

Project 1 – Vulnerable eHealth App

Universidade de Aveiro

Segurança Informática e nas Organizações 2022-2023

Catarina Barroqueiro (103895)

Ricardo Covelo (102668)

Rui Campos(103709)

Telmo Sauce (104428)



Introdução

Este relatório foi desenvolvido no âmbito da unidade curricular de Segurança Informática e nas Organizações, tendo como objetivo expor, explicar e demonstrar todas as vulnerabilidades testadas e posteriormente corrigidas no site implementado.

Testámos um total de 8 vulnerabilidades:

- CWE 89 - SQL Injection
- CWE 79 - Cross-Site Scripting
- CWE 256 - Armazenamento de uma password em texto simples
- CWE 620 - Password sem verificação
- CWE 20 - Inserção de dados sem validação
- CWE 311 - Dados sensíveis armazenados sem encriptação
- CWE 549 - Falha na máscara da password
- CWE 756 - Página de erro ausente

CWEs

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

O CWE-89, mais conhecido por 'SQL Injection', refere-se à não neutralização ou neutralização imprópria de elementos especiais que podem modificar o comando SQL. Este tipo de vulnerabilidade é bastante utilizada para alterar a lógica de consulta para ignorar as verificações de segurança ou para inserir instruções adicionais que modificam o banco de dados de *backend*, possivelmente incluindo a execução de comandos do sistema.

EXPLOIT DAS VULNERABILIDADES :

Esta vulnerabilidade tem lugar:

- No Login, uma vez que o invasor, ao inserir

<username> -- //

acaba por comentar a zona que seria onde o programa iria buscar a password.

- Na barra de pesquisa, onde é possível unir o conteúdo de 2 tabelas de base de dados e, assim, mostrar informação crítica ao atacante. Exemplos:

Receber passwords pelo *username*:

**"UNION SELECT 1,2,"
", pword FROM users WHERE username LIKE "Paulo" -- //**

Receber todos os *usernames* e *passwords*:

**" UNION SELECT 1, "
", username, pword FROM users -- //**

Ao escrever as linhas de código acima referidas o atacante consegue informação que não devia, utilizando acesso direto (através do código que comenta as linhas de código onde iria introduzir a password), e acesso indireto (descobrendo a password) dos utilizadores.

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

O CWE-79, mais conhecido por 'Cross Site Scripting', refere-se à não neutralização ou neutralização imprópria do *input* durante a geração da página *web*. Este tipo de vulnerabilidades tira partido do código que está em conteúdo do website, podendo executar o código do lado do utilizador involuntariamente.

EXPLOIT DAS VULNERABILIDADES :

Como já referido, o atacante aproveita e injeta código *javascript* para enganar o utilizador. Desta forma, ao introduzir o seguinte trecho de código:

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

Da mesma forma um utilizador mal intencionado poderia executar código prejudicial ao bom funcionamento do site, e à integridade da informação pessoal das pessoas.

CWE-256: Plaintext Storage of a Password

O CWE-256 refere-se armazenamento de palavras pass em texto sem qualquer tipo de encriptação. Este tipo de vulnerabilidades ocorrem quando uma palavra-passe é armazenada em texto simples nas propriedades, no arquivo de configuração ou na memória de uma aplicação. Armazenar uma palavra-passe de texto simples num arquivo de configuração permite que qualquer pessoa que possa ler o arquivo tenha acesso ao recurso protegido por essa mesma palavra-passe.

EXPLOIT DAS VULNERABILIDADES :

Esta vulnerabilidade é utilizada para ver quais as passwords dos utilizadores, uma vez que estas não estão cifradas. Se uma cifra estiver a ser utilizada, é impossível para o atacante ver a password, vendo apenas informação encriptada.

Este tipo de vulnerabilidade acontece sempre que um funcionário desonesto, ou seja, alguém com acesso à base de dados, tente ver algum tipo de informação mais confidencial. Estas informações não devem ser visíveis para qualquer tipo de pessoa, mesmo sendo administrador de alguma página web. No caso de um ataque em que o hacker tenha acesso à base de dados a informação essencial estará encriptada e não terá uso algum para o atacante.

Utilizando este tipo de vulnerabilidades, o atacante pode ficar a saber as passwords que desejar e pode também alterar a password desse mesmo utilizador, negando, assim, o acesso à sua própria conta.

CWE-620: Unverified Password Change

O CWE-620 refere-se ao momento em que, ao definir uma nova palavra-passe para um utilizador, o produto exibido não exija o conhecimento da sua palavra-passe original ou, de forma alternativa, o uso de outra forma de autenticação. Este tipo de vulnerabilidades pode ser usado por um atacante para alterar as palavras-passe de outro utilizador e, caso consiga aceder à conta do utilizador, este irá conseguir negar o acesso à mesma sem muito esforço.

EXPLOIT DAS VULNERABILIDADES :

Esta vulnerabilidade faz com que, no momento da mudança da password da conta de um utilizador, não seja necessário inserir a password atual.

O formulário apresentado é o seguinte:

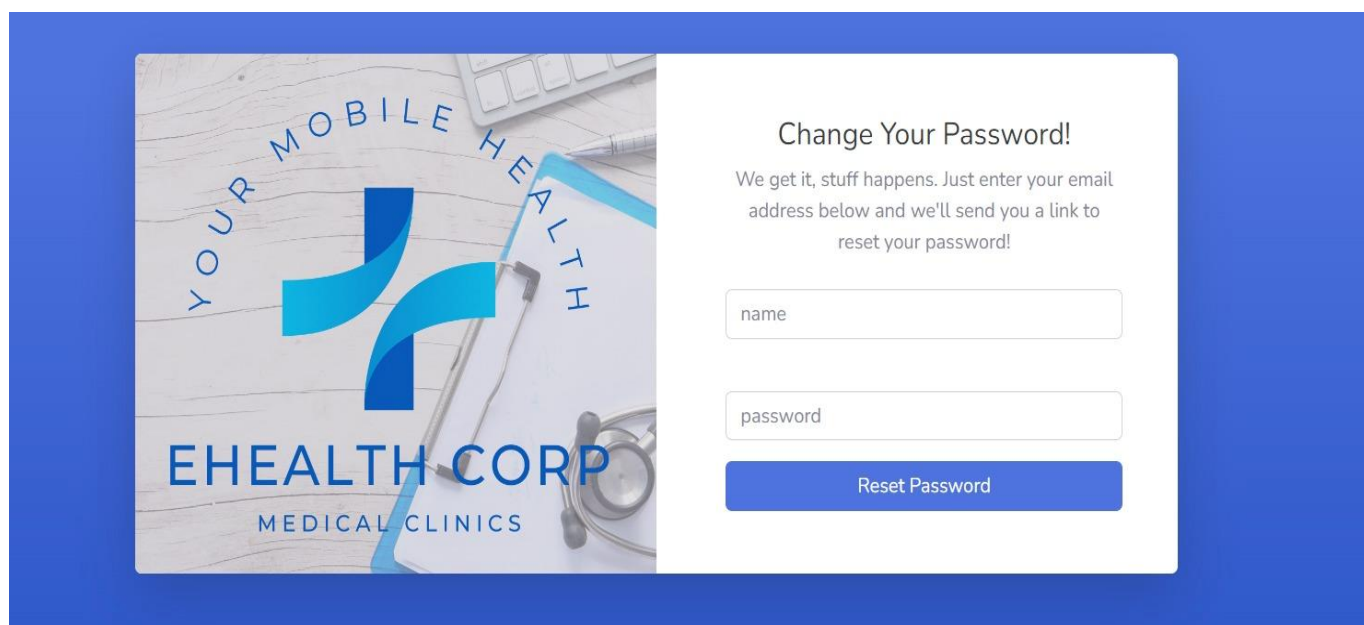


Figura 1- Formulário do site não seguro

Em suma, o atacante, ao realizar o ataque, ganha o acesso à conta do utilizador ao qual a password foi alterada.

CWE-20: Improper Input Validation

O CWE-20 refere-se ao momento em que, o produto, ao receber *input* ou dados de entrada, não os valida ou valida incorretamente se esse *input* contém as propriedades necessárias para processar os dados de forma correta e segura. Este tipo de vulnerabilidades pode ser usado como uma técnica para verificar entradas potencialmente perigosas, a fim de garantir que as entradas sejam seguras para processamento dentro do código ou para comunicar com outros componentes.

EXPLOIT DAS VULNERABILIDADES:

Por exemplo, no formulário preenchido pelo utilizador a fim de fazer login na página, caso a validação esteja incorreta (como demonstrado abaixo), e o utilizador tentar atacar a página seguindo os passos expostos na explicação das vulnerabilidades relativas ao CWE-89 (SQL Injections), o mesmo conseguirá aceder à página saltando o passo da autenticação. Assim como referido acima, o atacante poderia executar o código de exemplo:

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

Da mesma forma que os *scripts* e trechos de código mal intencionados, prejudicando a integridade geral do site.

CWE-311: Missing Encryption of Sensitive Data

O CWE-311 consiste na não encriptação de informação crítica antes da mesma ser armazenada, deste modo a informação é guardada sem qualquer garantia de confidencialidade, integridade ou responsabilidade que a encriptação garante.

EXPLOIT DAS VULNERABILIDADES :

Através de 'SQL Injections', por exemplo (entre outros tipos de ataques possíveis), os atacantes poderão facilmente aceder à base de dados extraindo a informação legível, ficando deste modo na posse de dados pessoais dos utilizadores, entre eles as passwords das respetivas contas. Tal como ilustrado abaixo os inputs são armazenados na base de dados sem sofrerem qualquer tipo de encriptação, tanto no *login* (Figura 6, como na página de registo (Figura 7):

```
@app.route("/login", methods=["POST", "GET"])
def login(*args):

    if len(args) != 0:
        return render_template('login.html', error=args[0])

    db=sql.connect("webDB.db")
    if(request.method == "POST"):
        name = request.form["name"]
        password = request.form["Password"]
        result=db.execute("SELECT * FROM users WHERE name='"+name+"' AND password='"+password+"'")
        if result.fetchall():
            db.close()
            session['name']=name
            return index()
        else:
            db.close()
            return login(1)
    else:
        return render_template('login.html')
```

Figura 2- Página de Login (ficheiro app.py)

```
if len(args) != 0:
    return render_template('register.html', error=args[0])

if(request.method == "POST"):
    name = request.form["Name"]
    emailAddress = request.form["EmailAddress"]
    password = request.form["Password"]
    db=sql.connect("webDB.db")
    result=db.execute("SELECT * FROM users WHERE name='"+name+"' OR email='"+emailAddress+"'")
    data=result.fetchall()
    if(data!=[]):
        if(data[0][1]==name):
            print("Username already exists")
            return register(1)
        else:
            print("email")
            return register(2)
    else:
        session['name']=name
        db.execute("INSERT INTO users VALUES (NULL,'"+name+"','"+emailAddress+"','"+password+"')")
        db.commit()
        db.close()
        return profile()
    else:
        return render_template('register.html')
```

Figura 3- Página de Registo (ficheiro app.py)

CWE-549: Missing Password Field Masking

O CWE-549 é relativo à falha da máscara no campo da password, preenchido no momento em que o *user* faz login ou se regista no site, falhando a máscara a probabilidade de atacantes observarem e capturarem as passwords dos utilizadores aumenta significativamente.

EXPLOIT DAS VULNERABILIDADES :

Faltando a máscara da password, no exato momento em que o utilizador introduzir a mesma com o objetivo de realizar login este poderá estar num espaço comum a várias pessoas, ou até mesmo a fazer partilha do seu ecrã, por exemplo, expondo deste modo a sua password de acesso ao *WebSite* ou *App*. Torna-se assim possível que seja efetuado login por uma pessoa que não o utilizador fidedigno.

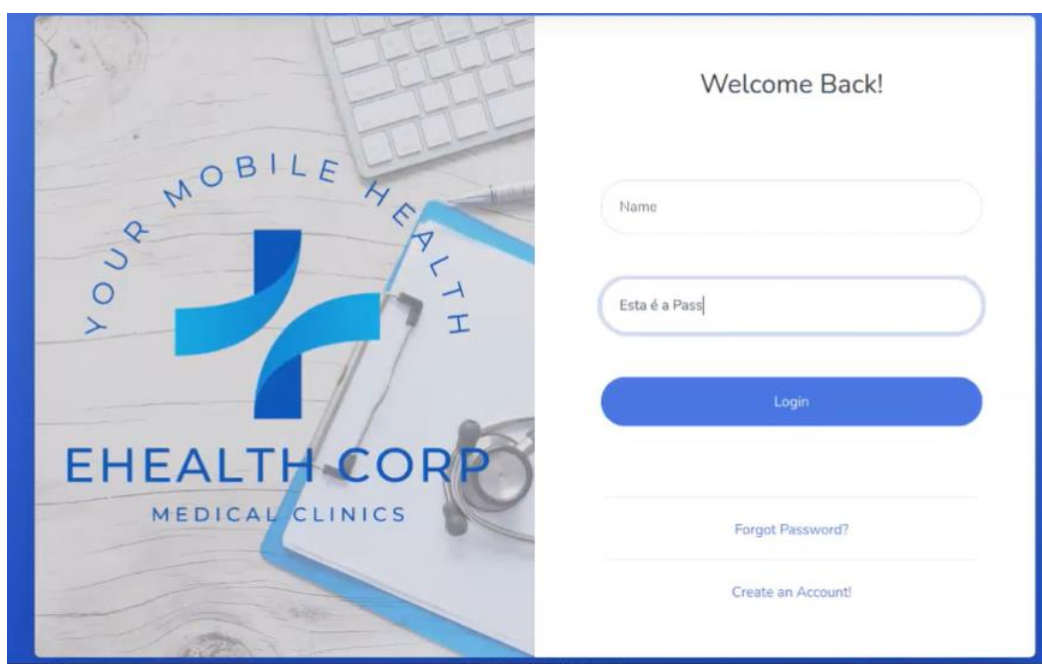


Figura 4 - Página de Login

Esta vulnerabilidade resulta da ausência de uma máscara definida no atributo *“type”* do input, tal como demonstrado abaixo:

```
<div class="form-group">
  <input type="text" class="form-control form-control-user"
    id="exampleInputPassword" placeholder="Password" name="Password">
</div>
```

Figura 5 – Campo de preenchimento da password sem máscara

CWE-756: Missing Custom Error Page

O CWE-756 é referente a produtos que não possuem em si integradas páginas relativas aos erros que possam ocorrer durante a experiência do utilizador. Isto é, quando ocorre algum tipo de erro, o mesmo não é denunciado numa página específica e é exposta muita informação a partir da página de erro gerada pela biblioteca utilizada para o *backend*, expondo deste modo erros descritivos podendo estar também expostas informações críticas, o que não abona a segurança da página.

EXPLOIT DAS VULNERABILIDADES :

O facto de ser exposta a página de erro gerada pela biblioteca utilizada para o *backend* (figura 11) expõe demasiada informação critica, entre ela poderá surgir o código utilizado para as *Queries*, como demonstrado:

OperationalError

sqlite3.OperationalError: table users has 5 columns but 4 values were supplied

Traceback (most recent call last)

```
File "C:\Users\catal\OneDrive\Ambiente de Trabalho\U N I\Segurança\Projeto1\assignment-1---vulnerable-ehealth-application-grupo_13\app\frontend\env\lib\site-packages\flaskapp.py", line 2548, in __call__
    return self.wsgi_app(environ, start_response)
File "C:\Users\catal\OneDrive\Ambiente de Trabalho\U N I\Segurança\Projeto1\assignment-1---vulnerable-ehealth-application-grupo_13\app\frontend\env\lib\site-packages\flaskapp.py", line 2528, in wsgi_app
    response = self.handle_exception(e)
File "C:\Users\catal\OneDrive\Ambiente de Trabalho\U N I\Segurança\Projeto1\assignment-1---vulnerable-ehealth-application-grupo_13\app\frontend\env\lib\site-packages\flaskapp.py", line 2525, in wsgi_app
    response = self.full_dispatch_request()
File "C:\Users\catal\OneDrive\Ambiente de Trabalho\U N I\Segurança\Projeto1\assignment-1---vulnerable-ehealth-application-grupo_13\app\frontend\env\lib\site-packages\flaskapp.py", line 1822, in full_dispatch_request
    rv = self.handle_user_exception(e)
File "C:\Users\catal\OneDrive\Ambiente de Trabalho\U N I\Segurança\Projeto1\assignment-1---vulnerable-ehealth-application-grupo_13\app\frontend\env\lib\site-packages\flaskapp.py", line 1820, in full_dispatch_request
    rv = self.dispatch_request()
File "C:\Users\catal\OneDrive\Ambiente de Trabalho\U N I\Segurança\Projeto1\assignment-1---vulnerable-ehealth-application-grupo_13\app\frontend\env\lib\site-packages\flaskapp.py", line 1796, in dispatch_request
    return self.ensure_sync(self.view_functions[rule.endpoint])(**view_args)
File "C:\Users\catal\OneDrive\Ambiente de Trabalho\U N I\Segurança\Projeto1\assignment-1---vulnerable-ehealth-application-grupo_13\app\frontend\app.py", line 85, in register
    db.execute("INSERT INTO users VALUES (NULL, '"+name+"', '"+emailAddress+"', '"+password+"');")
sqlite3.OperationalError: table users has 5 columns but 4 values were supplied
```

The debugger caught an exception in your WSGI application. You can now look at the traceback which led to the error.

Figura 6 - Página de erro gerada pela biblioteca utilizada para o backend

Esta página deve-se à atribuição do valor 'True' à *flag debug* aquando da definição do modo de execução da *main* da App (figura 12):

```
if(__name__) == "__main__":
    app.run(debug=True)
```

Figura 7 – debug=True