

# eHealth Application Analysis

Inês Santos  
Mauro Filho  
Patricia Cardoso

Trabalho realizado no âmbito da disciplina de  
Segurança Informática e nas Organizações

DETI  
Universidade de Aveiro  
16 de novembro de 2022

# Conteúdo

<b>1</b>	<b>Introdução</b>	<b>1</b>
<b>2</b>	<b>Vulnerabilidades</b>	<b>1</b>
2.1	CWE-89: SQL Injection . . . . .	1
2.2	CWE-79: Cross-site Scripting . . . . .	2
2.3	CWE-352: Cross-Site Request Forgery (CSRF) . . . . .	2
2.4	CWE-22: Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal') . . . . .	3
2.5	CWE-798: Use of Hard-coded Credentials . . . . .	4

# 1 Introdução

Este documento contém exemplos de exploração de cada vulnerabilidade implementada.

## 2 Vulnerabilidades

### 2.1 CWE-89: SQL Injection

Esta falha de segurança resulta de uma falta de controlo na geração de comandos SQL, normalmente resultantes da concatenação de strings com input do utilizador. Isso possibilita a que o utilizador altere o comando e receba/insira informação indevidamente.

```
result = cs.execute("SELECT * FROM results WHERE resultID = '"+str(code)+"' AND userID = '"+str(userID)+"';")
rows = result.fetchall()
```

Figura 1: Geração indevida de comandos SQL.

The image shows two web forms side-by-side. On the left is a 'Login' form with a single text input field containing the text 'abc@gmail.com';--' and a 'Submit' button below it. On the right is a 'Signup' form with three text input fields labeled 'Enter your name', 'Enter your email', and 'Enter your password', followed by a 'Submit' button.

Figura 2: A inserção dos símbolos ';- no final de um email registado permitem o acesso indevido a essa conta.

Apesar da sua gravidade, a resolução do problema é simples, através do uso de queries parametrizadas para gerar os comandos de forma mais controlada.

```
result = cs.execute("SELECT * FROM results WHERE resultID = ? AND userID = ?;",(str(code),str(userID)))
rows = result.fetchall()
```

Figura 3: Geração de comandos com recurso a queries parametrizadas.

## 2.2 CWE-79: Cross-site Scripting

Esta vulnerabilidade é causada pela falta de verificação sobre o input de utilizadores, que permite o armazenamento de scripts maliciosos que serão executados quando um utilizador carrega a página. Neste caso foi usado XSS juntamente com CSRF para guardar uma imagem no campo de data de uma marcação de consulta, com código que é executado quando esta é carregada pela vítima.

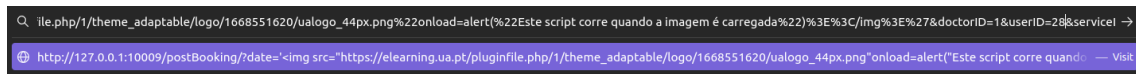


Figura 4: URL criado especificamente para armazenar código malicioso na página de perfil de um certo utilizador (determinado pelo seu userID).

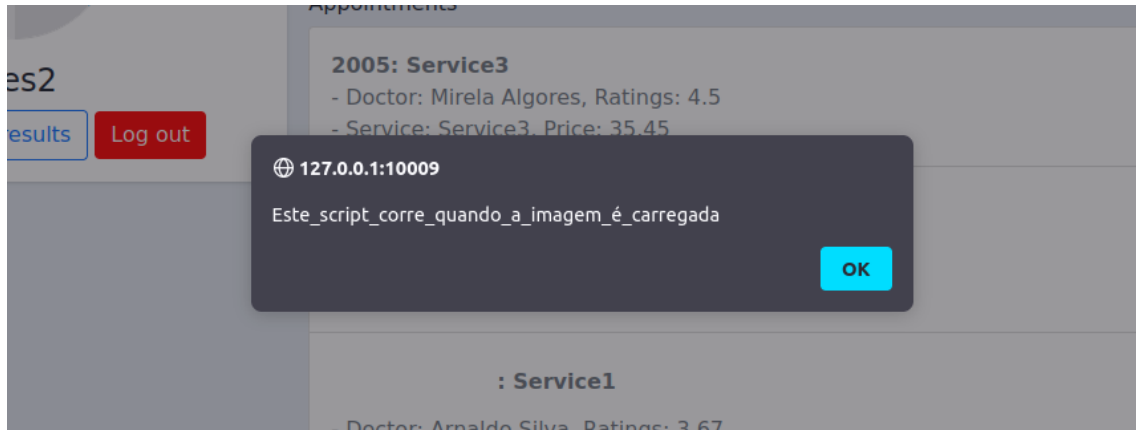


Figura 5: Resultado quando a página de perfil é carregada.

Este problema é resolvido com recurso a uma maior verificação dos inputs recebidos, sendo neste caso verificado se a string recebida é mesmo uma data.

```
bookDate = str(bookingInfo["date"]).split('-')
if len(bookDate) != 3 or not bookDate[0].isnumeric() or not bookDate[1].isnumeric() or not bookDate[2].isnumeric():
    return json.dumps(False)
```

Figura 6: Verificação da string para garantir que é uma data.

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

O exemplo anterior foi possível apenas porque não havia nenhuma verificação de que a marcação de consulta foi intencionalmente feita pelo utilizador. A string de email cifrada com a chave pessoal do utilizador serve para verificar se o utilizador realmente fez login antes de marcar uma consulta

```

for row in rows:
    dic = {}
    dic["email"] = row[1]
    dic["password"] = row[2]
    dic["userID"] = row[3]
    dic["name"] = encryption.decryptBytes( row[0], dic["password"], "cryptoKeys/"+str(dic["userID"])+".privk.pem")
    dic["email"] = base64.b64encode(encryption.encryptString(row[1], "cryptoKeys/"+str(dic["userID"])+".pubk.pem")).decode('utf-8')
    array.append(dic)
cs.close()
db.close()
return json.dumps(array)

```

Figura 7: Quando o login é feito, o browser recebe a string de email cifrada com a chave do utilizador, que servirá para verificar se as marcações de consulta são legítimas

## 2.4 CWE-22: Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')

Quando um utilizador contacta a clínica através da página inicial, esta mensagem é armazenada num ficheiro no servidor, usando o título da mensagem como nome. Isto permite a que o utilizador manipule o path de armazenamento para que o ficheiro seja guardado fora do diretório '/app/Contacts', possivelmente substituindo ficheiros essenciais ao funcionamento da aplicação.

# Contact us

Do you have any questions? Please do not hesitate to contact us directly.

Your name
Your email

Subject

This will overwrite the database

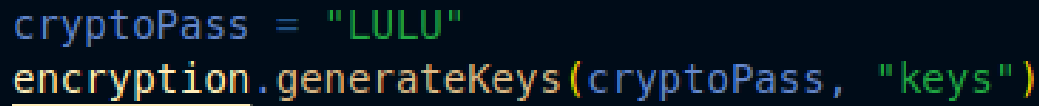
Your message

Figura 8: Este título permite que a base de dados seja substituída pelo novo ficheiro, indisponibilizando todos os serviços da aplicação.

Para evitar esta falha pode-se processar o título para impedir tentativas de sair do diretório, ou simplesmente utilizar outra informação não fornecida pelo utilizador para gerar o path do ficheiro, como por exemplo a data e hora a que a mensagem foi recebida.

## 2.5 CWE-798: Use of Hard-coded Credentials

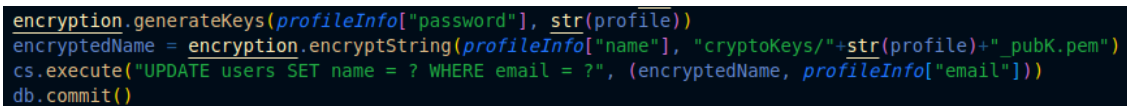
O armazenamento de chaves/passwords diretamente no código, e a utilização destas para várias situações diferentes facilita a que um atacante as obtenha. A quantidade de dados comprometidos também será maior do que se forem usadas várias chaves, criadas para uso específico de cada utilizador.



```
cryptoPass = "LULU"  
encryption.generateKeys(cryptoPass, "keys")
```

Figura 9: Geração de chaves com password estática<sup>1</sup>

Chaves individuais para cada utilizador protegem melhor os dados confidenciais deste. Neste caso o único dado confidencial será o nome.



```
encryption.generateKeys(profileInfo["password"], str(profile))  
encryptedName = encryption.encryptString(profileInfo["name"], "cryptoKeys/"+str(profile)+"_pubK.pem")  
cs.execute("UPDATE users SET name = ? WHERE email = ?", (encryptedName, profileInfo["email"]))  
db.commit()
```

Figura 10: Chaves geradas com recurso à password do utilizador

---

<sup>1</sup>O ficheiro encryption.py contém as funções de criptografia usadas pelo servidor