



universidade de aveiro

VULNERABILIDADES

Segurança Informática e nas Organizações

2022/2023

Filipe Antão, 103470
Nuno Sousa, 103664
Pedro Matos 102993,
Simão Antunes, 104092

Índice

Índice	2
Introdução	3
Funcionalidades	4
Login/Signup	4
Make appointments	5
Get exams/ Make exams	5
Get/Write Messages	5
O utilizador com role User pode escrever mensagens a um Doctor e vice-versa. Estes têm acesso às mensagens recebidas na aba Messages da NavBar.	5
Tecnologias	5
Vulnerabilidades exploradas	6
CWE-79: Improper Neutralization of Input During Web Page Generation ("Cross-site Scripting")	7
CWE-89: Improper Neutralization of Special Elements used in an SQL Command ("SQL Injection")	8
CWE-522: Insufficiently Protected Credentials	9
CWE-200: Expose of Sensitive Information to an Unauthorized Actor	11
CWE-311: Missing Encryption of Sensitive Data	11
CWE-521: Weak Password Requirements	12
Vulnerability Score	13
Conclusão	15

Introdução

O primeiro projeto de Segurança Informática e nas Organizações tem como objetivo a exploração e implementação de vulnerabilidades.

De forma a corresponder ao pedido, foram desenvolvidas duas versões de um *website* para uma clínica de saúde (eHealth Corp). A primeira versão consiste num *website* inseguro com algumas vulnerabilidades presentes e a segunda versão é um *website* semelhante mas com as vulnerabilidades corrigidas.

O desenvolvimento deste projeto envolveu o estudo das vulnerabilidades lecionadas (Cross-site Scripting e SQL Injection) mas também de outros tipos de vulnerabilidades que também foram aplicadas.

Funcionalidades

Ao entrar no site, o utilizador terá que criar uma conta ou iniciar sessão numa conta já existente.

De seguida, está disponível um leque de funcionalidades tais como marcação e visualização de consultas, envio de mensagens a um médico e consulta de exames.

Login/Signup

Para poder utilizar o nosso website, o utilizador vai ter que ou entrar com a sua conta já criada ou fazer o registo no nosso sistema. Os processos de login e sign up são simples e intuitivos. Ao registar tem que criar um email, username, password e escolher uma role: User ou Doctor. Consoante esta role o utilizador terá acesso a diferentes funcionalidades.

Make appointments

O utilizador com a role User tem acesso à marcação de consultas, que pode marcar selecionando um serviço, uma data e uma descrição. Pode também consultar as consultas que tem marcadas.

Get exams/ Make exams

O utilizador com a role Doctor pode fazer exames para determinados utilizadores User. Os utilizadores com role User por sua vez têm acesso a esses exames através de um código.

Get/Write Messages

O utilizador com role User pode escrever mensagens a um Doctor e vice-versa. Estes têm acesso às mensagens recebidas na aba Messages da NavBar.

Tecnologias

As tecnologias utilizadas neste projeto foram:

- HTML e CSS (front-end);
- Módulo Flask;
- SQLAlchemy;

Vulnerabilidades exploradas

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

O que é?

O software não neutraliza o input do utilizador ou fá-lo incorretamente antes de este ser colocado no html que é renderizado na página web por outros utilizadores.

Implementação

Esta vulnerabilidade encontra-se representada nas páginas em que aparece informação que foi anteriormente introduzida pelo utilizador tais como as páginas *appointments* e *messages*. Em ambas as páginas existe um campo de texto em que o utilizador introduz uma mensagem ou descrição. Caso este input não seja neutralizado podem acontecer casos como o que é apresentado de seguida.

Exemplo:

Contact a Doctor

doctor@gmail.com

`<script>alert("vulnerabilidade")</script>`

Enviar

127.0.0.1:5000 diz
vulnerabilidade

OK

Make Appointment

Cardiology 2022-09-13

`<script>alert("hackeado")</scrip`

Make Appointment

127.0.0.1:5000 says
hackeado

OK

Solução

Verificar o conteúdo e garantir que quando é passado para o html este não tem ação na página. Aproveitamos a função `html.escape()`.

No ficheiro `app_sec/website/views.py`:

```
@views.route('/appointment', methods=['GET', 'POST'])
def appointment():
    if request.method == "POST":
        service = request.form.get("service")
        date = request.form.get("date")
        description = html.escape(request.form.get("description"))

        appointment = Appointment(service=service, date=date, description=description, user_id=current_user.id)
        db.session.add(appointment)
        db.session.commit()
        flash("Appointment scheduled")

    return render_template("appointment.html")
```

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

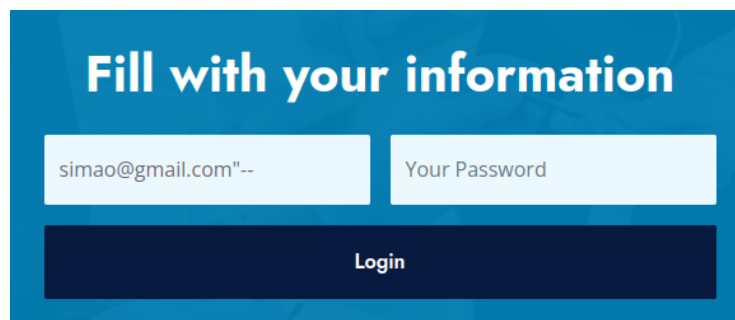
O que é?

O software constrói um comando SQL ou parte dele usando input do utilizador mas não neutraliza, ou fá-lo incorretamente, elementos especiais que podem modificar o comando SQL quando este é enviado.

Implementação

Esta vulnerabilidade encontra-se representada na página de login, onde se pode iniciar sessão sem a palavra-passe, ou seja, sabendo apenas o e-mail do utilizador.

Exemplo: simao@gmail.com"--



Solução

Para resolver e prevenir esta vulnerabilidade, implementámos dois métodos. O primeiro é a definição do tipo input como um email fazendo com que não sejam permitidas plicas (') ou aspas (") depois do arroba (@). O segundo é a enumeração dos caracteres que podem fazer parte do email e caso essa lista de caracteres não seja respeitada, o login não é efetuado.

```
<input type="email"
```

```
AllowList_email = "[a-zA-Z0-9-_.]+@[a-zA-Z0-9]+\.[a-z]{1,3}$"
if re.match(AllowList_email,email):
    pass
else:
    return render template("login.html", user=current user)
```


CWE-522: Insufficiently Protected Credentials

O que é?

Guardar uma password de forma desprotegida de forma a que qualquer atacante consiga obter esta informação, resultando numa falha de segurança.

Implementação

Esta vulnerabilidade encontra-se representada na função de login do ficheiro auth.py em que a password do utilizador é encriptada com um método (MD5) que é breakable, ou seja, não garante a segurança.

```
password = request.form.get('password')
digest = hashes.Hash(hashes.MD5())
digest.update(password.encode('UTF-8'))
hashpass=digest.finalize()
```

Solução: utilização de outro método de encriptação (SHA256)

```
password = request.form.get('password')

digest = hashes.Hash(hashes.SHA256())
digest.update(password.encode('UTF-8'))
hashpass=digest.finalize()
```

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

O que é?

O website expõe informação sensível a um utilizador que não tem permissão para aceder a essa informação comprometendo assim a privacidade dos utilizadores e a segurança.

Implementação

Esta vulnerabilidade encontra-se representada na página *exams*. O utilizador deveria apenas conseguir ver os seus exames mas consegue ver os exames de outros utilizadores.

Solução:

```
if exam != None:
    if exam.user_id == current_user.id:
        return render_template("exams.html", user=current_user, exam=exam)
    else:
        flash("ERRO!")
```

CWE-311: Missing Encryption of Sensitive Data

O que é?

Informação importante não é encriptada antes do seu armazenamento ou transmissão.

Implementação

Esta vulnerabilidade encontra-se representada nas mensagens enviadas pelos utilizadores aos Doctors. Este tipo de informação sensível devia ser encriptada.

Solução

```
pub_key = key.key

key = serialization.load_pem_public_key(
    pub_key,
    backend=default_backend()
)

encrypted_message = rsa_encrypt(message, key)
```

```
decrypt.py
encrypt.py
generatekeys.py
```

CWE-521: Weak Password Requirements

O que é?

O site não obriga os utilizadores a ter passwords seguras, o que faz com que as suas contas sejam comprometidas mais facilmente.

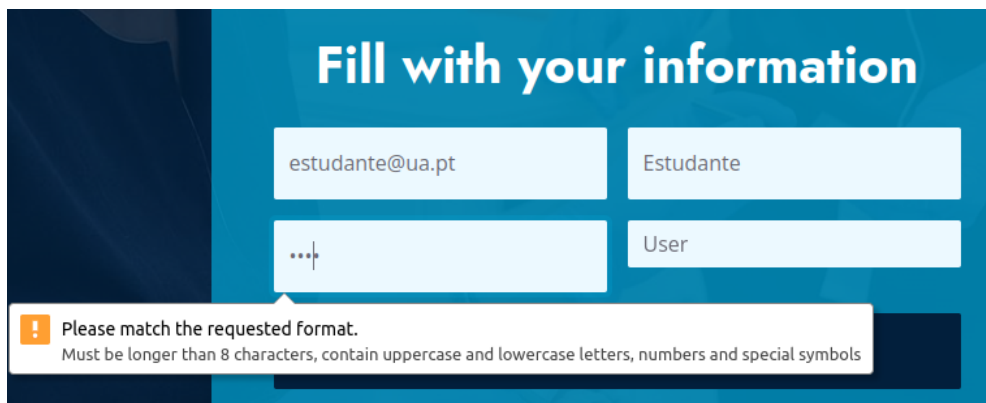
Implementação

Esta vulnerabilidade acontece quando o utilizador cria uma nova conta.

Solução

Implementámos os seguintes requerimentos na criação da password:

- Tem de ter números, caracteres especiais, minúsculas e maiúsculas e pelo menos 8 caracteres no total.



The screenshot shows a registration form with the title "Fill with your information". It contains four input fields: an email field with "estudante@ua.pt", a first name field with "Estudante", a password field with masked characters "...", and a last name field with "User". A red error message box is displayed at the bottom left, stating: "Please match the requested format. Must be longer than 8 characters, contain uppercase and lowercase letters, numbers and special symbols".

Vulnerability Score

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

Base Score		7.8 (High)
Attack Vector (AV)	Scope (S)	
<input type="button" value="Network (N)"/> <input type="button" value="Adjacent (A)"/> <input checked="" type="button" value="Local (L)"/> <input type="button" value="Physical (P)"/>	<input checked="" type="button" value="Unchanged (U)"/> <input type="button" value="Changed (C)"/>	
Attack Complexity (AC)	Confidentiality (C)	
<input checked="" type="button" value="Low (L)"/> <input type="button" value="High (H)"/>	<input type="button" value="None (N)"/> <input type="button" value="Low (L)"/> <input checked="" type="button" value="High (H)"/>	
Privileges Required (PR)	Integrity (I)	
<input checked="" type="button" value="None (N)"/> <input type="button" value="Low (L)"/> <input type="button" value="High (H)"/>	<input type="button" value="None (N)"/> <input type="button" value="Low (L)"/> <input checked="" type="button" value="High (H)"/>	
User Interaction (UI)	Availability (A)	
<input type="button" value="None (N)"/> <input checked="" type="button" value="Required (R)"/>	<input type="button" value="None (N)"/> <input type="button" value="Low (L)"/> <input checked="" type="button" value="High (H)"/>	

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

Base Score		8.4 (High)
Attack Vector (AV)	Scope (S)	
<input type="button" value="Network (N)"/> <input type="button" value="Adjacent (A)"/> <input checked="" type="button" value="Local (L)"/> <input type="button" value="Physical (P)"/>	<input checked="" type="button" value="Unchanged (U)"/> <input type="button" value="Changed (C)"/>	
Attack Complexity (AC)	Confidentiality (C)	
<input checked="" type="button" value="Low (L)"/> <input type="button" value="High (H)"/>	<input type="button" value="None (N)"/> <input type="button" value="Low (L)"/> <input checked="" type="button" value="High (H)"/>	
Privileges Required (PR)	Integrity (I)	
<input checked="" type="button" value="None (N)"/> <input type="button" value="Low (L)"/> <input type="button" value="High (H)"/>	<input type="button" value="None (N)"/> <input type="button" value="Low (L)"/> <input checked="" type="button" value="High (H)"/>	
User Interaction (UI)	Availability (A)	
<input checked="" type="button" value="None (N)"/> <input type="button" value="Required (R)"/>	<input type="button" value="None (N)"/> <input type="button" value="Low (L)"/> <input checked="" type="button" value="High (H)"/>	

CWE-522: Insufficiently Protected Credentials

Base Score		8.4 (High)
Attack Vector (AV)	Scope (S)	
<input type="button" value="Network (N)"/> <input type="button" value="Adjacent (A)"/> <input checked="" type="button" value="Local (L)"/> <input type="button" value="Physical (P)"/>	<input checked="" type="button" value="Unchanged (U)"/> <input type="button" value="Changed (C)"/>	
Attack Complexity (AC)	Confidentiality (C)	
<input checked="" type="button" value="Low (L)"/> <input type="button" value="High (H)"/>	<input type="button" value="None (N)"/> <input type="button" value="Low (L)"/> <input checked="" type="button" value="High (H)"/>	
Privileges Required (PR)	Integrity (I)	
<input checked="" type="button" value="None (N)"/> <input type="button" value="Low (L)"/> <input type="button" value="High (H)"/>	<input type="button" value="None (N)"/> <input type="button" value="Low (L)"/> <input checked="" type="button" value="High (H)"/>	
User Interaction (UI)	Availability (A)	
<input checked="" type="button" value="None (N)"/> <input type="button" value="Required (R)"/>	<input type="button" value="None (N)"/> <input type="button" value="Low (L)"/> <input checked="" type="button" value="High (H)"/>	

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

Base Score

4.0
(Medium)

Attack Vector (AV)

Network (N)Adjacent (A)**Local (L)**Physical (P)

Attack Complexity (AC)

Low (L)High (H)

Privileges Required (PR)

None (N)Low (L)High (H)

User Interaction (UI)

None (N)Required (R)

Scope (S)

Unchanged (U)Changed (C)

Confidentiality (C)

None (N)**Low (L)**High (H)

Integrity (I)

None (N)Low (L)High (H)

Availability (A)

None (N)Low (L)High (H)

CWE-311: Missing Encryption of Sensitive Data

Base Score

6.4
(Medium)

Attack Vector (AV)

Network (N)Adjacent (A)Local (L)**Physical (P)**

Attack Complexity (AC)

Low (L)**High (H)**

Privileges Required (PR)

None (N)Low (L)High (H)

User Interaction (UI)

None (N)Required (R)

Scope (S)

Unchanged (U)Changed (C)

Confidentiality (C)

None (N)Low (L)**High (H)**

Integrity (I)

None (N)Low (L)**High (H)**

Availability (A)

None (N)Low (L)**High (H)**

CWE-521: Weak Password Requirements

Base Score

7.4
(High)

Attack Vector (AV)

Network (N)Adjacent (A)**Local (L)**Physical (P)

Attack Complexity (AC)

Low (L)**High (H)**

Privileges Required (PR)

None (N)Low (L)High (H)

User Interaction (UI)

None (N)Required (R)

Scope (S)

Unchanged (U)Changed (C)

Confidentiality (C)

None (N)Low (L)**High (H)**

Integrity (I)

None (N)Low (L)**High (H)**

Availability (A)

None (N)Low (L)**High (H)**

No total consideramos que o nosso site inseguro tem um vulnerability score de 42,4.

Conclusão

Ao longo do desenvolvimento deste projeto tivemos a oportunidade de consolidar os nossos conhecimentos sobre vulnerabilidades e como as evitar em futuros projetos. Revimos o nosso conhecimento em HTML e CSS e exploramos módulos de python tais como o Flask e o SQLAlchemy. Tendo isto em conta, consideramos ter atingido o objetivo do projeto.