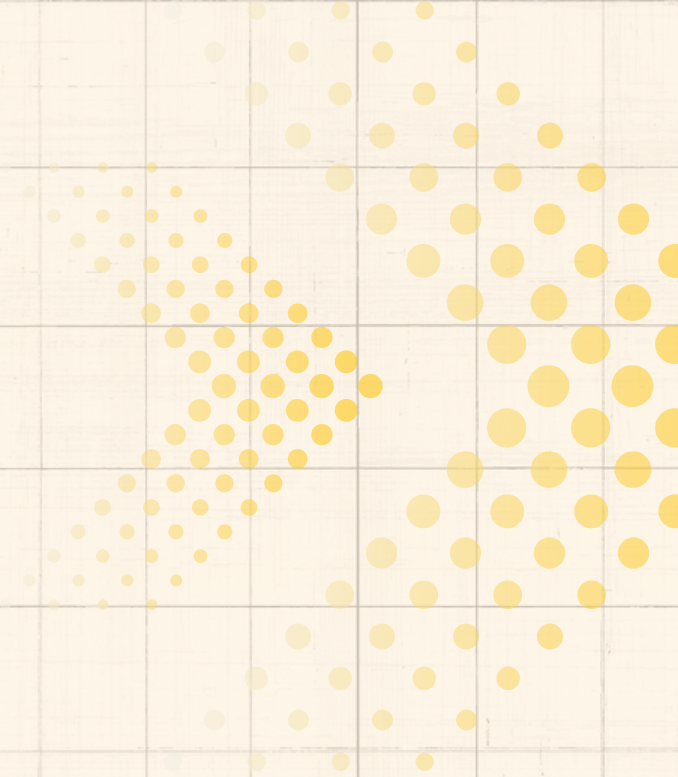


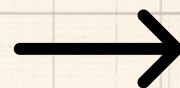


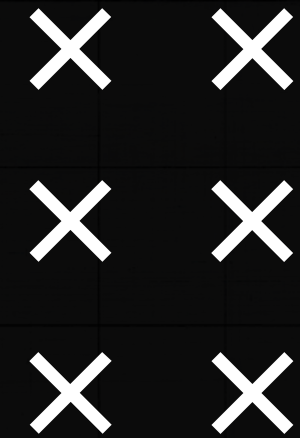
SQL Injection

O que é?

Como se proteger?

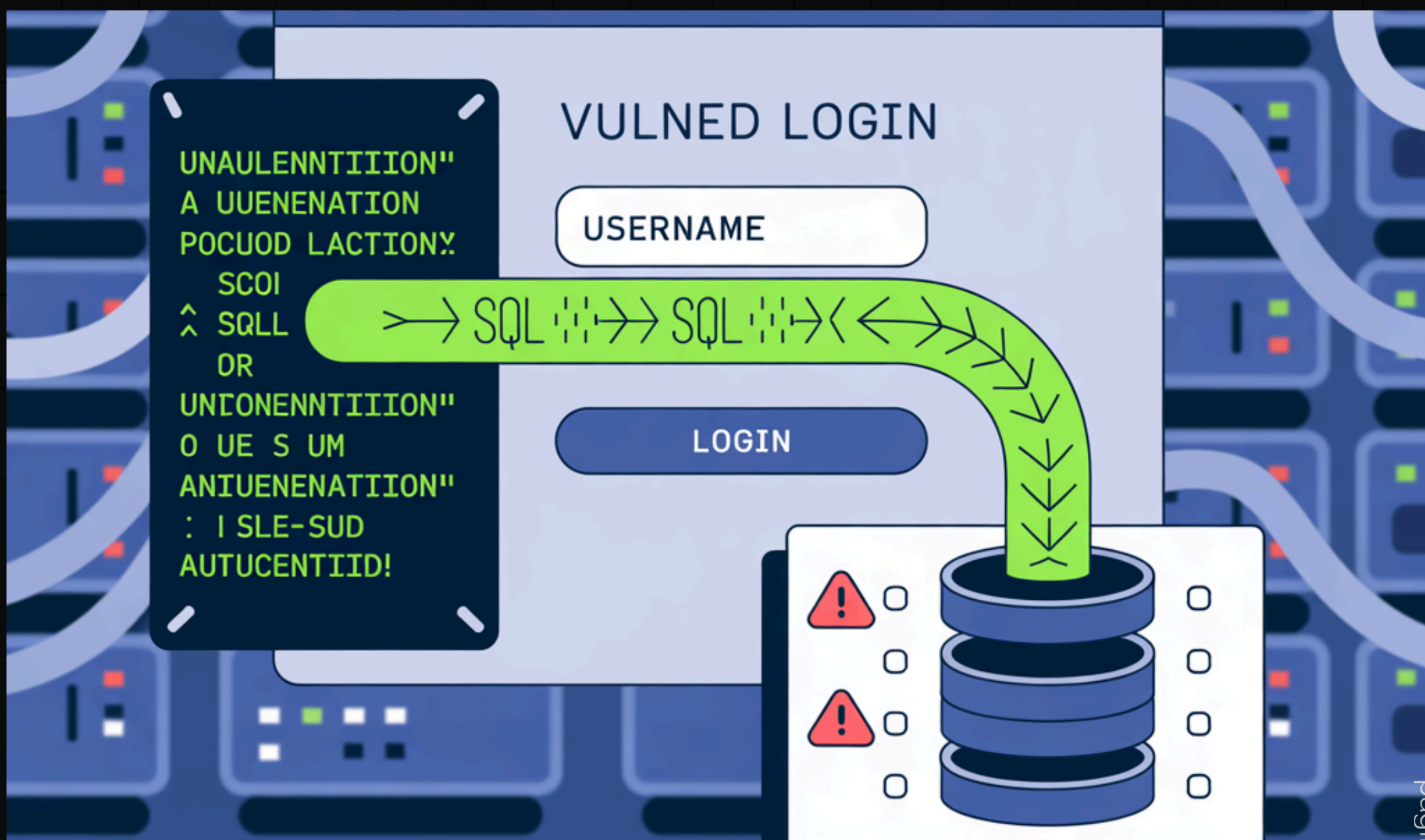
Como uma falha pode abrir portas para hackers

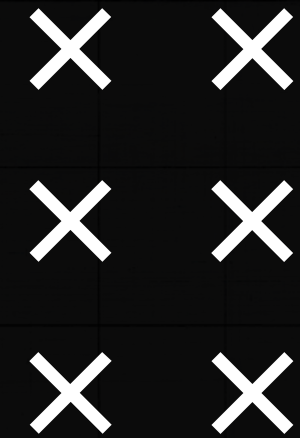




1. O Ataque que Pode Comprometer sua Aplicação

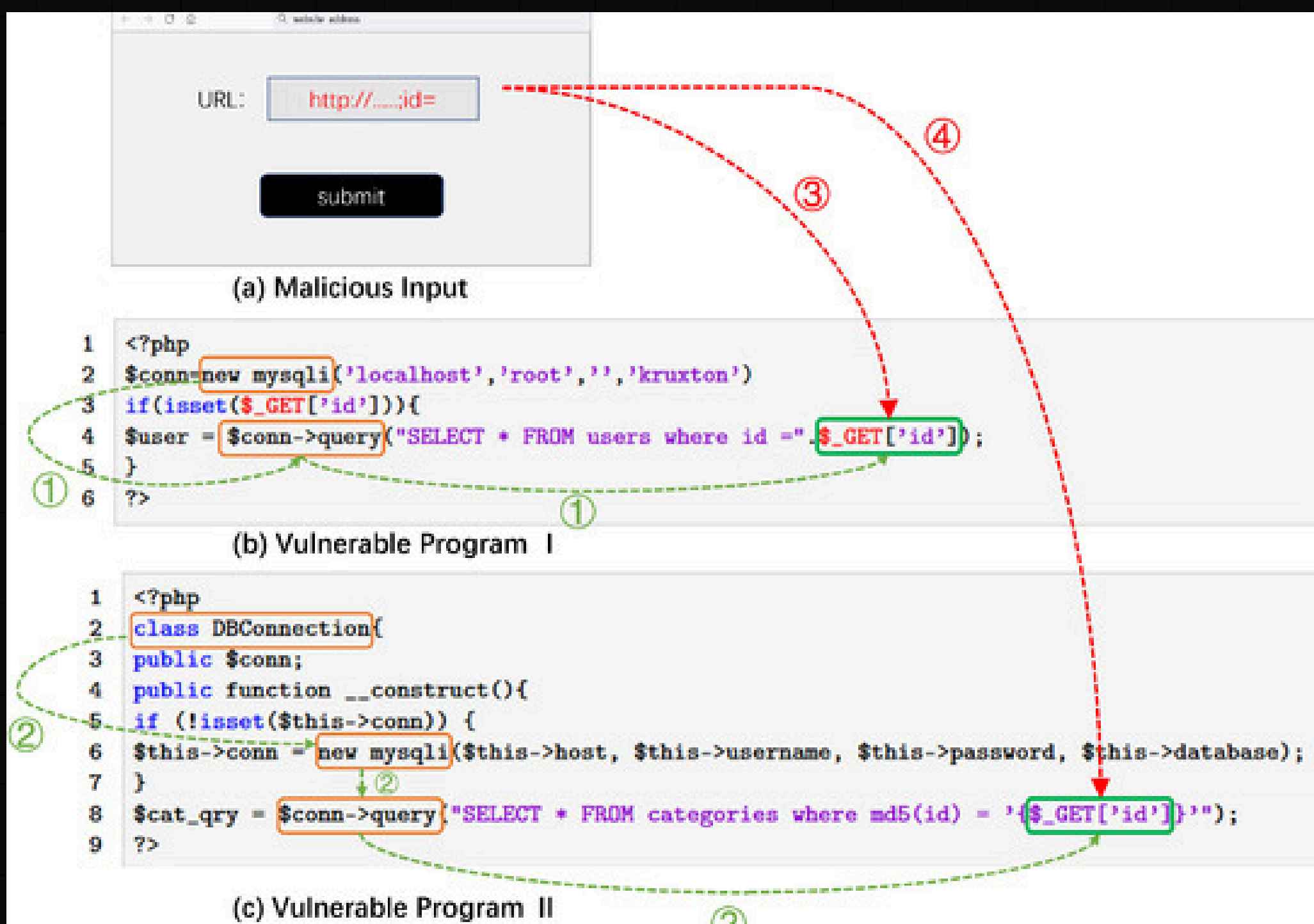
Em 2025, ainda existem muitos desenvolvedores que ignoram a gravidade do SQL Injection. Se você trabalha com desenvolvimento web ou APIs, este conteúdo é essencial!

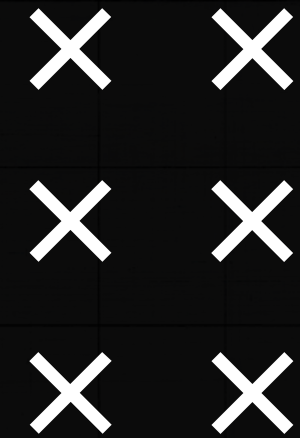




2. O que é SQL Injection?

SQL Injection é um ataque em que comandos maliciosos são inseridos em campos de entrada de dados para manipular consultas SQL e acessar, modificar ou excluir dados sensíveis.





3. Como Funciona o SQL Injection?

Imagine um formulário de login com uma consulta SQL vulnerável:



```
1 SELECT * FROM usuarios WHERE email = 'usuario@exemplo.com' AND senha = 'senha123';  
2
```

Se um invasor inserir algo assim no campo de senha



```
1 ' OR '1'='1'  
2
```

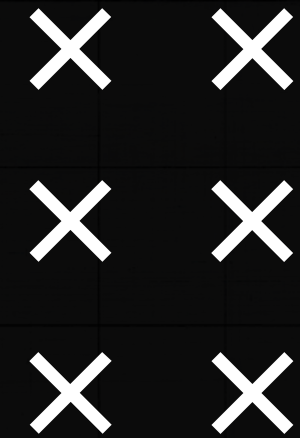
Resultado: A condição '1'='1' sempre é verdadeira, permitindo o acesso não autorizado. 

A consulta se torna:



```
1 SELECT * FROM usuarios WHERE email = 'usuario@exemplo.com' AND senha = '' OR '1'='1';  
2
```

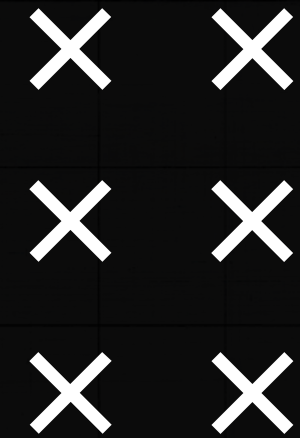




4. Principais Riscos do SQL Injection

- Exposição de dados sensíveis
- Alteração ou exclusão de registros
- Execução de comandos administrativos
- Comprometimento total do servidor

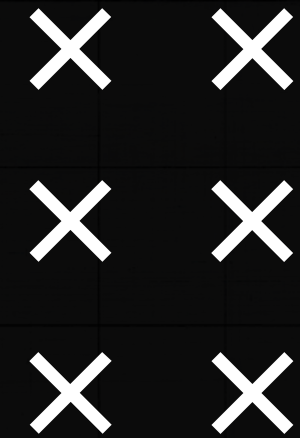




5. Tipos de SQL Injection

- In-band: O atacante usa o mesmo canal para injetar e coletar dados.
- Inferencial (Blind): O atacante usa o comportamento da aplicação (como tempo de resposta) para entender o banco de dados.
- Out-of-band: Usa canais diferentes para o ataque e coleta dos dados.

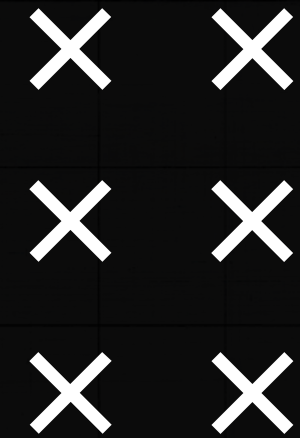




6 Como se Proteger?

- Use queries parametrizadas (Prepared Statements)
- Valide e sanitize todos os inputs
- Utilize ORMs seguros como Prisma ou Sequelize
- Configure corretamente os privilégios do banco de dados
- Monitore tentativas de acesso suspeitas





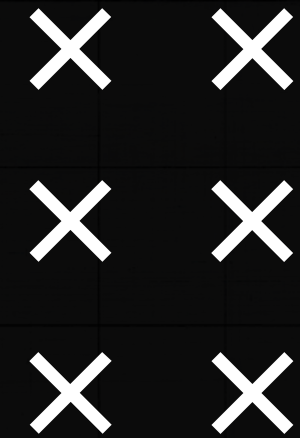
7 Exemplo de Proteção Prepared Statements



```
1 const query = 'SELECT * FROM usuarios WHERE email = $1 AND senha = $2';  
2 const result = await pool.query(query, [email, senha]);  
3
```

A utilização de placeholders impede a manipulação maliciosa da consulta.

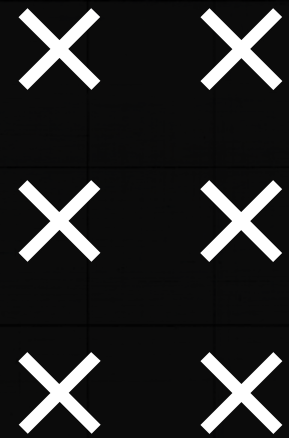




8. Ferramentas de Segurança

- OWASP ZAP: Scanner de segurança para aplicações web
- SQLMap: Ferramenta para detectar e explorar SQL Injection
- Burp Suite: Plataforma para testes de segurança web

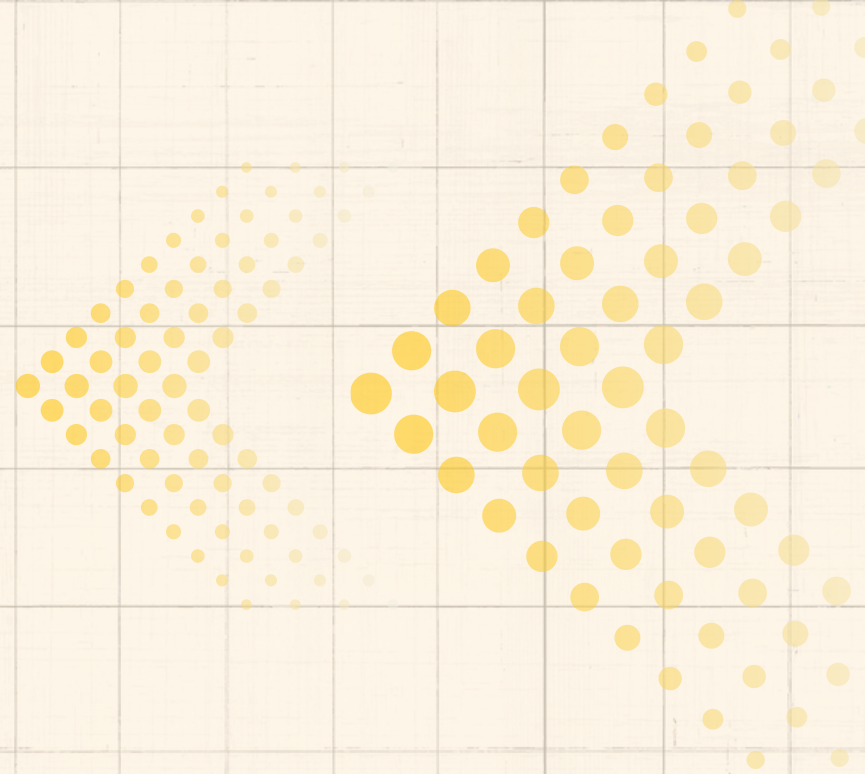
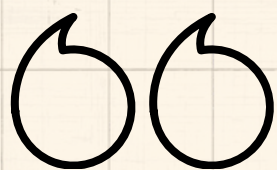




9. Conclusão

SQL Injection é uma das vulnerabilidades mais perigosas, mas pode ser evitada com boas práticas de segurança. Invista em proteção desde o início e blinde sua aplicação!





Você utiliza Prepared Statements?

**Ou tem alguma outra
estratégia que você
utiliza e não apareceu
aqui?**

Me conta nos comentários! ↓

