Análise e Exploração de Vulnerabilidades em Aplicação Web Simulada

# Sumário

1. Objetivo

2. Ferramentas Utilizadas

3. Estrutura do Projeto

4. Vulnerabilidades Exploradas

5. Mitigações

6. Conclusão

7. Referências

# Objetivo

O projeto tem como objetivo simular uma aplicação web com falhas de segurança para explorar vulnerabilidades comuns   
 (como SQL Injection, XSS, CSRF, etc.), identificando e propondo soluções para mitigação dessas falhas.

# Ferramentas Utilizadas

- Flask (Python): Para desenvolvimento da aplicação web.  
 - SQLite: Banco de dados simples utilizado para armazenar dados dos usuários.  
 - Burp Suite, OWASP ZAP: Ferramentas de análise de segurança.  
 - Metasploit, Nikto: Ferramentas para exploração de vulnerabilidades e análise de servidores.

# Estrutura do Projeto

A estrutura de diretórios do projeto é a seguinte:  
 /flask-vuln-app/  
 - app.py: Código principal da aplicação.  
 - templates/: Arquivos HTML (index.html, login.html).  
 - static/: Arquivo CSS (style.css).  
  
 O código da aplicação realiza as seguintes funções:  
 - app.py: Lógica de aplicação, como login, criação de banco de dados e manipulação de sessões.  
 - templates/: Contém as páginas HTML.  
 - static/: Contém o CSS.

# Vulnerabilidades Exploradas

SQL Injection

SQL Injection é uma técnica onde o atacante pode manipular consultas SQL por meio de dados inseridos no sistema.   
 A aplicação é vulnerável a esse tipo de ataque, permitindo que um invasor acesse dados sensíveis de usuários. Exemplo de injeção SQL:  
 ' OR 1=1 --

XSS (Cross-Site Scripting)

XSS ocorre quando o sistema permite que o atacante insira scripts maliciosos em um site.   
 Exemplo de injeção XSS simples:  
 <script>alert('XSS')</script>  
 Isso pode ser explorado para roubar dados ou executar ações maliciosas no navegador do usuário.

Autenticação Fraca

A aplicação utiliza autenticação fraca sem criptografia de senhas. Isso permite que atacantes consigam comprometer   
 contas de usuários facilmente, como em ataques de brute force ou sessão sequestrada.

# Mitigações

SQL Injection

Usar consultas parametrizadas (prepared statements) para evitar manipulação de SQL.  
 Exemplo: usando placeholders em vez de concatenar dados diretamente na consulta.

XSS

Escapar as entradas de usuário para evitar que scripts sejam executados. Uma abordagem comum é usar funções como html.escape().

Autenticação

Implementar criptografia de senhas utilizando bibliotecas como bcrypt e usar sessões seguras (cookies HttpOnly ou tokens JWT).

# Conclusão

O projeto demonstrou como falhas de segurança podem ser exploradas em uma aplicação web. A mitigação de vulnerabilidades é essencial   
 para garantir a integridade, confidencialidade e segurança dos dados. O aprendizado obtido pode ser aplicado em práticas de desenvolvimento seguro.

# Referências

- OWASP: https://owasp.org/  
 - Burp Suite: https://portswigger.net/burp  
 - Metasploit: https://www.metasploit.com/