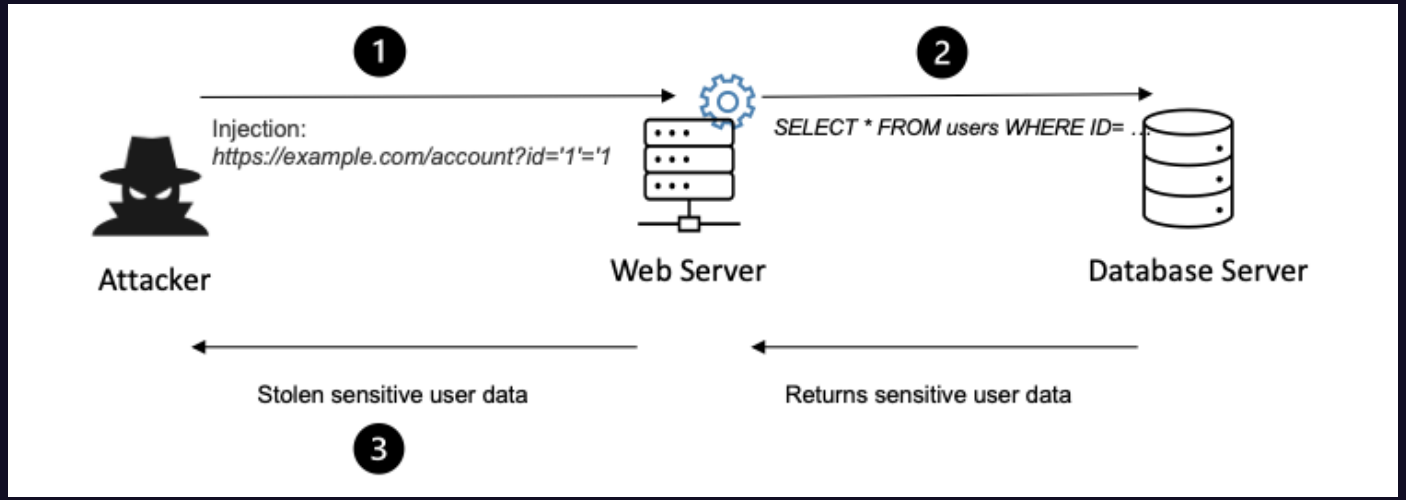

INJECTION

INTRODUÇÃO

Segundo a OWASP, Injeção é a tentativa de um invasor de enviar dados para uma aplicação de forma a alterar o significado dos comandos enviados a um interpretador.



Principais tipos de Injection

01 SQL Injection

02 Command Injection

03 LDAP Injection

04 noSQL Injection

01

SQL Injection

Manipula consultas de SQL

```sql
SELECT * FROM usuarios WHERE nome = 'usuarioDigitado' AND senha = 'senhaDigitada';
```

```sql
SELECT * FROM usuarios WHERE nome = 'João' AND senha = '123';
```

```sql
' OR '1'='1
```

```sql
SELECT * FROM usuarios WHERE nome = '' OR '1'='1' AND senha = '';
```

02 Command Injection

Executa comandos no sistemas operacional

```
ping 127.0.0.1
```

```
ping 127.0.0.1; apagar_tudo
```

03 LDAP Injection

Exploração de consultas LDAP (Diretórios corporativos)

```
usuário = joao
senha = 123
```

```
*)(|(uid=*))
```

04 noSQL Injection

Manipulação de banco de dados NoSQL como MongoDB

```
usuario = joao
senha = 123
```

```
{ "$ne": null }
```

Impactos Típicos

- ❑ Roubos de dados sensiveis
- ❑ Acesso não autorizado
- ❑ Controle total do sistema
- ❑ Alteração ou exclusão de informações
- ❑ Prejuizos financeiros e reputacionais

DETECÇÃO DE INJECTIONS

Principais Formas

01

Revisão de Código

- Análise do Código Fonte manualmente

02

Testes de Segurança

- Simulação de ataques contra a aplicação

03

Softwares Especializados

- Ferramentas que detectam vulnerabilidades

Ferramentas de Análise Estática - SAST

O que fazem: Analisam o Código Fonte sem executar o sistema

Como: Procuram trechos de código suspeitos, como concatenação de strings em queries SQL.

Exemplo: Fortify – analisa padrões inseguros em linguagens diversas.

Ferramentas de Análise Dinâmica - DAST

O que fazem: Simulam ataques enquanto a aplicação está rodando

Como: tentam injetar comandos maliciosos nos formulários, URLs ou parâmetros da aplicação

Exemplo: Burp Suite – automatiza ataques e detecta pontos fracos

Ferramentas de Análise Interativa - IAST

O que fazem: Analisam a aplicação enquanto ela roda

Como: monitoram o código em execução e identificam quando um comando suspeito

Exemplo: Contrast Security – insere agentes dentro da aplicação para monitorar as execuções.

PREVENÇÃO E BOAS PRÁTICAS

Como Mitigar Falhas de Injection

- Evitar construir queries concatenando dados de entrada do usuário
- Utilizar Prepared Statements (declarações parametrizadas)
- Restringir permissões de usuários no banco de dados
- Monitorar logs e atividades suspeitas

Prepared Statements / Parametrização

- Separação entre código SQL e dados do usuário
- Impede que o banco interprete entrada maliciosa como comando
- Disponível em linguagens como PHP (PDO), Java (JDBC), Python (SQLite, psycopg2)

Todas as linguagens modernas oferecem prepared statements.
A forma de escrever muda, mas a ideia é sempre a mesma:
Não misturar código SQL, com dados vindos do usuário.

Exemplo em PHP: Query vulnerável vs Prepared Statement (PDO)

- Vulnerável

```
$query = “SELECT * FROM produtos WHERE nome=“”.$_POST[‘produto’].””;
```

- Seguro

```
$stmt = $pdo->prepare(“SELECT * FROM produtos WHERE nome=:produto”);
$stmt->bindParam(‘:produto’,$_POST[‘produto’],PDO::PARAM_STR);
$stmt->execute( );
```

Exemplo em PHP: Query vulnerável vs Prepared Statement (PDO)

- O statement impede que os dados fornecidos pelo usuário sejam interpretado como código SQL
- Isso bloqueia o código malicioso (ex: OR 1=1, DROP TABLE, etc.) porque o banco só aceita o valor como parâmetro

Boas Práticas

- Usar ORMs (Ex: Hibernate, Eloquent, SQLAlchemy)
- Ativar WAF (Web Application Firewall)
- Seguir recomendações da OWASP
- Realizar testes de segurança periódicos
- Nunca confiar em dados vindos do cliente

Conclusão

- Injection é uma das vulnerabilidades mais graves segundo a OWASP Top 10
- Prevenção depende principalmente de código seguro
- Boas práticas + ferramentas = maior proteção contra ataques

OBRIGADO!