

Aula12 - 06/maio/2019

Grupo 3

1) Injection

Pergunta 1.1_ String SQL Injection

Para responder a esta questão fomos ao “WebGoat” e resolvemos o exercício *Injection Flaws* -> *String SQL Injection*:

O objetivo foi utilizar *SQL Injection* que resultasse em todos os cartões de crédito serem apresentados. A tautologia usada foi a seguinte: *'grupo3' or '1'='1'*. E isso permitiu-nos obter a lista de todos os cartões de crédito como podemos verificar na imagem seguinte.

Enter your last name:

SELECT * FROM user_data WHERE last_name = 'grupo3' 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_ Numeric SQL Injection

No WebGoat tente resolver o exercício *Injection Flaws* -> *Numeric SQL Injection*:

O objetivo é utilizar *SQL Injection* que resulte em todos os dados meteorológicos serem apresentados. Para que a selecionar *Columbia* aparecessem todos os resultados foi necessário através no botão *Go* selecionar o *Inspect Element* onde encontramos a *query* correspondente ao país mencionado acima. Porém foi necessário alterar o conteúdo da descrição do input do HTML com uma tautologia. O resultado desta questão encontra na figura a seguir:



Pergunta 1.3_ Database Backdoors

No WebGoat vamos resolver o exercício *Injection Flaws -> Database Backdoors*:

O objetivo foi utilizar *SQL Injection* para executar mais do que um comando SQL, para uma conta com o *ID 101*. A seguinte imagem mostra como é que a aplicação se comporta com este *ID 101*. Portanto é fornecida a informação do ID introduzido.

Stage 1: Use String SQL Injection to execute more than one SQL Statement. The first stage of this lesson is to teach you how to use a vulnerable field to create two SQL statements. The first is the system's while the second is totally yours. Your account ID is 101. This page allows you to see your password, son and salary. Try to inject another update to update salary to something higher

User ID:

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

User ID	Password	SSN	Salary	E-Mail
101	larry	386-09-5451	55000	larry@stooges.com

Ao explorar a vulnerabilidade para alterar o salário para um valor mais elevado através da criação de uma instrução *SQL* "*userid=101; update employee set salary=1005500*", obtivemos o seguinte resultado:

Stage 2: Use String SQL Injection to inject a backdoor. The second stage of this lesson is to teach you how to use a vulnerable field to inject a trigger that would act as SQL backdoor, the syntax of a trigger is: CREATE TRIGGER myBackDoor BEFORE INSERT ON employee FOR EACH ROW BEGIN UPDATE employee SET email='john@hackme.com' WHERE userid = NEW.userid

Note that nothing will actually be executed because the current underlying DB doesn't support triggers.

* You have succeeded in exploiting the vulnerable query and created another SQL statement. Now move to stage 2 to learn how to create a backdoor or a DB worm

User ID:

select userid, password, ssn, salary, email from employee where userid=**101; update employee set salary=1005500**

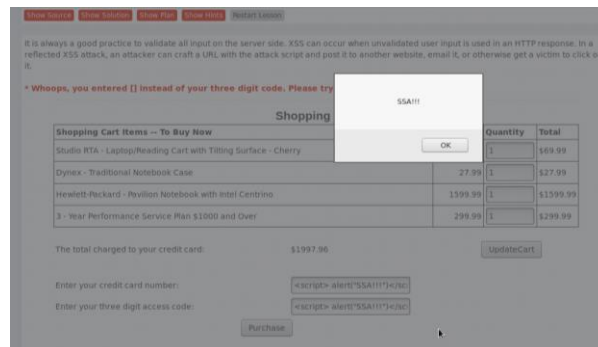
User ID	Password	SSN	Salary	E-Mail
101	larry	386-09-5451	1005500	larry@stooges.com

2) XSS

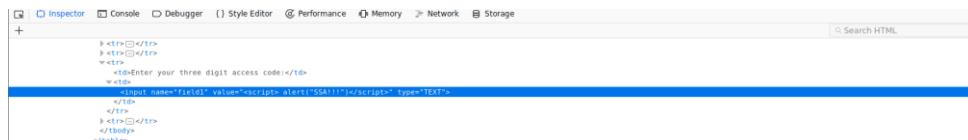
Pergunta 2.1_ Reflected XSS

No WebGoat tente resolver o exercício Cross-Site Scripting (XSS) -> Reflected XSS Attacks:

O objetivo é utilizar o formulário de compras para refletir o input do utilizador. Como o código de acesso que deverá possuir apenas três dígitos é vulnerável, então através da script `<script>alert("SSA!!!") </script>` foi aberta uma nova janela com apenas "SSA!!!".



Se explorarmos o campo que deve ter apenas três dígitos no código HTML no *campo name="field1"*, vemos que é possível introduzir algo malicioso no *url* e consequentemente fazer com que alguém execute um script malicioso no seu browser.



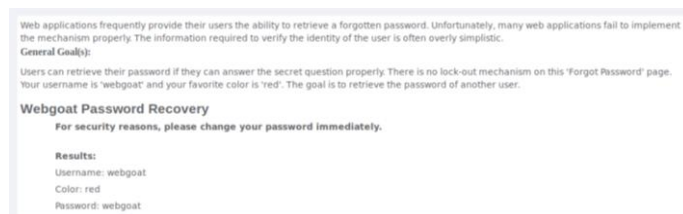
3) Quebra na Autenticação

Pergunta 3.1_ Forgot Password

No WebGoat tente resolver o exercício *Authentication flaws* -> *Forgot Password*.

Os mecanismos de autenticação têm usualmente a funcionalidade de permitirem recuperar a password através da resposta a uma questão pessoal.

- A aplicação *WebGoat* permite que os utilizadores recuperem a password se conseguirem responder a uma questão sobre a sua cor favorita. A figura a seguir permite ver o que acontece ao tentar o utilizador "*webgoat*" e cor "*red*" é possível ver a password correspondente ao utilizador inserido:



- Agora vamos tentar atacar outro utilizador. Os utilizadores que existem normalmente num sistema são os "administradores." De todos esses utilizadores, a conta que gostaria de comprometer seria a do administrador, assim vamos tentar o username "admin". Através de *força bruta*, introduzindo as cores possíveis, foi praticável recuperar a password deste utilizador através dar cor: "green".

