**Relatório de Análise**

Projeto 1 – Vulnerabilidades em produtos de software

Segurança Informática e nas Organizações (SIO)

Departamento de Eletrónica, Telecomunicações e Informática

Ano Letivo 2023/2024

# **Índice**

[**Índice** 2](#_Toc150466914)

[**Introdução** 3](#_Toc150466915)

[**CWEs** 4](#_Toc150466916)

[**CWE-79** 4](#_Toc150466917)

[**CWE-89** 4](#_Toc150466918)

[**CWE-521** 5](#_Toc150466919)

[**CWE-256** 6](#_Toc150466920)

[**CWE-620** 7](#_Toc150466921)

[**CWE-434** 7](#_Toc150466922)

# **Introdução**

O nosso projeto retrata uma loja online vendedora de artigos de recordação do DETI na Universidade de Aveiro, abrangendo produtos como *t-shirts*, *hoodies* e sapatinhas. Dentro do projeto é possível criar utilizadores, *resetar* a sua palavra-passe caso o mesmo se tenho esquecido da anterior e dar *login* mantendo sessão iniciada, isto é, enquanto o servidor estiver a “funcionar” basta ser feito *login* uma vez para que o website reconheça o utilizador que o está a aceder e posteriormente, se necessário, a opção de terminar a sessão também está disponível. É possível fazer compras de diversos produtos adicionando-os ao *cart* (carrinho), caso o utilizador se arrependa pode removê-los mais tarde e proceder ao seu *checkout*. O *website* dispõe de uma página relativa a criar *posts* com título e conteúdo, em que os mesmos são descritos pelo utilizador e outra página onde tem opção de mudar de palavra-passe atual sendo a introdução da antiga palavra-passe obrigatória para o procedimento ser feito corretamente. Também dispõe de uma conta *admin* pré-definida, *username*: *admin* e *password*: *admin*. Esta é capaz de adicionar produtos novos e de ter acesso aos *users*, *e-mail* e *passwords* dos mesmos.

É de ter em conta que caso os utilizadores não tenham a sessão iniciada não conseguem aceder a páginas críticas bem como fazer compras.

O projeto possui duas pastas principais, a *app*, relativa à aplicação não segura (contém diversos tipos de vulnerabilidades) e a *app\_sec* que apresenta a aplicação segura (sem fraquezas).

# **CWEs**

## **CWE-79**

**O que é?**

O CWE-79, ou “*Cross-Site Scripting*” (*XSS*), é um tipo de vulnerabilidade encontrado em aplicações *web*, onde o atacante injeta códigos maliciosos (geralmente *JavaScript*) em um input de uma página *web*. A partir dele, consegue obter informações privadas dos utilizadores (como cookies e outros tipos de recursos) que carregarem essa página *web* com o *script* injetado.

**Exploração da vulnerabilidade**

Estes ataques ocorrem facilmente em páginas *web* que são guardadas pelo servidor, onde neste caso temos a página “Fórum”. Aqui os *posts* são obtidos através de chamadas *ajax* (*jquery*), e a aplicação não executa a limpeza adequada dos inputs. Isso significa que um atacante, ao explorar essa falha, pode facilmente inserir códigos *JavaScript* maliciosos nos *posts* do “Fórum” e que, uma vez executados pelos utilizadores quando carregam a página, comprometem a segurança do sistema e expõem os utilizadores.

Na página “Loja”, a vulnerabilidade permite que um atacante, previamente autenticado como admin, insira o código *JavaScript* ao adicionar um novo produto. Quando outro utilizador carrega essa página, o atacante pode explorar essa falha e obter dados do utilizador.

**Solução**

Para corrigir esta vulnerabilidade, é crucial validar e sanitizar os dados inseridos pelos utilizadores, garantindo que não contenham código malicioso. Assim usamos a função *render\_template()* para resolver o problema do XSS.

## **CWE-89**

**O que é?**

O CWE-89, ou “*SQLInjection*”, é um tipo de vulnerabilidade encontrado em aplicações *web* que possuem bases de dados, onde o atacante consegue injetar código malicioso (neste caso *SQL*) nos pedidos de *query* feitas à base de dados e, a partir daí, consiga receber informação restrita e/ou alterar dados dentro da base de dados.

**Exploração da vulnerabilidade**

Esta vulnerabilidade específica torna o *website* altamente vulnerável, sendo uma porta de entrada para diversos *exploit*, como o CWE-256 e o CWE-756.

Ao inserir **admin’--** no campo de *username* na página de *login*, o atacante consegue fazer *login* sem a necessidade de password. Essa técnica explora uma falha ao adicionar um comentário que interfere no processo de autenticação, anulando a necessidade de inserir a password associada ao utilizador “*admin*”.

**Solução**

Para resolver esta vulnerabilidade, no contexto de SQLInjection, ao utilizar o comando ‘execute’ e incorporar os argumentos diretamente na função, como abaixo apresentado:



O SQLite, por si mesmo, já incorpora mecanismos de prevenção contra injeções de SQL. Ao usar o parâmetro ‘(username,)’, o SQLite resolve automaticamente os inputs como dados a serem inseridos na query, eliminando assim a possibilidade de ataques. Esta abordagem evita a necessidade de realizar a sanitização manual dos inputs, proporcionando segurança adicional.

## **CWE-521**

**O que é?**

O CWE-521, ou “*Weak Password Requirements*”, refere-se a vulnerabilidades relacionadas à segurança de senhas em sistemas e aplicações. Isso acontece quando as políticas de senha são inadequadas, permitindo senhas fracas e suscetíveis a ataques. Para evitar isso, é crucial implementar políticas de senha mais fortes/robustas, exigindo combinações complexas de caracteres.

**Exploração da vulnerabilidade**

Esta vulnerabilidade aparece no momento na criação de conta de um utilizador, onde estes não precisam de requisitos para a criação da sua palavra-passe, isto leva com que a maioria dos utilizadores crie palavras-passes inseguras, ou seja, pequenas, sem maiúsculas e sem caracteres especiais. Assim este problema pode levar a atacantes conseguirem invadir facilmente a conta do utilizador e obter informações sobre ele.

**Solução**

Para este problema fizemos uma função de validação, onde verificamos se a palavra-passe do utilizador possui os requisitos necessários, isto é, tamanho, números e maiúsculas.

## **CWE-256**

**O que é?**

O CWE-256, ou “*Plaintext Storage of a Password*” é um tipo de vulnerabilidade que consiste no armazenamento de palavras-passe de utilizadores sem qualquer forma de criptografia, fazendo com que haja obtenção direta das palavras-passe aos atacantes se estes conseguirem aceder à base de dados.

**Exploração da vulnerabilidade**

Esta vulnerabilidade, combinada com *SQLInjection*, é bastante interessante, é possível utilizar o CWE-89 para visualizar em texto simples a palavra-passe de todos os utilizadores. Se esta não estiver cifrada, o atacante consegue ver a palavra-passe, enquanto ao usar uma cifra (por exemplo, sha256), o atacante apenas consegue visualizar uma combinação aleatória de caracteres.

Esta vulnerabilidade ocorre onde todos os campos onde a *SQLInjection* é possível. No caso da aplicação não segura, afeta qualquer input que tenha acesso à base de dados. Também ocorre sempre que alguém com acesso à base de dados tenta visualizar as palavras-passe. Isso acontece porque elas não devem ser visíveis para qualquer pessoa, mesmo que seja o proprietário do *website* ou um administrador do sistema.

**Solução**

Para a solução desta vulnerabilidade tivemos de cifrar as *passwords* das contas dos utilizadores no momento do registo da conta, assim, a *password* inserida na base de dados já está cifrada, e por isso, na eventualidade de algum atacante ter acesso à base de dados, este não conseguirá saber as *passwords* dos utilizadores.



Uma imagem com texto, Tipo de letra, captura de ecrã, file

Descrição gerada automaticamente

## **CWE-620**

**O que é?**

O CWE-620, ou “*Unverified Password Change*” é uma vulnerabilidade onde o sistema permite a alteração da senha sem verificar adequadamente a identidade do usuário (palavra-passe antiga). Caso o atacante consiga aceder à conta do utilizador, este irá conseguir negar o acesso à mesma sem muito esforço.

**Exploração da vulnerabilidade**

Esta vulnerabilidade permite a alteração de *password* de uma conta de utilizador sem a necessidade de inserir a *password* atual. Em essência, estamos perante uma falha de design na página destinada à modificação de *passwords*.

Esta vulnerabilidade ocorre na página de *login* e na página de alteração da *password* no perfil do utilizador. Neste contexto, o utilizador poderá “esquecer-se da palavra-passe” e ao pressionar o botão de recuperação é direcionado para um formulário, possibilitando assim a alteração da sua *password* atual sem a autenticação da *password* existente, são apenas necessários o seu nome de utilizador e o seu *e-mail*. Relativamente à alteração da *password* no perfil do utilizador o mesmo poderá alterá-la sem a necessidade da palavra-passe atual.

**Solução**

Para a resolução desse problema, retiramos o input de *“old password”*. Assim o utilizador poderá utilizar apenas o nome de utilizador e o e-mail para repor a palavra-passe.

## **CWE-434**

**O que é?**

O CWE-434, ou “*Unrestricted Upload of File with Dangerous Type*”, é um tipo de vulnerabilidade de uma aplicação *web* que permite com que o atacante insira/envie arquivos maliciosos para o servidor e, a partir deles, consigam obter informações restritas ou modificar certas funções do servidor, provocando uma alteração do comportamento dele.

**Exploração da vulnerabilidade**

Esta vulnerabilidade pode ser encontrada no catálogo da loja, onde o administrador, ou uma outra entidade que consiga obter esse papel, pode inserir um ficheiro/arquivo com código malicioso fazendo com que o atacante consiga através dele obter informações sobre o servidor ou modificá-lo.

**Solução**

Para impedir com que ocorra este problema, adicionamos uma constante de extinções permitidas (*“ALLOWED\_EXTENSIONS”*) que possuem todos os tipos de formato de ficheiros aceites para *upload*. Também adicionamos uma função *“allowed\_file”* que verifica se um ficheiro faz parte dos tipos de ficheiros aceite.



