Web Pentest 101

from abrasax to /dev/null

0x02: SQL Injection

Learning through cenarios

- WebGoat
  - setup
- Cenario 2
  - Get credentials
- Bonus: shell upload

- Hands on


WEBGOAT
Learn To Fix Web Application Flaws In Real-time

Instalation

- Github page: https://github.com/WebGoat/WebGoat
- Download project and run with java or...
- Use docker! (Recommended)
- With docker just type:

curl https://raw.githubusercontent.com/WebGoat/WebGoat/develop/docker-compose.yml | docker-compose -f - up

- http://localhost:8080/WebGoat/login

Cenario 1

- Imagine that we want to build a site, this site can be called http://lalapathanshadi.com/
- Admins can login to manage this website, but only them has this login
- This website must store administrators data, like username, password (we will talk about this later), name, and picture name
- When someone type it's login, the server must check in the database if this user and password matches with some administrator credentials
- If it's work, nice, authorized and the user is redirected to the admin panel, but if this doesn't work you're back to login page
- But someone is breaking this system...

But how this validation works?

```
select * from tbl_employees
SELECT COUNT(emp_name)FROM tbl_employees
```

	id	emp_name	emp_age	emp_salary	join_date
1	1	Mike	35	5000.50	2016-01-01 00:00:00.000
2	2	Michale	30	4500.50	2016-05-01 00:00:00.000
3	3	Jimmy	27	3000.00	2016-05-03 00:00:00.000
4	4	Shaun	30	3500.00	2017-04-03 00:00:00.000
5	5	Ben	45	4500.00	2015-03-15 00:00:00.000
6	6	John	28	2600.00	NULL

	(No column name)
1	6

Basic queries structure

- One way to check if there is a admin's username called *vrechson* with password *secureP4ss* is to ask:
- Hey system, select for me each information from admins table where i can find a username with name vrechson and password secureP4ss
- Or, in sql:
- SELECT * FROM admins WHERE username = 'vrechson' AND password = 'secureP4ss';

- So, if we have some return we can assume that we have an admin and the entire line with each attribute value will be returned. This data will be used in his panel saying things like, welcome back matheus vrech

So what?

Poisoning inputs

- Well, we saw that we has to validade this query: SELECT * FROM admins WHERE username = 'vrechson' AND password = 'secureP4ss';
- To validate this query the system must get the user and password from POST (or GET) parameters. Well, so we have complete control to what is sent to this comparation.
- What could happens if instead of vrechson we send 1' or '1' = '1'--
- This query would be:
- SELECT * FROM admins WHERE username = '1' or '1' ='1'#-- ' AND password = 'secureP4ss'
- -- act as comment so we can disconsider everything that happens after this, so we get:
- SELECT * FROM admins WHERE username = '1' or '1' ='1'#-- what is true, so we can log into the system
- This is a classic attack of building malicious inputs that the programmer must not thought that someone would try

SQL (Structured Query Language)

- Well the database has a language specifically designed to work with it and query informations. So, if you want to check if there is a user called mark with password securep4ss in the database, you can build what we call query and ask for this program to check with the database if some information exists, or it's value, change, delete information, or whatever you want to
- So when you send your login and password the system look into its system and look for credentials like yours
- The result changes depending if it find your information, or if you're admin, etc
- This language changes slightly depending the database you're using to store your information
- Let's take a look of how this database works

Cenario 2

- Imagine we want to build a site, this site can be called http://3gcountryfarms.com
- The purpose of our website is to store recipes, and everyday you want to insert some recipes to your website
- Make static pages is not a good way of build this website, since news will be add everyday
- So, to build this site, the programmer created a database, and basically this website works like this: when you request the site, the server open it's database and replace blank data in the php field with the requested information
- Also, there isn't a page for each recipe, instead there is one page that receive GET parameter id and return the database recipe that matches with this id

How to exploit?

Exploitation

- After the ID we can insert which SQL statement we want to
- Imagine that this database get the query:

```
SELECT * FROM recipes WHERE id = ' + id
```

- Some SQL statements allow us to overlap tables, so we can query others tables, not only recipes, for example... the table admin
- So we can ask for

```
recipes.php?id=NULL UNION ALL SELECT 1,username,3,password,5,6 FROM admin--
```

- And get the admin credentials :)

Challenges

WebGoat

- As you saw, WebGoat come with a lot of exercises and explanations of SQL Injection, do lessons from starting from SQL Injection Mitigation, to SQL Injection and to end SQL Injection Advanced :)
- If you have some trouble feel free to discuss this things in P.O.M.B.O telegram group

More information:

- Matheus Vrech
- Telegram e Twitter: @vrechson
- Email: abrasax@cocaine.ninja

POMBO:

- Grupo do Telegram: @pomboufscar
- Canal do Telegram: @pombocorreio
- Github: https://github.com/pombo-ctf