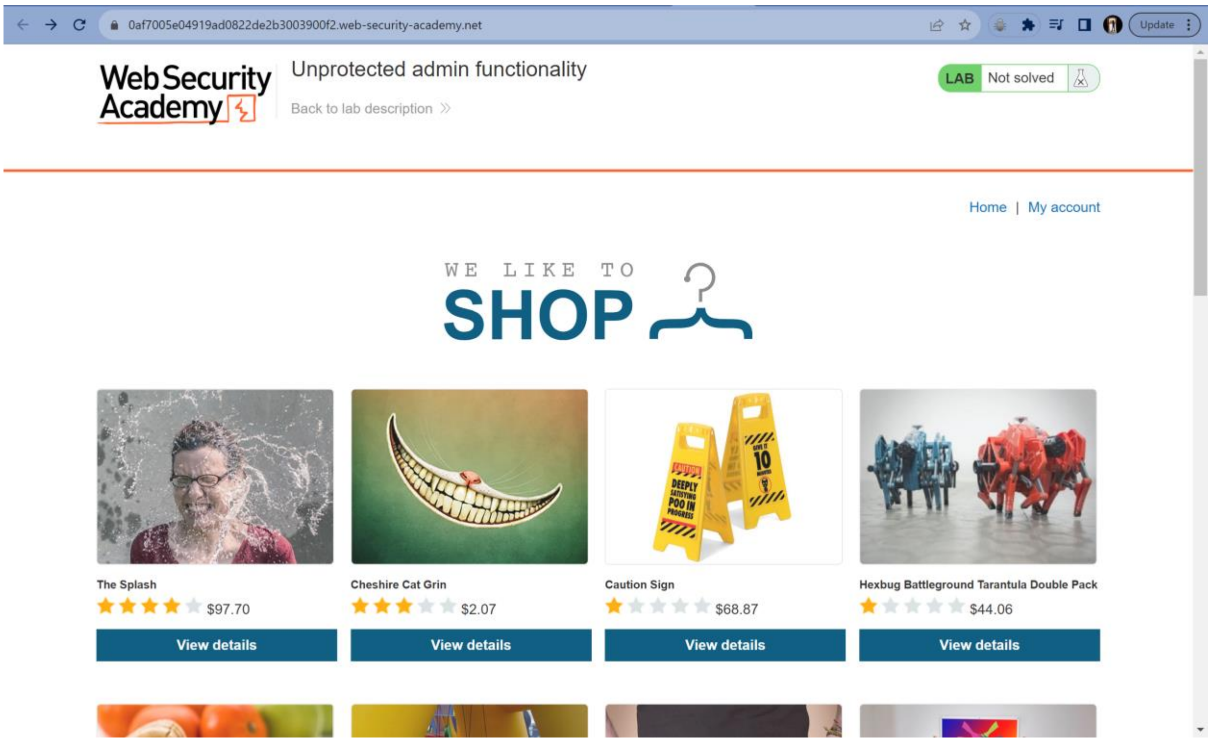
CWE: CWE 285 - Improper Authorization

DESCRIPTION: The product does not perform or incorrectly performs an authorization check when an actor attempts to access a resource or perform an action.

BUSINESS IMPACT: Authorization is a critical safeguard in maintaining the security and integrity of a system. When the product lacks proper or accurate authorization checks, it permits actors to access resources and perform actions without appropriate privileges. This vulnerability can result in unauthorized disclosure of sensitive information, unauthorized modifications to data, and potentially catastrophic breaches of system functionality. Malicious actors can exploit this weakness to manipulate, steal, or tamper with data, leading to legal consequences, financial losses, and damage to an organization's reputation. Furthermore, regulatory compliance may be compromised, and the organization's credibility in safeguarding user data can be undermined. Failing to implement robust authorization checks can expose an organization to a wide range of threats, making it imperative to address this vulnerability promptly to prevent unauthorized access and its subsequent harmful effects.

EXPLORATION OF VULNERABILITY:

The lab features an exposed admin panel that requires attention. The task involves addressing this issue by effectively removing the user named "Carlos."


0af7005e04919ad0822de2b3003900f2.web-security-academy.net

Unprotected admin functionality

Back to lab description

LAB Not solved

Home | My account

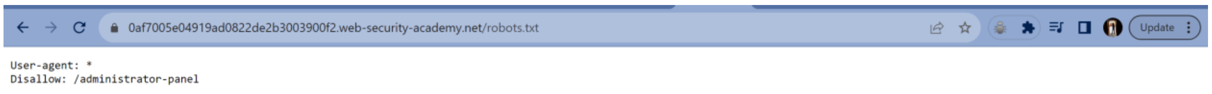
WE LIKE TO SHOP

The Splash $97.70 View details

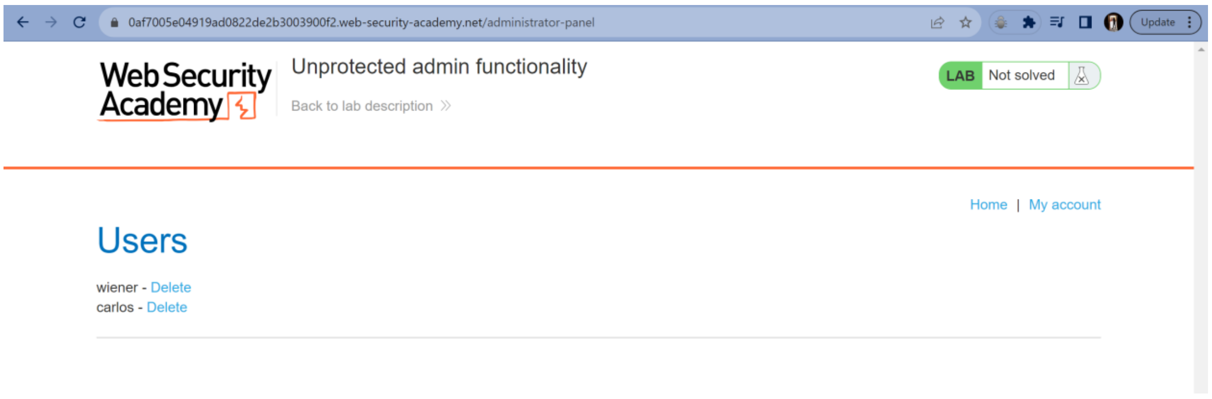
Cheshire Cat Grin $2.07 View details

Caution Sign $68.87 View details

Hexbug Battleground Tarantula Double Pack $44.06 View details


0af7005e04919ad0822de2b3003900f2.web-security-academy.net/robots.txt

```
User-agent: *
Disallow: /administrator-panel
```


0af7005e04919ad0822de2b3003900f2.web-security-academy.net/administrator-panel

Unprotected admin functionality

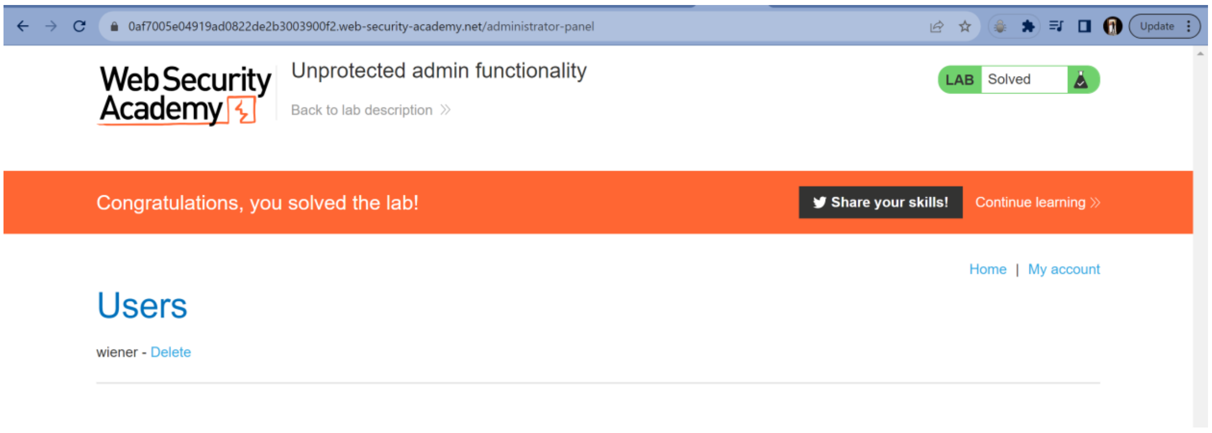
Back to lab description

LAB Not solved

Home | My account

Users

wiener - Delete
carlos - Delete


0af7005e04919ad0822de2b3003900f2.web-security-academy.net/administrator-panel

Unprotected admin functionality

Back to lab description

LAB Solved

Congratulations, you solved the lab!

Share your skills! Continue learning

Home | My account

Users

wiener - Delete

In the "Broken Access Control" lab, the primary vulnerability revolves around inadequate access control mechanisms, allowing unauthorized users to perform actions they shouldn't be able to. Specifically, this lab scenario involves an unprotected admin panel, which grants users unauthorized access to perform administrative actions.

To solve the lab, the task is to delete the user "carlos." The implication here is that there's a flaw in the access control logic that enables users, even those without appropriate privileges, to access and manipulate the admin panel. By exploiting this vulnerability, we are able to navigate to the admin panel and delete the user "carlos" despite lacking the necessary authorization.

2. Cryptographic Failures

OWASP CATEGORY: A02 2021 Cryptographic Failures

CWE ENTRIES ASSOCIATED:

- CWE-310: Cryptographic Issues
- CWE-327: Use of a Broken or Risky Cryptographic Algorithm
- CWE-328: Reversible One-Way Hash

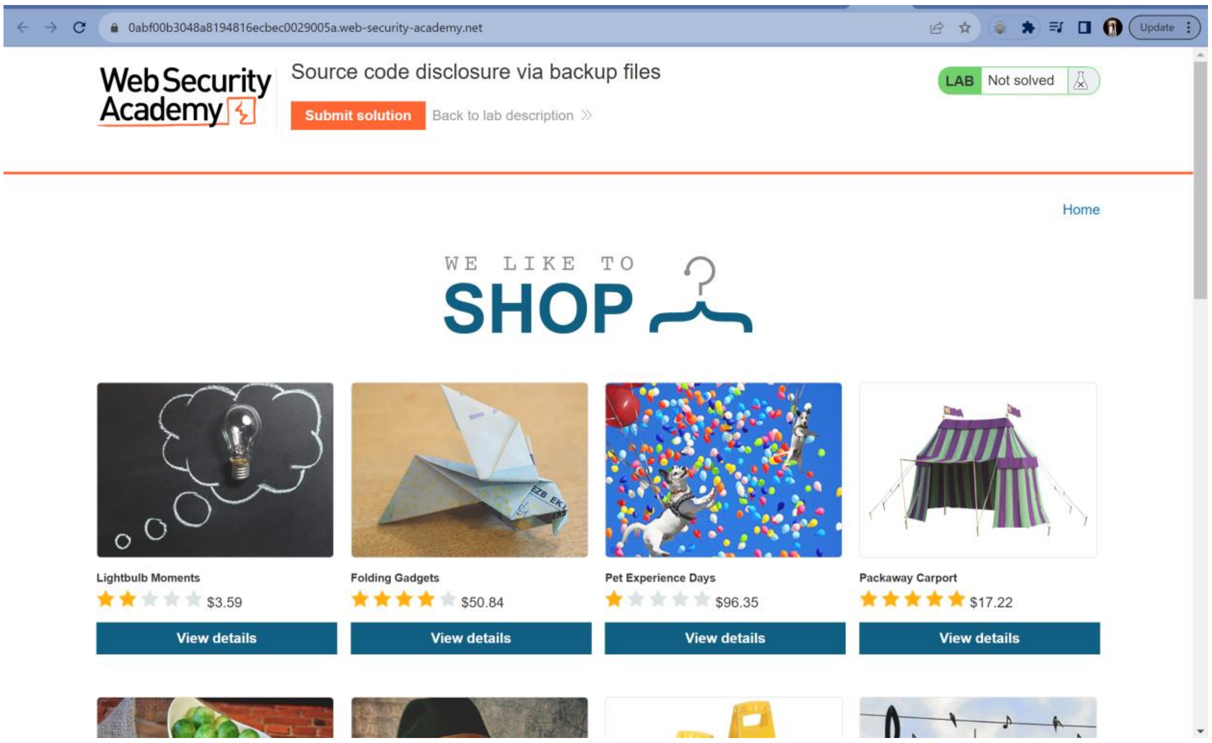
CWE: CWE 310 - Cryptographic Issues

DESCRIPTION: The product experiences vulnerabilities related to cryptographic operations, indicating that encryption and decryption mechanisms are not implemented correctly. This can lead to weak encryption, improper key management, or other cryptographic weaknesses that compromise the confidentiality and integrity of sensitive data.

BUSINESS IMPACT: Cryptography is a cornerstone of secure communication and data protection. When cryptographic failures occur, sensitive information becomes susceptible to unauthorized access, tampering, or decryption by malicious actors. Weak encryption algorithms or poorly managed cryptographic keys can undermine the very purpose of encryption, putting sensitive data at risk. Breaches in cryptographic mechanisms can lead to data leaks, financial loss, regulatory penalties, and damage to an organization's reputation. Addressing cryptographic failures is essential for maintaining the trust of users and ensuring that data remains confidential and secure throughout its lifecycle. Organizations must prioritize strong encryption practices and robust key management to mitigate the potential impact of cryptographic vulnerabilities.

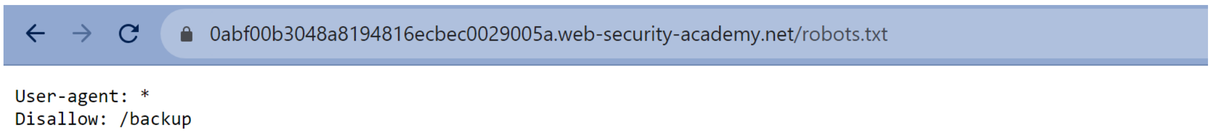
EXPLORATION OF VULNERABILITY:

The lab presents a situation where its source code becomes susceptible to exposure through backup files located within a concealed directory.



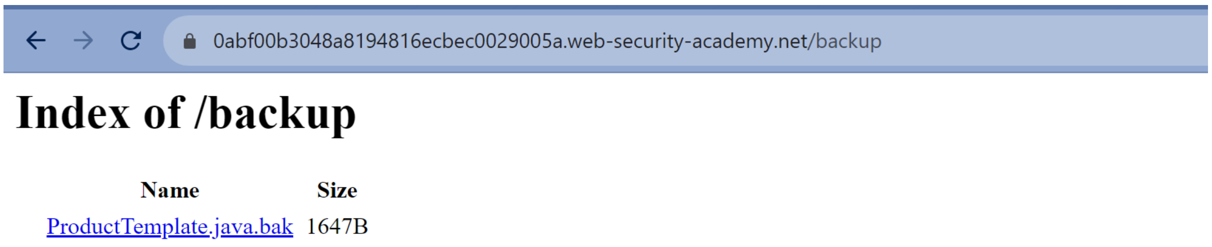
0abf00b3048a8194816ecbec0029005a.web-security-academy.net

Source code disclosure via backup files



0abf00b3048a8194816ecbec0029005a.web-security-academy.net/robots.txt

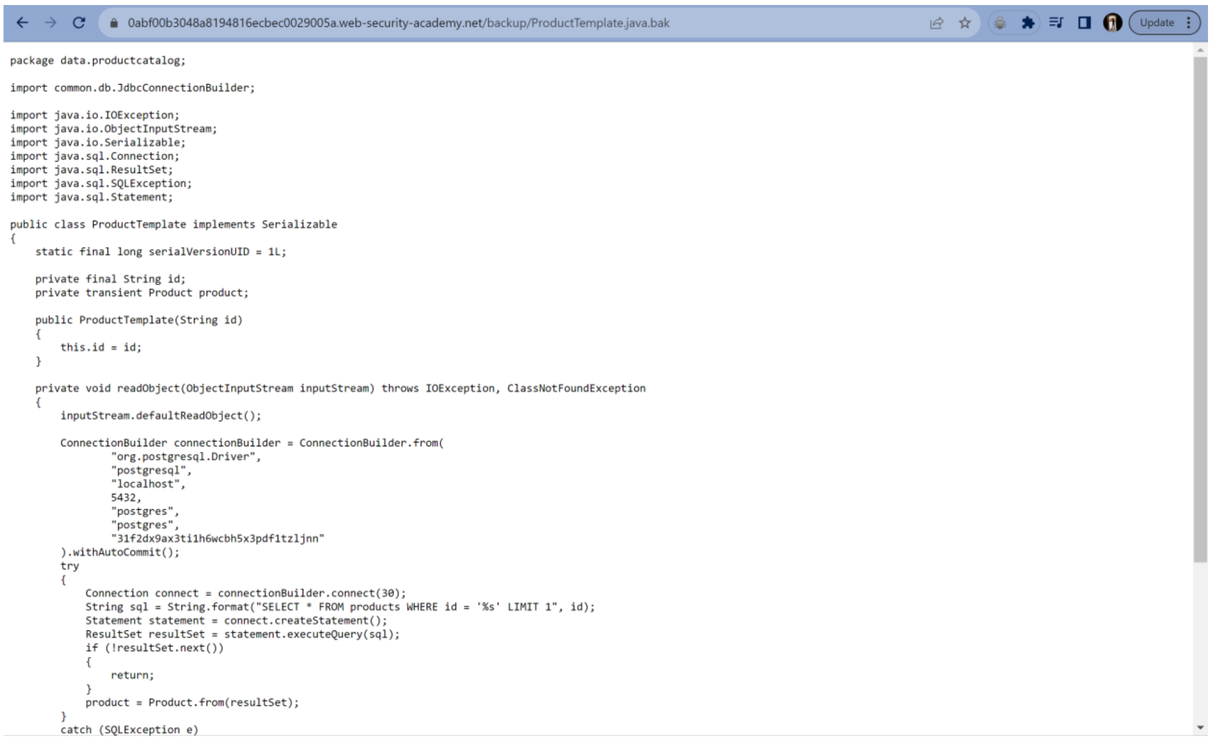
```
User-agent: *
Disallow: /backup
```



0abf00b3048a8194816ecbec0029005a.web-security-academy.net/backup

Index of /backup

Name	Size
ProductTemplate.java.bak	1647B



0abf00b3048a8194816ecbec0029005a.web-security-academy.net/backup/ProductTemplate.java.bak

```java
package data.productcatalog;

import common.db.JdbcConnectionBuilder;

import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.Serializable;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class ProductTemplate implements Serializable
{
    static final long serialVersionUID = 1L;

    private final String id;
    private transient Product product;

    public ProductTemplate(String id)
    {
        this.id = id;
    }

    private void readObject(ObjectInputStream inputStream) throws IOException, ClassNotFoundException
    {
        inputStream.defaultReadObject();

        ConnectionBuilder connectionBuilder = ConnectionBuilder.from(
                "org.postgresql.Driver",
                "postgresql",
                "localhost",
                5432,
                "postgres",
                "postgres",
                "31f2dx9ax3ti1h6wcbh5x3pdf1tzljnn"
        ).withAutoCommit();
        try
        {
            Connection connect = connectionBuilder.connect(30);
            String sql = String.format("SELECT * FROM products WHERE id = '%s' LIMIT 1", id);
            Statement statement = connect.createStatement();
            ResultSet resultSet = statement.executeQuery(sql);
            if (!resultSet.next())
            {
                return;
            }
            product = Product.from(resultSet);
        }
        catch (SQLException e)
```

The "Cryptographic Failures" lab focuses on vulnerabilities related to cryptographic operations. In this lab scenario, there is a vulnerability where the source code is unintentionally exposed through backup files located in a hidden directory. The objective is to exploit this vulnerability to discover a hard-coded database password that is present in the leaked source code.

Essentially, we find a way to access the source code, which contains a password that should have been kept secure. The exposure of this password through the leaked source code demonstrates the potential consequences of cryptographic vulnerabilities and poor code security practices.

3. Injection

OWASP CATEGORY: A03 2021 Injection

CWE ENTRIES ASSOCIATED:

- CWE-89: SQL 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 ('Server-Side Includes Injection')

- CWE-98: Improper Control of Filename for Include/Require Statement in PHP Program ('PHP File Inclusion')
- CWE-99: Improper Control of Resource Identifiers ('Resource Injection')

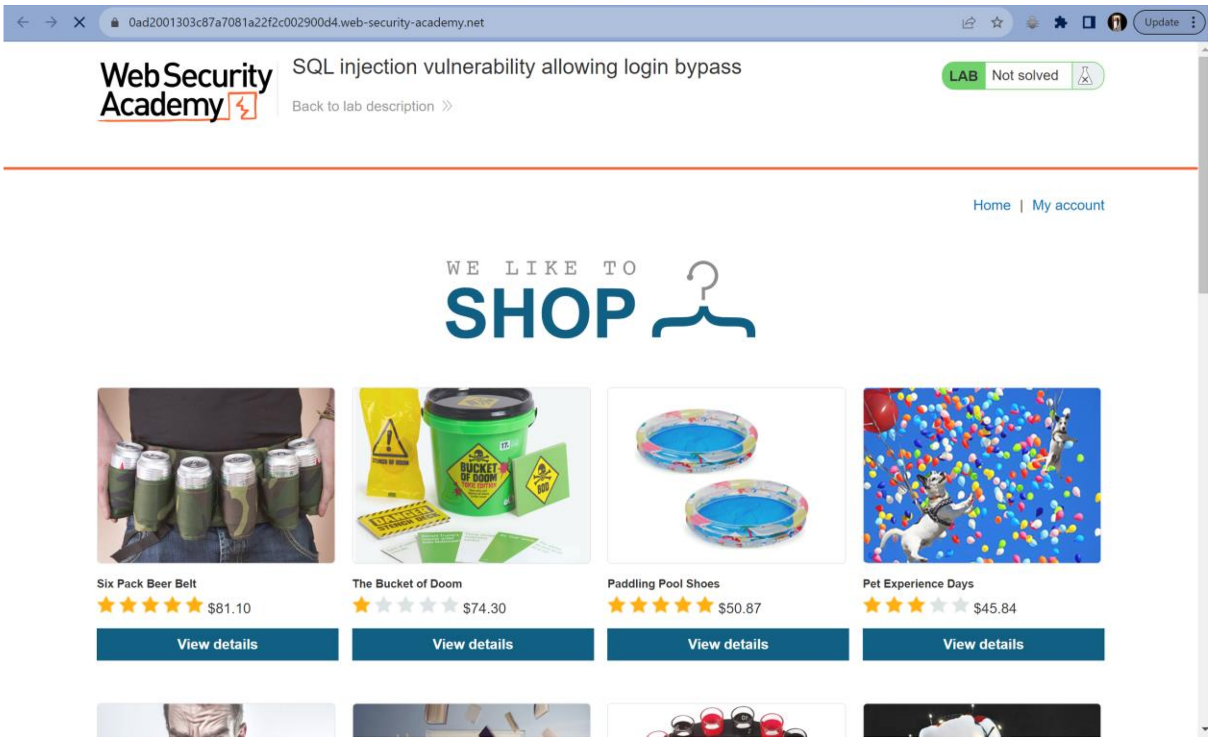
CWE: CWE 94 - Code Injection

DESCRIPTION: The product constructs all or part of a code segment using externally influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the syntax or behaviour of the intended code segment.

BUSINESS IMPACT: The failure to properly neutralize or sanitize externally influenced input in code construction can have severe consequences for an application or system. This vulnerability, often referred to as "code injection," can allow malicious actors to manipulate the intended behaviour of the code segment, leading to unauthorized access, data breaches, and even full system compromise. Attackers can exploit this weakness to inject malicious code, execute arbitrary commands, or modify the application's logic. As a result, sensitive data can be exposed, critical operations can be disrupted, and the confidentiality, integrity, and availability of the system can be compromised. Organizations that do not address this vulnerability are at risk of financial losses, regulatory non-compliance, reputational damage, and legal liabilities. It is crucial for software developers to implement proper input validation, output encoding, and secure coding practices to prevent code injection attacks and maintain the security posture of their applications.

EXPLORATION OF VULNERABILITY:

This lab features a SQL injection vulnerability in the product category filter. To solve it, execute a SQL injection attack to display unreleased products.


0ad2001303c87a7081a22f2c002900d4.web-security-academy.net

SQL injection vulnerability allowing login bypass

Back to lab description »

LAB Not solved

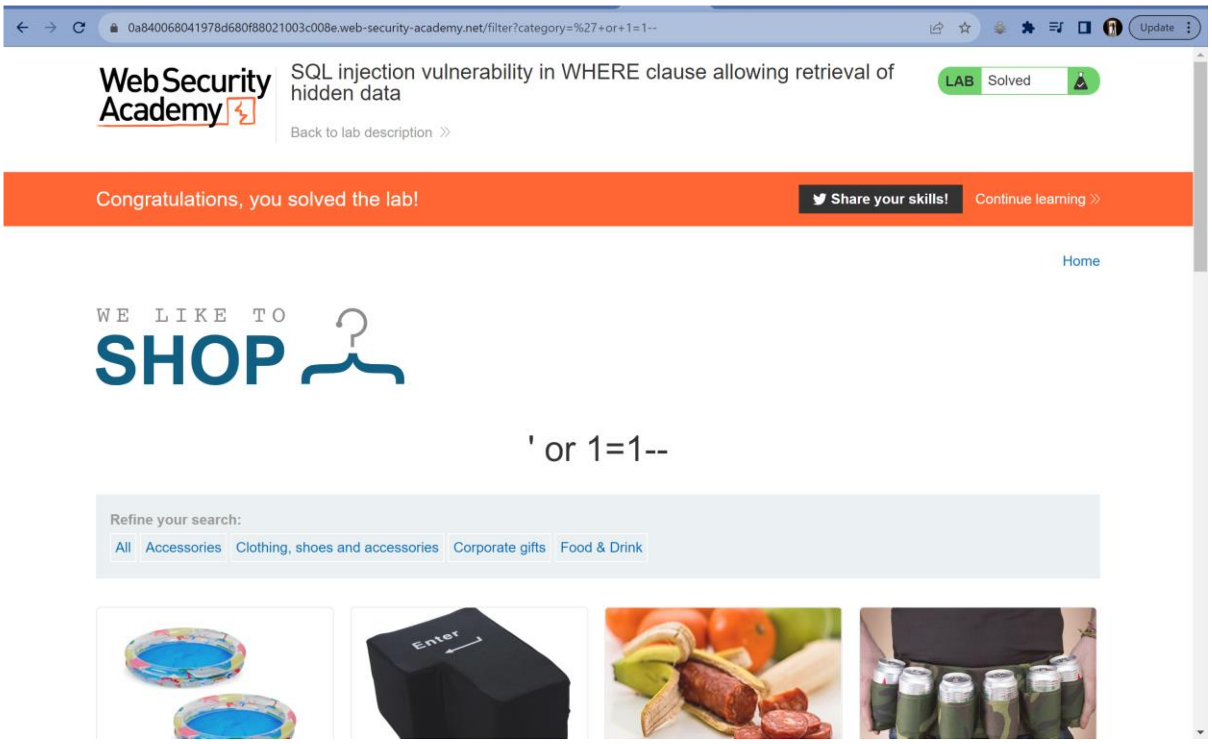
Home | My account

Six Pack Beer Belt $81.10

The Bucket of Doom $74.30

Paddling Pool Shoes $50.87

Pet Experience Days $45.84


0a840068041978d680f88021003c008e.web-security-academy.net/filter?category=%27+or+1=1--

SQL injection vulnerability in WHERE clause allowing retrieval of hidden data

Back to lab description »

LAB Solved

Congratulations, you solved the lab!

' or 1=1--

Refine your search:

All | Accessories | Clothing, shoes and accessories | Corporate gifts | Food & Drink

In the "Injection" lab, the focus is on exploiting vulnerabilities related to injection attacks. Specifically, the lab presents a SQL injection vulnerability within the product category filter of the application. The objective is to use this vulnerability to perform a SQL injection attack that tricks the application into revealing unreleased products.

SQL injection involves manipulating input data to insert malicious SQL code into a query. In this lab, the vulnerability lies in how the application handles user input within the product category filter. By carefully crafting an input that contains SQL code, participants can manipulate the application's

4. Insecure Design

OWASP CATEGORY: A04 2021 Insecure Design

CWE ENTRIES ASSOCIATED:

- CWE-311: Missing Encryption of Sensitive Data
- CWE-426: Untrusted Search Path
- CWE-613: Insufficient Session Expiration
- CWE 676 - Use of Potentially Dangerous Function

CWE: CWE 676 - Use of Potentially Dangerous Function

DESCRIPTION: The product invokes a potentially dangerous function that could introduce a vulnerability if it is used incorrectly, but the function can also be used safely.

BUSINESS IMPACT: The use of potentially dangerous functions within a product introduces a delicate balance between functionality and security. While these functions can offer valuable features, their misuse or incorrect implementation can lead to severe vulnerabilities. If not used correctly, these functions might inadvertently expose entry points for attackers to exploit, resulting in unauthorized data access, data manipulation, and even system compromise.

5. Security Misconfiguration

OWASP CATEGORY: A05 2021 Security Misconfiguration

CWE ENTRIES ASSOCIATED:

- CWE-319: Cleartext Transmission of Sensitive Information
- CWE-452: Missing Initialization
- CWE-697: Incorrect Comparison
- CWE 829 - Inclusion of Functionality from Untrusted Control Sphere

CWE: CWE 829 - Inclusion of Functionality from Untrusted Control Sphere

DESCRIPTION: The product imports, requires, or includes executable functionality (such as a library) from a source that is outside of the intended control sphere.

BUSINESS IMPACT: Incorporating external executable content without proper control poses a serious security risk. This vulnerability can lead to the introduction of malicious code, potentially causing unauthorized access, data breaches, service disruptions, financial losses, and reputational harm. Ensuring trusted sources and robust code review practices are crucial to prevent these risks and maintain system integrity.