ETAPA 9: Responda as perguntas abaixo, enviando um arquivo com suas respostas em PDF. No arquivo de respostas deverá ter as perguntas também, não somente as respostas:

Considere o seguinte Exemplo do quadro abaixo:

Para exemplificar o funcionamento da injeção de SQL, consideremos o comando básico de consuta abaixo, a instrução SQL query:

```
SELECT id, nome, sobrenome FROM autores WHERE nome = 'josé' AND sobrenome
= 'silva';
```

Com base nesta instrução, é fácil supor que os itens "josé" e "silva" são do tipo texto (strings), solicitados por algum usuário que esteja usando a aplicação.

Problema

Portanto, supondo que a aplicação não faça o entendimento apropriado do conteúdo inserido pelo usuário, o mesmo pode fazer o uso acidental do caractere apóstrofo. Gerando a entrada:

- nome = *jo'sé*
- sobrenome = *silva*

E fazendo com que a aplicação gere o código:

```
SELECT id, nome, sobrenome FROM autores WHERE nome = 'jo'sé' AND
sobrenome = 'silva';
```

De acordo com a especificação da linguagem SQL, o uso de apóstrofo na consulta causa uma quebra na consulta, ocorrendo um erro de sintaxe nessa instrução, a string será considerada no campo **nome** apenas a palavra "jo" (dentro da primeira dupla de apóstrofo 'texto'). O interpretador do SQL espera que a continuação da instrução sejam outros comandos SQL válidos que completam a instrução principal. No entanto, como a outra parte do texto, o "sé" não é um identificador válido, essa instrução não será executada e retornará um erro inesperado.

Ataque

Assim, um atacante pode personalizar os dados de entrada a fim de gerar um comportamento inesperado na base de dados. Para exemplificar este conceito, consideremos na consulta apresentada, a entrada dos seguintes dados através da aplicação:

- nome = *jo'; DROP TABLE autores ; --*
- sobrenome = *silva*

A instrução completa ficaria:

```
SELECT id, nome, sobrenome FROM autores WHERE nome = 'jo'; DROP TABLE
autores ; --' AND sobrenome = 'silva';
```

A instrução personalizada funcionará da seguinte forma:

```
SELECIONE (select) todos os "ids", "nomes" e "sobrenome" DA TABELA (from)
"autores" (nome da tabela) ONDE (where) os nomes deverão ser iguais a
'josé'; (quebra, novo comando) Em seguida EXCLUA (drop) a tabela
"autores"; -- (continuação) E os sobrenomes iguais a 'silva' (condições
do filtro);
```

Neste caso, a instrução será executada normalmente, pois a adição do caractere ponto-e-vírgula ";" na instrução representa o fim de uma SQL query e o começo de outra. Assim no exemplo acima, a SQL query será reconhecida como completa - não ocorrendo erro de sintaxe - de modo prematuro dando espaço para uma nova instrução. de livre escolha do atacante. Podendo retornar dados confidenciais armazenados na base de dados ou de executar instruções que comprometam o sistema, como a remoção de dados e/ou tabelas, como apresentado no exemplo acima.

A seqüência de caracteres "--" representa um comentário em uma linha de SQL. O "--" no fim do campo username é obrigatório para que a SQL query continue sendo executada sem erros.

Pergunta 1: Explique com suas palavras qual o papel do caractere Apostrofo (‘) na parte grifada em amarela abaixo, do Ponto e Virgula (;) na parte grifada em amarelo e do caractere traço traço (--) parte grifada em amarelo, no exemplo dado abaixo, que representa um ataque de SQL INJECTION:

```
SELECT id, nome, sobrenome FROM autores WHERE nome =
'jo'; DROP TABLE autores ; --' AND sobrenome = 'silva';
```

RESPOSTA 1:

O apostrofo é utilizado para quebrar a estrutura de consulta o código fonte, para que então se adicione mais condições a mesma consulta ou então que se finalize a consulta principal. O ponto e virgula é utilizado para que se termine a consulta original e então acrescente uma nova consulta maliciosa. E os dois traços são utilizados para que se comente o restante da consulta, invalidando o que vem depois dele.

PERGUNTA 2: Considerando a tela de Login abaixo, e considerando que a mesma não tem proteção alguma contra SQL Injection. Explique como um atacantes poderia se aproveitar dessa vulnerabilidade, para realizar o acesso sem saber o usuario ou a senha:

RESPOSTA 2:

Um atacante pode explorar a ausência de proteção contra SQL Injection inserindo comandos maliciosos, como `' OR '1'='1`, no campo de login, dessa forma, manipulando a consulta SQL para que sempre se retorne verdadeiro. Assim, ele obtém acesso ao sistema sem precisar de credenciais válidas, autenticando-se como o primeiro usuário da tabela, geralmente um administrador.