

SQL Injection - OWASP Mutillidae II

1. Tentar logar com usuário aleatório, exemplo 'sgoncalves' senha 'senha'
2. Observar mensagem de erro que informa a inexistência deste usuário, logo, a mensagem pode nos levar a conhecer os usuários existentes.
3. Tentar com o usuário 'admin' e senha 'senha'
4. Verificar que o erro informa apenas a senha errada, logo, o usuário admin existe.
5. Para forçar um erro de SQL usamos ' no campo de senha, e vejamos as mensagens que surgem.
6. Diante das mensagens, conseguimos identificar exatamente a consulta SQL realizada pela aplicação ao BD.
7. Alteremos a consulta sql, para isso vamos adicionar uma condição ao campo de senha, por meio do código: ' or '9'='9
8. Ao inserir qualquer usuário no campo referente, e o código SQL no campo senha, temos acesso ao sistema.
9. Agora, voltando a aplicação Multillidae, vamos explorar mais vulnerabilidades nesta aplicação por meio do SQL Injection. Clique em OWASP 2013 > A1 - Injection (SQL) > SQLi Extract Data > User Info (SQL)
10. Nesta tela, vamos explorar os dados do BD por meio de queries SQL. Vamos começar tentando descobrir o número de colunas desta tabela. Para isso, utilizaremos a query 'order by'.
11. Podemos pedir para que a tabela accounts ordene pelas colunas 1 e 2 através do statement: admin' order by 1,2 --[espaço] esta consulta retornará resultado, logo, temos pelo menos duas colunas no sistema.
12. Podemos pedir para que a tabela accounts ordene pela coluna 50, através do statement: admin' order by 50 --[espaço] esta consulta retornará erro, logo, temos entre 2 e 50 colunas.

13. Vamos aumentar o número de colunas gradativamente, para descobrir o número real de colunas que a tabela possui.
14. Tendo descoberto o número de colunas, podemos seguir com os testes e verificar as informações contidas nestas colunas. Para isso, utilizaremos uma tabela interna do MySQL chamada `information_schema`. A query que precisaríamos informar, seria: `SELECT * from information_schema.columns where table_name='accounts'` e nós sabemos que a página já requisita a query `SELECT username from accounts where username= '' AND password = ''`.
15. Para consultar o que pretendemos, vamos utilizar a função `UNION` do SQL. Colocando no campo `username` o statement: `admin' UNION select 1,database(),3,4,5,6,7 from information_schema.columns where table_name='accounts' --[espaço]`
16. Agora que sabemos o nome do Banco de Dados, vamos consultar as informações das colunas presentes na tabela `accounts`. Para isso, vamos usar o statement: `admin' UNION select 1,column_name,3,4,5,6,7 from information_schema.columns where table_name='accounts' and table_schema='nowasp' --[espaço]`
17. Vamos fazer os mesmos testes feitos até aqui na aplicação DVWA? Faça na próxima atividade!
18. Nós não precisamos verificar as vulnerabilidades manualmente, podemos utilizar algumas ferramentas pra isso, nesta aula usaremos o `SQLMAP`. Para isso, a sintaxe é: `sqlmap -u "url_alvo"`. A ferramenta percorrerá a URL informada e nos avisará se houver vulnerabilidades.
19. Descobrimos que a página possui vulnerabilidades, bem como descobrimos que ela usa MySQL. Agora, vamos descobrir o BD atual utilizado. Adicione ao comando anterior a flag: `--current-db`
20. Descobrimos o nome do banco de dados! Agora, vamos ver se existem mais tabelas neste banco. Adicione a flag: `--tables -D nomedobd`
21. Descobrimos uma tabela chamada `users`. Agora, imagine essa vulnerabilidade em um sistema que você criou ou administra? Um hacker gostaria de ter os

dados desta tabela, não é mesmo? Ele utilizaria a função 'dump'. Vamos utilizar, acrescente a flag --dump -T users -D nomedobd ao comando anterior.

22. Pronto... Acessamos os dados de usuários do sistema! E ainda conseguimos quebrar os hashes... Vejam o quanto esta vulnerabilidade é séria!

23. **Prevenção:** Como evitar este ataque, no desenvolvimento da aplicação? Vamos ver como seria em JavaScript. Bastaria que nós separássemos os códigos da query SQL. Para isso, em JavaScript utilizaremos:

```
PreparedStatement stmt = connection.prepareStatement(sql);  
    stmt.setString(1,usuario);  
    stmt.setString(2,senha);  
    stmt.execute();
```