

# PROJETO – Vulnerabilidades

Departamento de Eletrónica, Telecomunicações e  
Informática Universidade de Aveiro

## Segurança Informática e nas Organizações

Equipa 13 – Frederico Vieira (98518), Ricardo Antunes (98275),  
Tiago Coelho (98385), Vladyslav Mysnyk (97548)

2021/2022



# Introdução

Este projeto consiste na criação de uma app que de forma propositada terá algumas fraquezas na sua segurança para que nela sejam testadas algumas vulnerabilidades que foram aprendidas na Unidade Curricular de SIO.

Para tal, foi criado um fórum online no qual é possível fazer o registo e consequentemente fazer o login, a fim de se poder criar publicações que serão visíveis para todos os utilizadores do fórum.

Para testar as vulnerabilidades, foram feitas duas apps idênticas. No entanto, as duas diferem-se por uma delas ser segura e a outra insegura. Ou seja, na primeira, não é possível testar as vulnerabilidades que queremos, pois tem um nível de segurança maior que não permite sofrer com esses ataques. Já a segunda tem várias fraquezas a nível de segurança e por isso é possível testar várias vulnerabilidades associadas a CWE-79 e CWE-89, por exemplo.

## **Vulnerabilidades**

- **CWE-20**
- **CWE-79**
- **CWE-89**
- **CWE-200**
- **CWE-287**
- **CWE-425**
- **CWE-521**

## Funcionamento

Como foi dito anteriormente, esta aplicação consiste num fórum onde é possível os utilizadores criarem publicações e também responderem a elas, interagindo assim com os outros utilizadores.

Para começar, é necessário criar uma conta no fórum, entrando na página de registo e preenchendo os campos necessários para o concluir (email, nome completo, password).

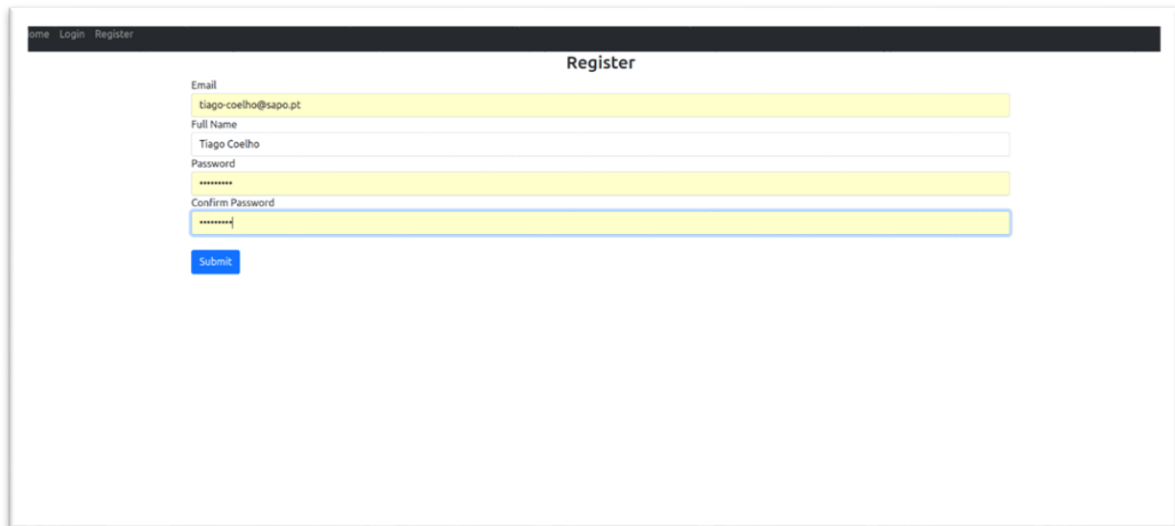
A screenshot of a web browser showing the 'Register' page. The page has a dark header with links 'Home', 'Login', and 'Register'. The title 'Register' is centered. Below it, there are four input fields: 'Email' (containing 'tiago-coelho@sapo.pt'), 'Full Name' (containing 'Tiago Coelho'), 'Password' (with masked characters), and 'Confirm Password' (also with masked characters). A blue 'Submit' button is at the bottom left of the form area.

Fig. 1 – Página de registo.

Após realizar o registo no site, já é possível fazer login, indicando no campo “email” o email que foi utilizado no registo e de seguida a password.

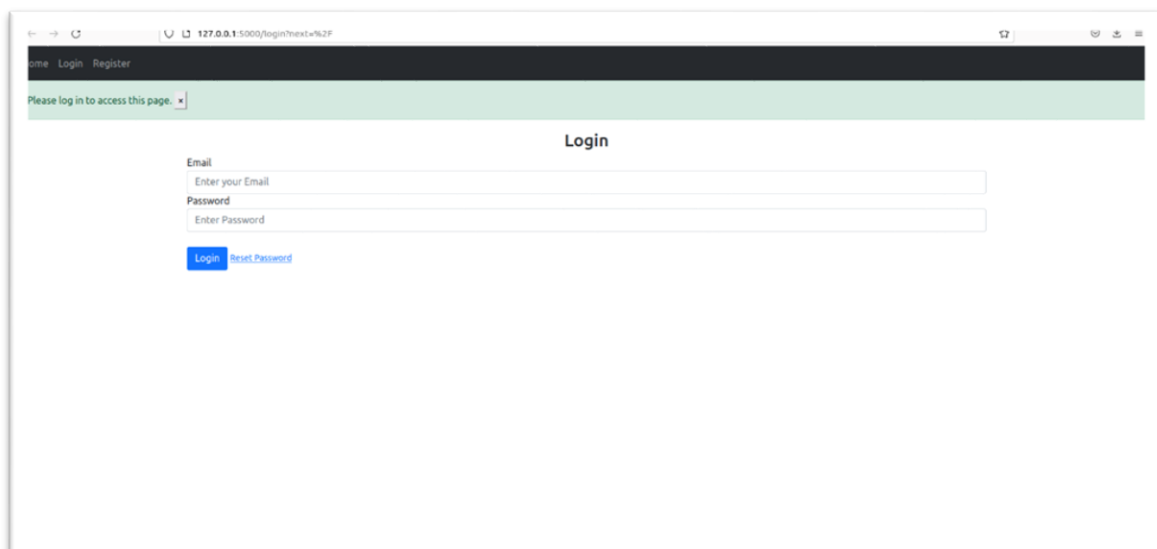
A screenshot of a web browser showing the 'Login' page. The page has a dark header with links 'Home', 'Login', and 'Register'. A green banner at the top says 'Please log in to access this page.' The title 'Login' is centered. Below it, there are two input fields: 'Email' (with placeholder 'Enter your Email') and 'Password' (with placeholder 'Enter Password'). At the bottom left, there is a blue 'Login' button and a link 'Reset Password'.

Fig. 2 – Página de login

Caso ocorra algum problema realizando o login ou o utilizador se tenha esquecido da password, é possível fazer um pedido para dar reset à atual password e criar uma nova, para isso é preciso clicar em “Reset Password” (que aparece ao lado do botão de “Login” na respetiva página) e em seguida indicar o email com qual o utilizador fez o registo.

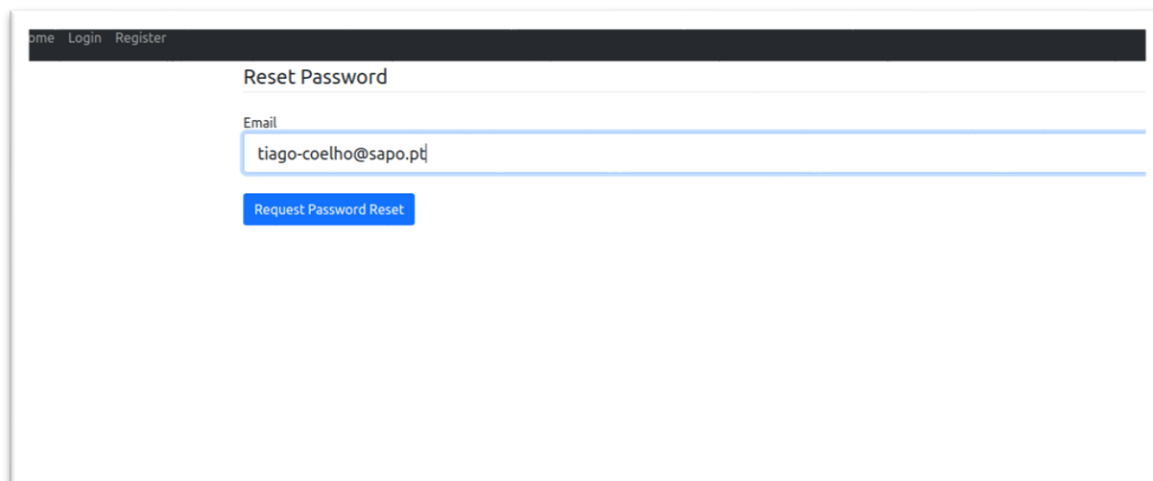
A screenshot of a web application's 'Reset Password' page. At the top, there is a dark navigation bar with links for 'Home', 'Login', and 'Register'. Below this, the page title 'Reset Password' is centered. A text input field labeled 'Email' contains the address 'tiago-coelho@sapo.pt'. Below the input field is a blue button labeled 'Request Password Reset'.

Fig. 3 – Inserir email para dar reset à password.

De seguida, receberá no seu mail uma mensagem com um link com uma token através do qual é possível criar uma nova password que passará a utilizar daí para a frente.


A screenshot of an email titled 'Password Reset Request'. The header shows the sender as '<forumsendergyym@gmail.com>' and the recipient as '<tiago-coelho@sapo.pt>'. The date and time are 'quarta, 10 nov 2021 18:14'. The main body of the email contains a link to reset the password: 'http://127.0.0.1:5000/resetpassword/eyJhbGciOiJIUzUxMiIsImhhdCI6MTYzNjU2ODA1NiwiZXhwIjoxNjM2NTY5ODU2fQ.eyJ1c2VyX2lkIjo0fQ\_-vdb0-uwMo1IjY9A3ueFqeb159UfRg56cVD1vf2za9cGFndQ4hgrCX01Jbifb4a4jeloJX0ga9vJLMPix9X9w'. A note at the bottom states: 'If you did not make this request then simply ignore this email and no changes will be made.'

Fig. 4 – Mensagem enviada para o mail do utilizador com o link que dá acesso à página de criação de nova password.

Através do link, o utilizador é redirecionado para a página onde poderá criar uma nova password.

A screenshot of the 'Reset Password' page after clicking the link in the email. The page has the same navigation bar as Figure 3. Below the 'Reset Password' title, there are two text input fields: 'Password' and 'Confirm Password'. At the bottom of these fields is a blue button labeled 'Reset Password'.

Fig. 5 – Página para criação de nova password.

Após fazer o login, podemos aceder à página onde é possível ver as informações básicas do nosso perfil, como o nome completo e o email. Para tal, basta clicar no botão “Profile” que se encontra no canto superior esquerdo.

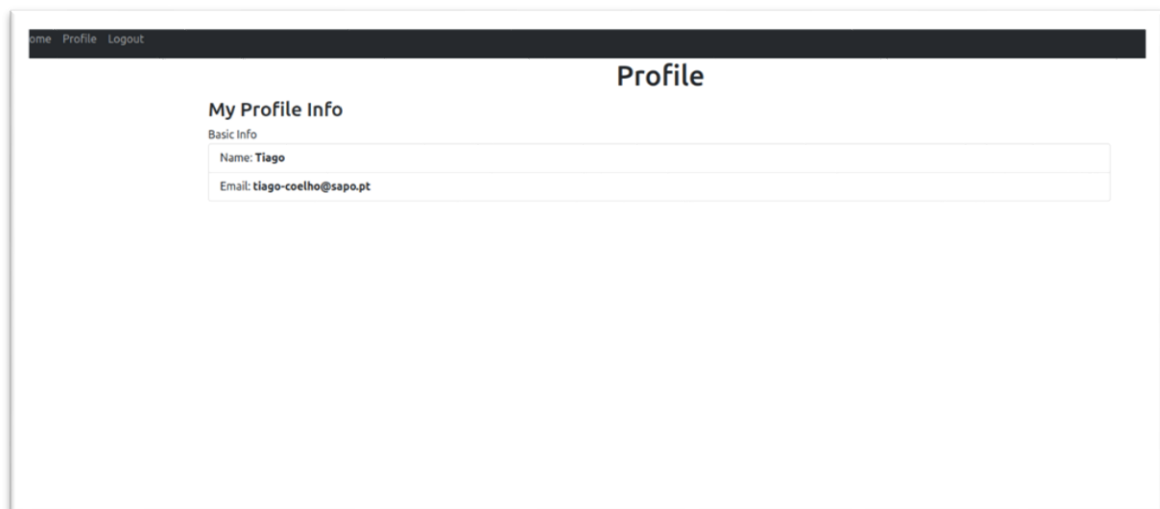


Fig. 6 – Informações básicas do perfil de um utilizador mostradas na página Profile.

Para criar publicações no fórum, é necessário entrar na página “Home”, aqui já é possível ver algumas publicações criadas por outros utilizadores.

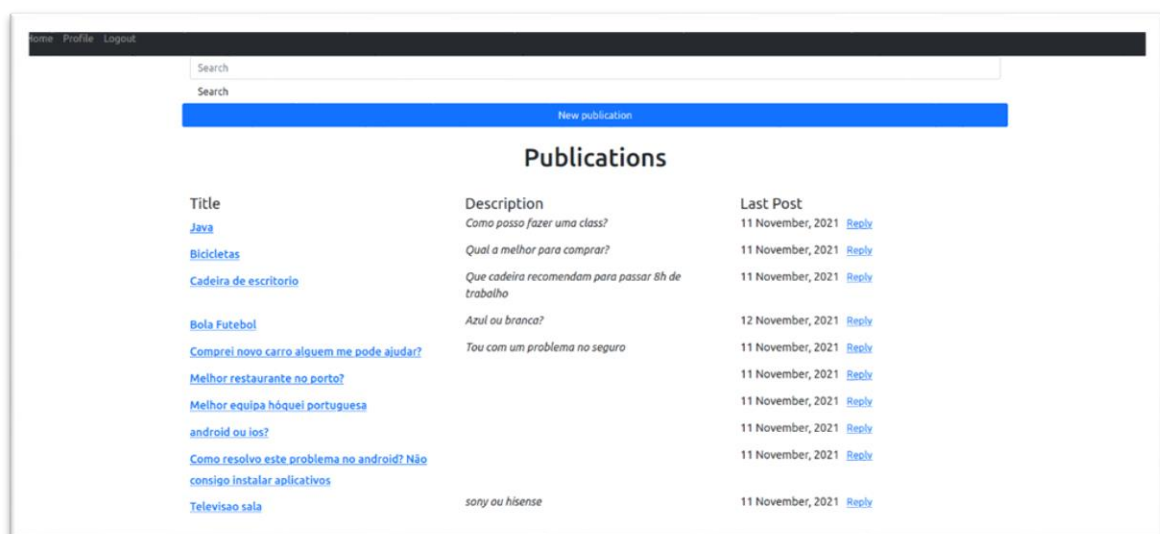


Fig. 7 – Página “Home” onde se pode criar publicações e visualizar outras criadas por utilizadores do fórum.

Ao clicar em “New publication” é possível também criar uma publicação. Irá aparecer um pop-up ao meio a pedir para ser inserido um título e uma descrição. Após preencher estes dois campos, basta clicar em “submit” e a publicação ficará disponível no fórum, sendo possível outros utilizadores também responderem a ela.

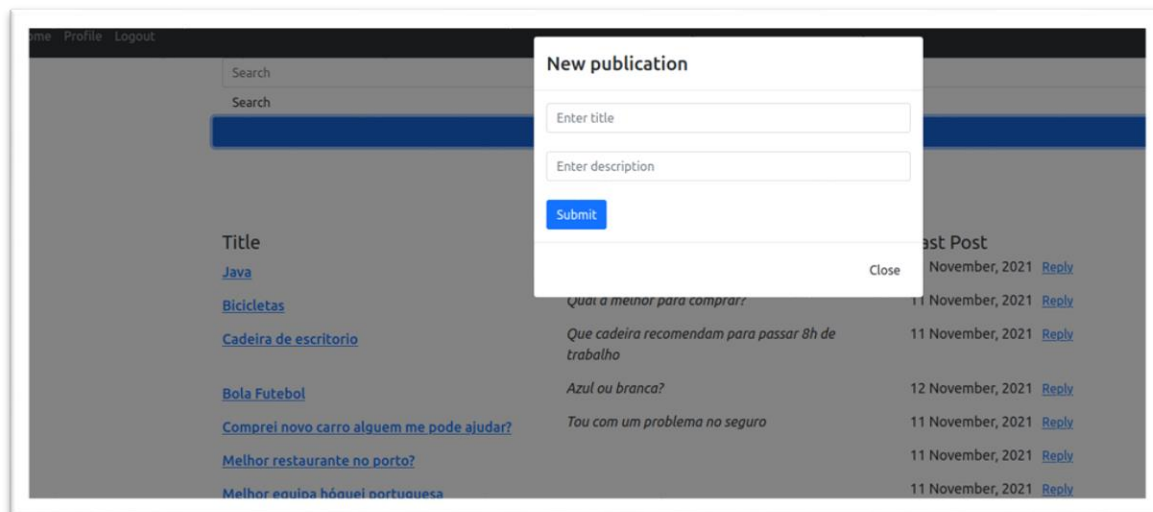


Fig. 8 – Criar uma nova publicação no fórum.

Para responder a uma publicação, na página “Home”, onde são visíveis os posts dos utilizadores, basta clicar no título da publicação ou no botão “Reply”. De seguida, somos redirecionados para a página da publicação onde temos a possibilidade de ver as respostas e escrever uma.

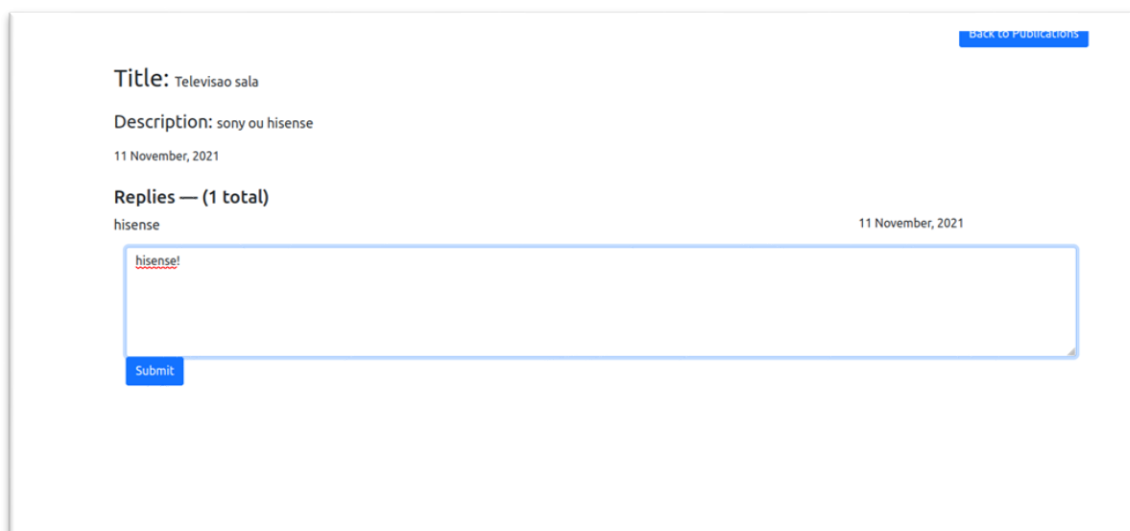
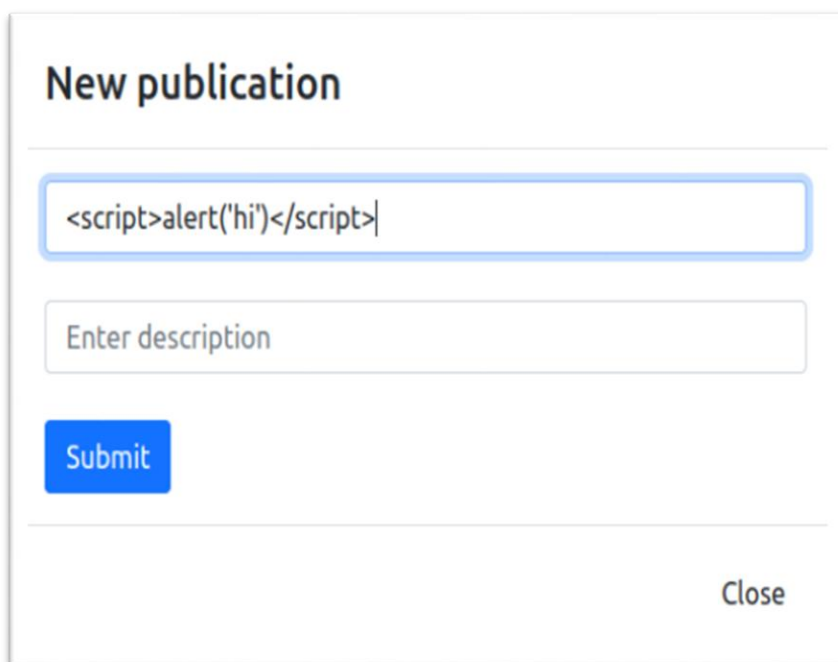


Fig. 9 – Submissão de respostas a uma publicação.

## CWE-79

Através Cross-site scripting (XSS) é possível inserir scripts maliciosos em páginas e aplicativos web que podem redirecionar os utilizadores para links maliciosos ou instalar malwares nos seus computadores.

Na aplicação insegura, é possível inserir scripts com conteúdo malicioso em publicações que ficarão visíveis para todos os utilizadores. Esta vulnerabilidade é do tipo 2, ou seja, é persistente, estando armazenada na publicação enquanto esta estiver no site. Desta forma, sempre que um utilizador abre a publicação, o script será executado podendo redirecionar o utilizador para um link que contém vírus, por exemplo.



The image shows a web form titled "New publication". It contains a text input field with the value `<script>alert('hi')</script>|`, a description input field with the placeholder text "Enter description", a blue "Submit" button, and a "Close" link at the bottom right.

Fig. 10 – Inserir um script através da criação de uma publicação.



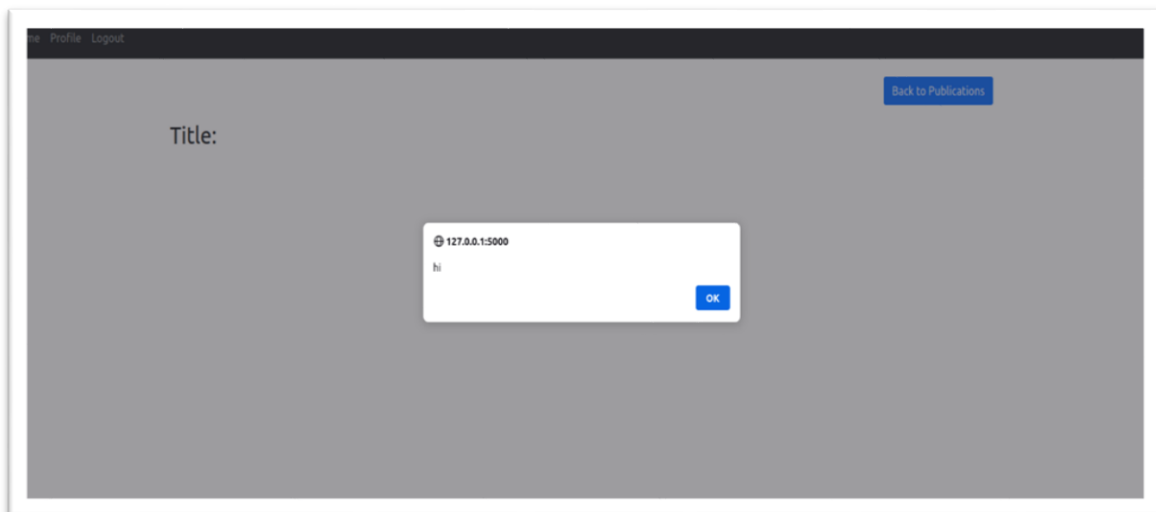


Fig. 11 – Script executado após um utilizador aceder à publicação.

Nesta vulnerabilidade, ao invés de se colocar uma mensagem no script, é colocado uma ligação para a qual o utilizador será redirecionado, esta ligação pode conter um link malicioso.

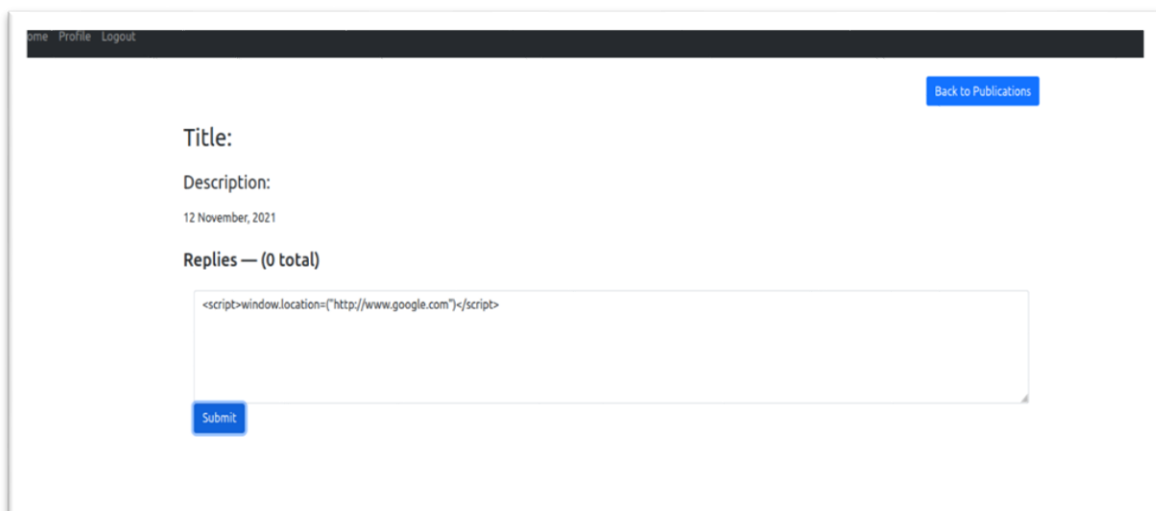


Fig. 12 – Agente envia resposta a uma publicação com um script que irá redirecionar os utilizadores para o link que está contido nele.

Na aplicação insegura, isto acontece devido ao Jinja2, que tem uma funcionalidade que protege a aplicação de ataques XSS, no entanto, essa proteção pode ser desabilitada com “| safe”, que é o que acontece no código da app insegura.

```
class="row pad">
<div class="col-lg-12">
  <div class="panel panel-primary">
    <div class="panel-body">
      <div class="row">
        <div class="col-lg-12">
          <p class="big"><font size="+3">Title: </font><font size="+1">{{ publication.title | safe }}
        </div>
        <div class="col-lg-12">
          <p class="big"><font size="+2">Description: </font><font size="+1">{{ publication.description | safe }}
        </div>
      </div>
    </div>
  </div>
</div>
</div>
```

Fig. 13 – Código de publicação inseguro.

Por outro lado, na app segura, esta proteção está ativa e por isso é impossível executar o mesmo ataque XSS.

```
align="right">
<a href="{{ url_for('views.home') }}" class="btn btn-primary">
  <i class="glyphicon glyphicon-arrow-left"></i> Back to Publications
</a>
>
<div class="row pad">
  <div class="col-lg-12">
    <div class="panel panel-primary">
      <div class="panel-body">
        <div class="row">
          <div class="col-lg-12">
            <p class="big"><font size="+3">Title: </font><font size="+1">{{ publication.title }}</font>
          </div>
          <div class="col-lg-12">
            <p class="big"><font size="+2">Description: </font><font size="+1">{{ publication.description }}</font>
          </div>
        </div>
      </div>
    </div>
  </div>
  <div class="col-lg-12">
    <i class="glyphicon glyphicon-calendar"></i>
  </div>
</div>
```

Fig. 14 – Código da publicação seguro.

## CWE-89

Utilizando SQL Injection é possível inserir uma instrução SQL personalizada e indevida dentro de uma consulta através das entradas de dados do website, como formulários de login ou registo.

No nosso fórum o SQL injection é utilizado para fazer login sem necessidade de saber a password. Para isso necessita de um email válido onde poderá usar outra vulnerabilidade que irá ser referida posteriormente. Já no campo da password, basta escrever algum caracter, pois este é um campo obrigatório (não podendo estar vazio).

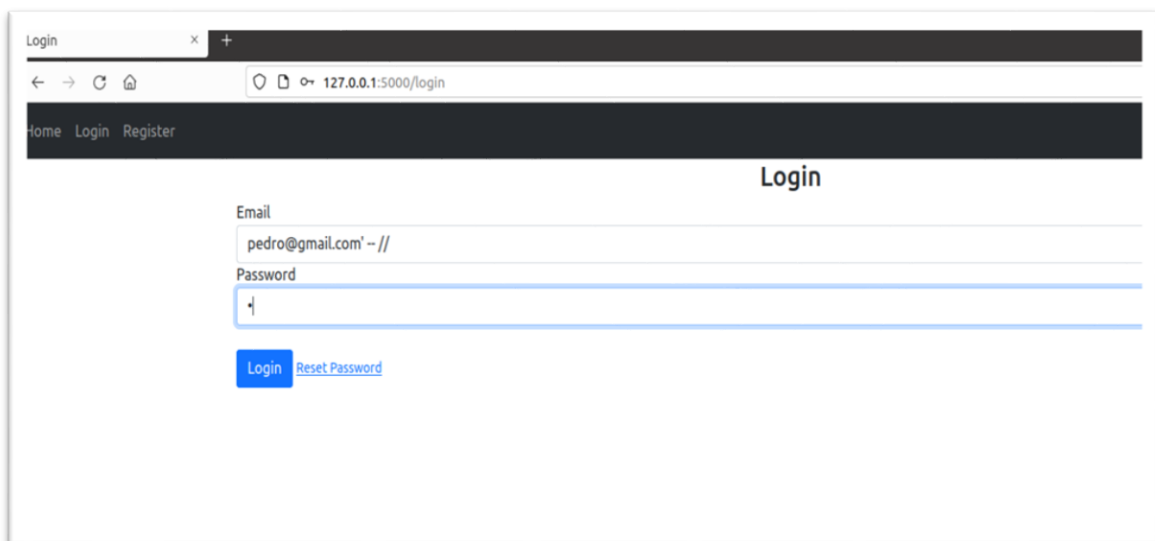


Fig. 15 – Aceder à conta de um utilizador apenas utilizando o seu email. Para isso é colocado ' para o email ficar entre duas plicas, e -- // para deixar tudo o resto em comentário, assim a parte da password será ignorada e é possível fazer o login apenas com email.

No código inseguro, não existe nada que impeça a utilização de caracteres que permitam executar uma SQL injection, por isso é possível usar caracteres como ' e / no campo do email.

```
@auth.route ('/login',methods=['GET','POST'])
def login():
    if request.method=='POST':
        email=request.form.get('email')
        password=request.form.get('password')
        if email==" or password=="":
            flash('Invalid Credentials',category='error')
            return render_template("login.html", user=current_user)
        sql=f"SELECT * FROM User WHERE email='{email}' AND password='{password}'"

        user = User.query.from_statement(db.text(sql)).first()
        if user:
            flash('Logged in sucessfully!',category='sucess')
            login_user(user,remember=True)
            return redirect(url_for('views.home'))
        else:
            flash('Incorrect password,try again.',category='error')
    return render_template("login.html", user=current_user)
```

Fig. 16 -Código da autenticação inseguro.

Já no código seguro, existem condições que limitam a utilização desses caracteres e que os substituem por outros, deste modo é impossível executar o mesmo ataque na versão segura da app.

```
@auth.route ('/login',methods=['GET','POST'])
def login():
    if request.method=='POST':
        email=request.form.get('email')
        password=request.form.get('password')
        if email.find('/'): email.replace('/','\')
        if email.find('-'): email.replace('-','\-')
        if email.find('"'): email.replace('-','\")
        if password.find('/'): password.replace('/','\')
        if password.find('-'): password.replace('-','\-')
        if password.find('"'): password.replace('-','\")
        user=User.query.filter_by(email=email).first()
        if user:
            if check_password_hash(user.password,password):
                flash('Logged in sucessfully!',category='sucess')
                login_user(user,remember=True)
                return redirect(url_for('views.home'))
            else:
                flash('Invalid Credentials, try again!',category='error')
        else:
            flash('Invalid Credentials, try again!',category='error')
    return render_template("login.html", user=current_user)
```

Fig. 17 – Código da autenticação seguro.

## CWE-200

A exposição de informação privada para terceiros (informação presente no URL, como por exemplo o número que corresponde a um id do utilizador, quando este se encontra na sua página de informação pessoal) pode dar acesso a informações de todos os outros utilizadores, ao simplesmente alterar o número correspondente ao id.

Ao estar na página Profile, no URL aparece um número que corresponde ao id do utilizador. Ao alterar esse valor somos redirecionados para as páginas de informação pessoal de outros utilizadores, podendo assim obter as credenciais destes, como o email e nome completo.

Esta vulnerabilidade está ligada à SQL injection pela qual conseguimos fazer login em contas de outros utilizadores sem a necessidade de saber a password. Consequentemente, basta-nos o email que obtivemos através da página de informação pessoal.

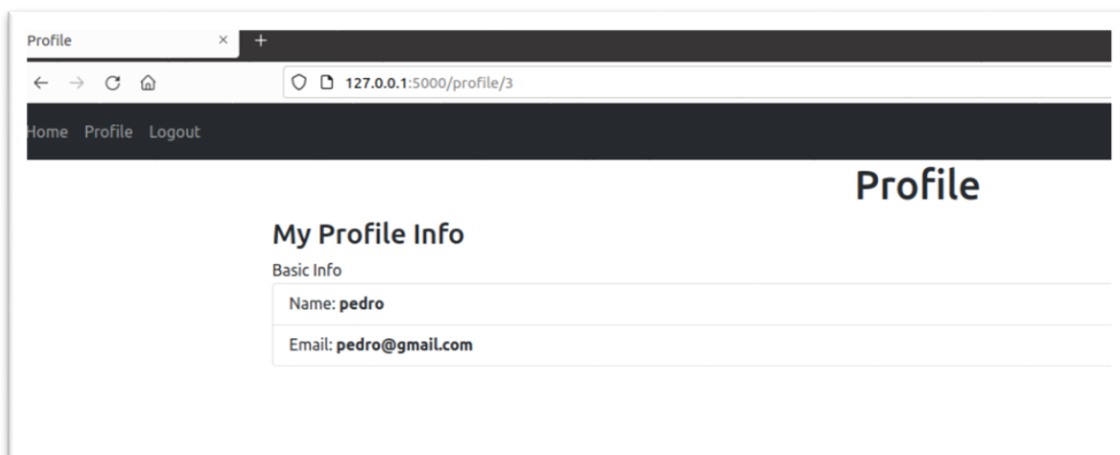


Fig. 18 – Acesso à página de informação pessoal de um utilizador que não o agente, podendo obter o email que é necessário para fazer reset à password e/ou fazer login no site.

Esta vulnerabilidade existe na app insegura pois devido ao código, no URL da página de informação pessoal aparece o id do utilizador, algo que não acontece no código seguro.

```
@auth.route ('/profile/<userid>', methods=['GET', 'POST'])
@login_required
def profile(userid):
    userr=User.query.get(int(userid))
    return render_template("profile.html", user=userr)
```

Fig. 19 – Código inseguro da página de informação pessoal (aparece o id no URL).

```
@auth.route ('/profile', methods=['GET', 'POST'])
@login_required
def profile():
    return render_template("profile.html", user=current_user)
```

Fig. 19 – Código seguro da página de informação pessoal.

## CWE-287

Um utilizador pode afirmar ter determinada identidade (que não a dele) e não havendo formas de provar de que a identidade lhe pertence, é possível roubar informações de outra pessoa (por exemplo, em alguns mecanismos fracos de recuperação de password, é possível alterá-la sabendo apenas o e-mail de outro utilizador).

Na aplicação insegura, o mecanismo de recuperação de password implementado não verifica se a pessoa que insere o email é a proprietária do mesmo, pois o link para repor a palavra-passe não é enviado através de mail. Apenas é necessário o email existente na base de dados para que seja criada uma nova password.

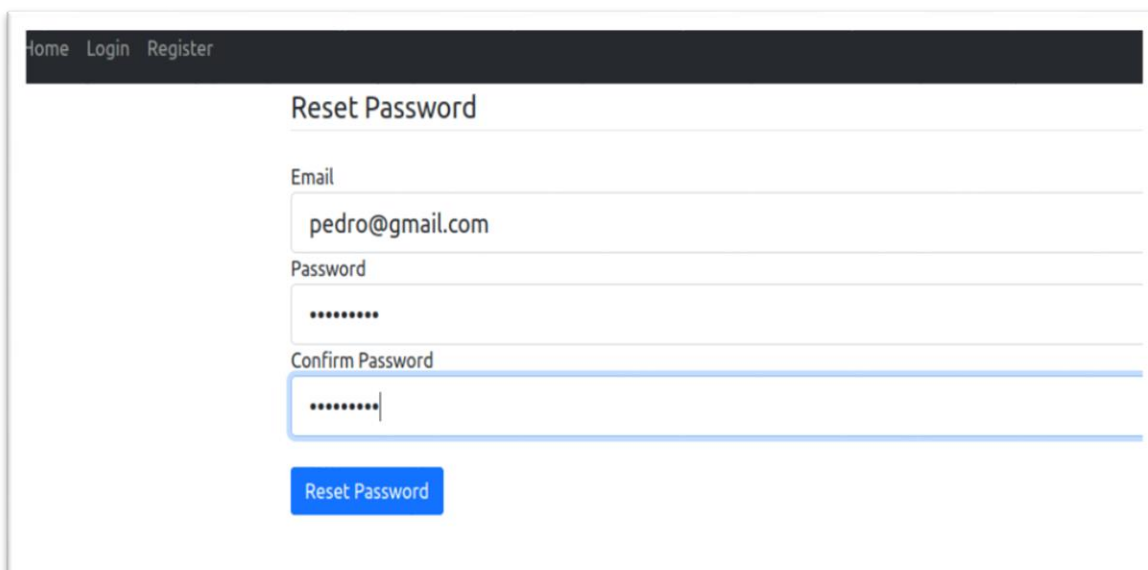


Fig. 20 – Reset da password (na app insegura) de uma conta que não pertence ao agente, apenas sabendo o email do outro utilizador.

```
auth.route('/resetpassword', methods=['GET', 'POST'])
def reset_password():
    if current_user.is_authenticated:
        return redirect(url_for('views.home'))
    form = ResetPasswordForm()
    user = User.query.filter_by(email=form.email.data).first()
    if user:
        if form.validate_on_submit():
            password=request.form.get('password')
            if password.find('/') : password.replace('/', '\\')
            if password.find('-') : password.replace('-', '\\-')
            if password.find('"') : password.replace('-', '\\-')

            user.password = password
            db.session.commit()
            flash('Your password has been updated! You are now able to log in',category='success')
            return redirect(url_for('auth.login'))
        else:
            return render_template('reset_token.html',title='Reset Password',form=form,user=current_user)
```

Fig. 21 – Código inseguro para a recuperação da password.

Por outro lado, na aplicação segura é impossível fazer reset da password sabendo apenas o email de um utilizador pois para criar uma nova palavra-passe é necessário aceder a um link que é enviado para o mail do utilizador. Deste modo, já existe uma maior proteção porque o utilizador deverá ser o único que tem acesso ao seu mail.

```
def send_reset_email(user):
    mail=Mail(current_app)
    token = user.get_reset_token()
    msg = Message('Password Reset Request', sender='forumsendergyym@gmail.com', recipients=[user.email])
    msg.body = f'''To reset your password, visit the following link:
{url_for('auth.reset_token', token=token, _external=True)}

If you did not make this request then simply ignore this email and no changes will be made.
'''
    mail.send(msg)

@auth.route('/resetpassword',methods=['GET','POST'])
def reset_request():
    if current_user.is_authenticated:
        return redirect(url_for('views.home'))
    form = RequestResetForm()
    if form.validate_on_submit():
        user = User.query.filter_by(email=form.email.data).first()
        send_reset_email(user)
        flash('An email has been sent with instructions to reset your password.',category='sucess')
        return redirect(url_for('auth.login'))
    return render_template('reset_request.html',title='Reset Password',form=form,user=current_user)

@auth.route('/resetpassword/<token>',methods=['GET','POST'])
def reset_token(token):
    if current_user.is_authenticated:
        return redirect(url_for('views.home'))
    user = User.verify_reset_token(token)
    if user is None:
        flash('That is an invalid or expired token',category='error')
        return redirect(url_for('auth.reset_request',user=current_user))
    form = ResetPasswordForm()
    if form.validate_on_submit():
        password=request.form.get('password')
        if password.find('/'): password.replace('/', '\\')
        if password.find('-'): password.replace('-', '\\-')
        if password.find(''): password.replace('-', '\\')
        hashed_password = generate_password_hash(password,method='sha256')
        user.password = hashed_password
        db.session.commit()
        flash('Your password has been updated! You are now able to log in',category='success')
        return redirect(url_for('auth.login'))
    return render_template('reset_token.html',title='Reset Password',form=form,user=current_user)
```

Fig. 22 – Código seguro para a recuperação da password.



## CWE-425

Autorização ausente. Neste tipo de vulnerabilidade, o agente consegue aceder a conteúdo da aplicação sem necessitar de passar pela autenticação, sendo esse conteúdo somente visível para os utilizadores da app.

Ao contrário daquilo que acontece na aplicação segura em que para aceder à página Home, onde aparecem todas as publicações dos utilizadores, é necessário estar com o login efetuado, na versão insegura da app, podemos aceder às publicações sem ter de aceder à página Home, ou seja, não é necessário fazer o login, mas apenas mudar o URL (acedendo diretamente à uma publicação).

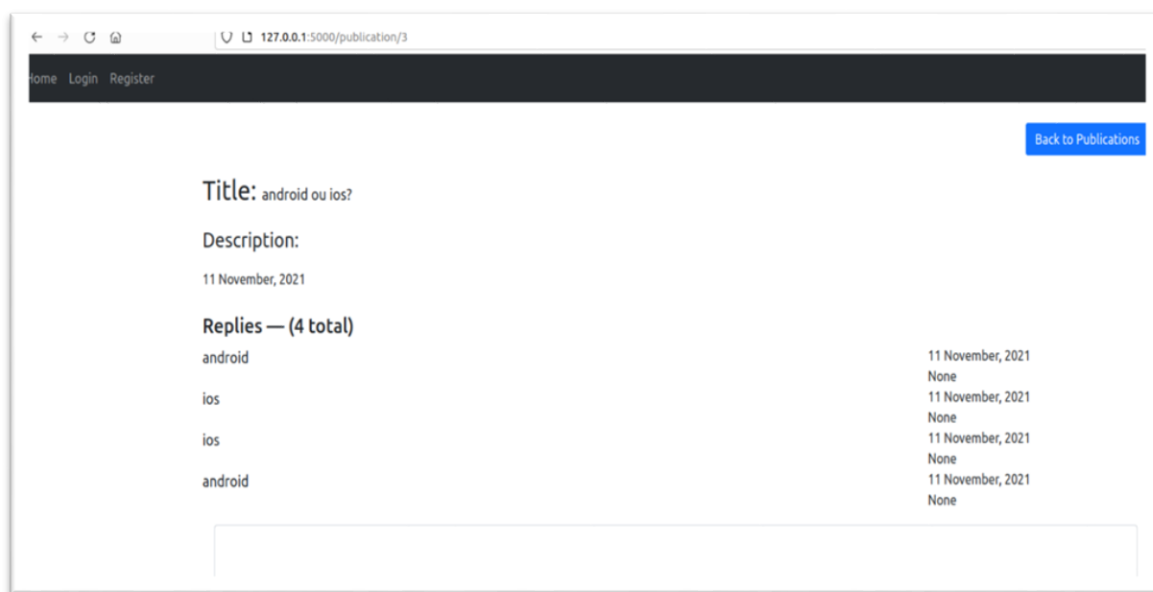


Fig. 23 – Aceder á página de uma publicação sem necessitar de login, bastando apenas inserir o link.

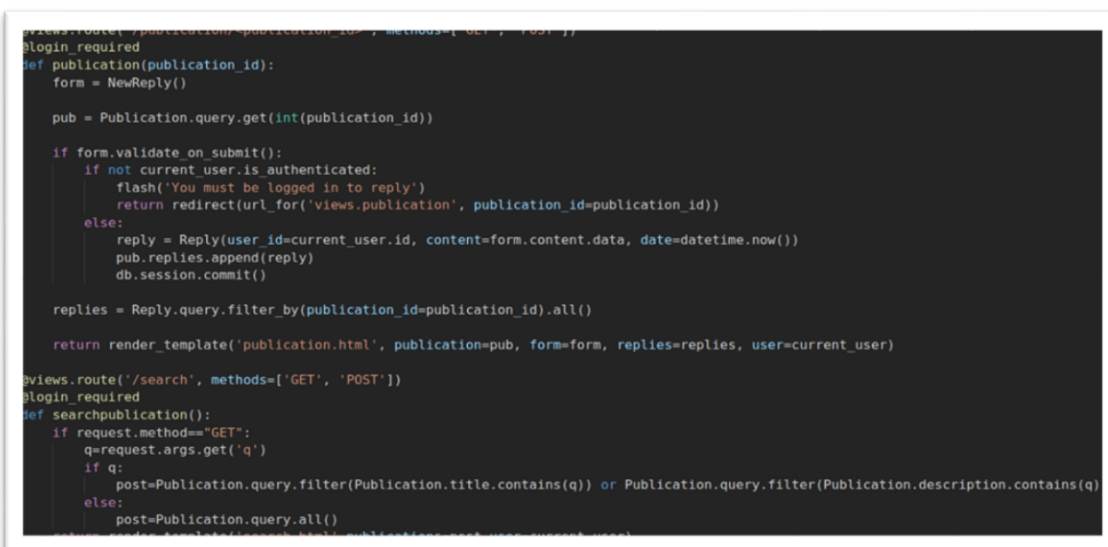


Fig. 24 – Código inseguro da página de uma publicação.



```

@views.route('/publication/<publication_id>', methods=['GET', 'POST'])
@login_required
def publication(publication_id):
    form = NewReply()

    pub = Publication.query.get(int(publication_id))

    if form.validate_on_submit():
        if not current_user.is_authenticated:
            flash('You must be logged in to reply')
            return redirect(url_for('views.publication', publication_id=publication_id))
        else:
            reply = Reply(user_id=current_user.id, content=form.content.data, date=datetime.now())
            pub.replies.append(reply)
            db.session.commit()

    replies = Reply.query.filter_by(publication_id=publication_id).all()

    return render_template('publication.html', publication=pub, form=form, replies=replies, user=current_user)

@views.route('/search', methods=['GET', 'POST'])
@login_required
def searchpublication():
    if request.method=="GET":
        q=request.args.get('q')
        if q:
            post=Publication.query.filter(Publication.title.contains(q)) or Publication.query.filter(Publication.description.contains(q))
        else:
            post=Publication.query.all()
    return render_template('search.html', publications=post, user=current_user)

```

Fig. 25 – Código seguro da página de uma publicação.

## CWE-20 / CWE-521

A aplicação não exige que os utilizadores tenham passwords fortes (com diferentes caracteres especiais), o que torna mais fácil para os invasores comprometerem as contas dos utilizadores.

É importante que uma password seja suficientemente complexa para que seja mais difícil para um invasor a adivinhar.

Na aplicação insegura, não existiam quaisquer requerimentos para formar uma password, o que potenciava a existência de passwords fracas.

Na nossa aplicação segura, para que se faça o registo ou o reset da password, é necessário criar uma que seja forte, com caracteres especiais, de forma que a password tenha alguma complexidade e que não seja fácil de ser comprometida.

```
def register():
    if request.method == 'POST':
        email = request.form.get('email')
        full_name = request.form.get('fullname')
        password = request.form.get('password')
        repassword = request.form.get('repassword')

        user = User.query.filter_by(email=email).first()
        has_symbol = False
        for c in '!@#%&*()_+=-':
            if c in password:
                has_symbol = True
                break

        if user:
            flash('Email already exists.', category='error')
        elif len(email) < 4:
            flash('Email must be greater than 3 characters.', category='error')
        elif len(full_name) < 2:
            flash('First name must be greater than 1 character.', category='error')
        elif password != repassword:
            flash('Passwords don\'t match.', category='error')
        elif len(password) < 10:
            flash('Password must be at least 10 characters.', category='error')
        elif re.search('[0-9]', password) is None:
            flash('Make sure your password has a number in it', category='error')
        elif re.search('[A-Z]', password) is None:
            flash('Make sure your password has a uppercase letter in it', category='error')
        elif not has_symbol:
            flash('Make sure your password has a symbol in it', category='error')
        else:
            if email.find('/'): email.replace('/', '\\')
            if email.find('-'): email.replace('-', '\\-')
            if email.find(''): email.replace(' ', '\\ ')
            if full_name.find('/'): full_name.replace('/', '\\')
            if full_name.find('-'): full_name.replace('-', '\\-')
            if full_name.find(''): full_name.replace(' ', '\\ ')
            if password.find('/'): password.replace('/', '\\')
            if password.find('-'): password.replace('-', '\\-')
            if password.find(''): password.replace(' ', '\\ ')
            new_user = User(email=email, full_name=full_name, password=generate_password_hash(password, method='sha256'))
            db.session.add(new_user)
            db.session.commit()
            login_user(new_user, remember=True)
            flash('Account created!', category='success')
            return redirect(url_for('views.home'))

    return render_template("register.html", user=current_user)
```

Fig. 26 – Código da página de registo da app segura. É possível observar que existem condições que obrigam que a password tenha alguma complexidade, como ter pelo menos 10 caracteres, ter um número, uma letra maiúscula e um símbolo (caracter especial).

Por outro lado, na app insegura, não existem estas condições para a criação de uma password forte, sendo que a única condição é que a password tenha no mínimo 7 caracteres.

```
@auth.route ('/register',methods=['GET','POST'])
def register():

    if request.method=='POST':
        email=request.form.get('email')
        full_name=request.form.get('fullname')
        password=request.form.get('password')
        repassword=request.form.get('repassword')

        user=User.query.filter_by(email=email).first()

        if user:
            flash('Email already exists.',category='error')
        elif len(email) < 4:
            flash('Email must be greater than 3 characters.', category='error')
        elif len(full_name) < 2:
            flash('First name must be greater than 1 character.', category='error')
        elif password != repassword:
            flash('Passwords don\'t match.', category='error')
        elif len(password) < 7:
            flash('Password must be at least 7 characters.', category='error')
        else:
            if email.find('/') : email.replace('/', '\\')
            if email.find('-') : email.replace('-', '\\-')
            if email.find('"') : email.replace('-', '\\"')
            if full_name.find('/') : full_name.replace('/', '\\')
            if full_name.find('-') : full_name.replace('-', '\\-')
            if full_name.find('"') : full_name.replace('-', '\\"')
            if password.find('/') : password.replace('/', '\\')
            if password.find('-') : password.replace('-', '\\-')
            if password.find('"') : password.replace('-', '\\"')
            new_user = User(email=email,full_name=full_name,password=password)
            db.session.add(new_user)
            db.session.commit()
            login_user(new_user,remember=True)
            flash('Account created!',category='success')
            return redirect(url_for('views.home'))

    return render_template("register.html",user=current_user)
```

Fig. 27 – Código da página de registo insegura. Apenas tem uma condição para a criação de uma password, que não a faz ser suficientemente forte.

## Conclusão

Após a realização deste trabalho, ficámos a ter uma maior noção das vulnerabilidades que podem estar presentes nos sites, aprendendo a corrigi-las e/ou evitá-las por forma a construir um site mais seguro e invulnerável.

No projeto, começámos por fazer uma pesquisa profunda e debater qual linguagem/framework usar, que vulnerabilidades usar e como as corrigir, estruturar o nosso Fórum app e o que implementar ou não no mesmo, etc. Após, dividimos tarefas e fomos dividindo novas conforme o projeto avançava, fazíamos mais e novas pesquisas para melhorar, corrigir código. Apesar das tarefas divididas, sempre houve bastante entre ajuda dos membros do grupo, portanto todos trabalhámos um pouco de tudo.

Em suma, tomámos conhecimento de vulnerabilidades que não conhecíamos ou nem nos deparávamos com tal insegurança presente, que na construção de sites nem paramos para pensar que o utilizador irá tentar X ou Y para tirar proveito de dados do site ou acessar páginas indevidas para o próprio. Aumentámos também o nosso conhecimento em python e na framework Flask, aprendendo novas funções, parâmetros, melhorar código, etc..

## **Contribuidores do projeto**

Frederico Vieira, número mec: 98518

Ricardo Antunes, número mec: 98275

Tiago Coelho, número mec: 98385

Vladyslav Mysnyk, número mec: 97548