|  |  |
| --- | --- |
| ‘\Desenho de rosto de pessoa visto de perto  Descrição gerada automaticamente com confiança média | **Universidade Tecnológica Federal do Paraná**  **Câmpus Apucarana**  **Graduação em Engenharia de Computação** |
| Disciplina Desenvolvimento de Aplicações Web  *Professor Wendel Góes* |

**Objetivo da atividade**: Simular a injeção de código SQL em um formulário HTML (ataque SQL Injection).

* Injeção de SQL é um tipo de ataque utilizado por usuários maliciosos para injetar código SQL ilícito dentro de uma instrução SQL lícita, podendo causar danos no banco de dados ligados a aplicação WEB.
* Geralmente utiliza campos de formulário ou a URL.
* Existe a possibilidade de o usuário malicioso realizar consultas, atualizações e até mesmo exclusões (DELETE ou DROP TABLE) no banco de dados, sem qualquer autorização.

**EXEMPLO 1: Vídeo do prof. Rodrigo** *(Instituto Federal de São Paulo)*

*https://youtu.be/Mu1xEzV2wCg*

**EXEMPLO 2: Exemplo a ser simulado nesta atividade de laboratório:**

Formulário de cadastro: **form\_cad\_aluno.html**

**Tabela

Descrição gerada automaticamente**

**ATIVIDADE – ETAPAS ABAIXO**

**ETAPA 1: Criar tabela no Banco de dados**

Abaixo tem-se o script para criação da Tabela no Banco de Dados e inserção de 2 alunos. Utilize a ferramenta gráfica PHP-MY-ADMIN do Xampp: <http://localhost/phpmyadmin/>

CREATE TABLE aluno

( nome varchar(52),

telefone varchar(52)

) ENGINE=InnoDB;

INSERT INTO aluno VALUES ("John", "abc");

INSERT INTO aluno VALUES ("Jimmy", "xyz");

**ETAPA 2: Crie pasta da aplicação e formulário HTML**

**- Crie uma pasta chamada:** app\_vulneravel\_sql\_injection **para inserção dos arquivos desta atividade. Esta pasta deverá estar dentro da pasta do Servidor Web:**

- c:/XAMPP/xamppfiles/htdocs/dashboard/ app\_vulneravel\_sql\_injection (windows)

-Macintosh\_hd/Applications/XAMPP/ xamppfiles/htdocs/dashboard/app\_vulneravel\_sql\_injection (apple ios)

**- /**opt/lampp/htdocs**/**dashboard**/** app\_vulneravel\_sql\_injection (Ubuntu Linux)

**- Criar um Formulário em HTML de nome: form\_cad\_aluno.html**

**com os campos Nome e Telefone com opção form action = cad\_aluno.php (*ver etapa 3)***

Uma imagem contendo Tabela

Descrição gerada automaticamente

**ETAPA 3: Criar o arquivo PHP cad\_aluno.php p/ tratar o formulário, conforme abaixo.**

<?php

// recuperando dados do formulario

$nome = $\_POST['nome'];

$telefone = $\_POST['telefone'];

// variáveis ref. A conexão ao bd

$servername = "localhost";

$username = "seu usuario bd";

$password = "sua senha";

$dbname = "nome da bd";

// criando a string de Conexao com o BD

$strcon = mysqli\_connect($servername, $username, $password, $dbname, 3306) or die('Erro ao conectar ao banco de dados');

// criação da Instrução SQL DML para inserção dos dados

$sql = "INSERT INTO aluno (nome, telefone) VALUES ('$nome', '$telefone')";

// execução da instrução SQL DML de Insert

mysqli\_query($strcon,$sql) or die("Erro ao tentar cadastrar no BD");

// fechamento da conexão com o bd.

mysqli\_close($strcon);

echo "aluno cadastrado com sucesso!";

?>

**ETAPA 4: Teste da aplicação SEM a inserção de código malicioso:**

1. Acesse o formulário form\_cad\_aluno.html
2. Insira um aluno (nome e telefone).
3. Verifique se o mesmo aparece no Banco de dados (ferramenta PHP-MY-ADMIN)

**ETAPA 5: Teste da aplicação Com a inserção de código malicioso:**

1. Acesse o formulário **form\_cad\_aluno.html**
2. Insira um nome de aluno qualquer, e no campo Telefone insira um código Malicioso, conforme exemplo abaixo:

Tabela

Descrição gerada automaticamente

1. Verifique se foi executado o código malicioso, ou seja, verificar se a tabela Aluno esta vazia (execução da clausula: delete from Aluno).
   * use a ferramenta PHP-MY-ADMIN para verificar o conteúdo da tabela. Se a tabela estiver vazia, está ok a simulação desta atividade (Etapa 5).

**ETAPA 6: Pesquisar formas de proteger o seu código de SQL Injection.**

* Pesquisar e implementar forma(s) de se proteger da injeção de SQL em formulários HTML, neste projeto desta atividade.
* Fazer uma cópia do projeto anterior (etapas 2 e 3), e fazer uma versão que tenha proteção de SQL Injection (implementar o código).
* Fazer comentários no código fonte (html ou php), referente aos locais no código que são responsáveis pela defesa/proteção contra Injeção de SQL.

**ETAPA 7: Forma de envio do código fonte.**

* Compactar o projeto vulnerável com o nome: app\_vulneravel\_sql\_injection.zip
* Compactar o projeto com proteção a SQL Injection com o nome: app\_protegido\_sql\_injection.zip
* Enviar os dois arquivos compactados no moodle desta atividade.

**ETAPA 8: Validação do Trabalho e apresentação.**

* Apresentar a execução dos dois projetos (vulnerável e não vulnerável) somente ao professor, durante a aula do dia **29/11** ou no dia **03/12**\_ (data máxima possível).

**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 consulta abaixo, a instrução [SQL](https://pt.wikipedia.org/wiki/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](https://pt.wikipedia.org/wiki/String)), 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](https://pt.wikipedia.org/wiki/Ap%C3%B3strofo). 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](https://pt.wikipedia.org/wiki/Linguagem) SQL, o uso de apóstrofo na consulta causa uma quebra na consulta, ocorrendo um erro de [sintaxe](https://pt.wikipedia.org/wiki/Sintaxe) nessa instrução, a string será considerada no campo **nome** apenas a palavra "jo" (dentro da primeira dupla de apóstrofo 'texto'). O [interpretador](https://pt.wikipedia.org/wiki/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](https://pt.wikipedia.org/wiki/Caractere) [ponto-e-vírgula](https://pt.wikipedia.org/wiki/Ponto-e-v%C3%ADrgula) ";" 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 sequência 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 apóstrofo ( ' ) é usado na consulta SQL original para delimitar o início e o fim de uma string. No ataque, o apóstrofo fecha prematuramente a string, terminando a consulta SQL original. O ponto e vírgula ( ; ) indica o fim de uma instrução SQL e o início de outra, permitindo ao atacante executar comandos adicionais, como DROP TABLE autores, que pode deletar uma tabela. O traço traço ( -- ) inicia um comentário, fazendo com que o resto da consulta seja ignorado, garantindo a execução do comando DROP TABLE autores sem erros de sintaxe.

**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:

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

RESPOSTA 2: Em uma tela de login sem proteção contra SQL Injection, um atacante pode inserir ' OR '1'='1 no campo de login e qualquer coisa no campo de senha. Isso transforma a consulta SQL original SELECT \* FROM usuarios WHERE login = 'usuario' AND senha = 'senha'; em SELECT \* FROM usuarios WHERE login = '' OR '1'='1' AND senha = 'qualquer\_coisa';. A condição OR '1'='1 é sempre verdadeira, retornando todos os registros da tabela **"usuarios"** e ignorando a validação da senha. Assim, o atacante consegue acessar a aplicação sem um login ou senha válidos.