

# **GARANTINDO A SEGURANÇA DE APLICAÇÕES WEB CONTRA SQL INJECTIONS**

**Arilo Claudio Dias Neto (ExperTS)**

Coordenador do Grupo ExperTS - [ariloclaudio@gmail.com](mailto:ariloclaudio@gmail.com)

**Renata Magalhães Rêgo (ExperTS)**

Mestranda em Informática - [renatamagalhaesrego@gmail.com](mailto:renatamagalhaesrego@gmail.com)



# AGENDA

---

1. Principais Conceitos
2. Tipos de Vulnerabilidades
3. SQL Injection.
  - 3.1 Técnicas prevenção
  - 3.2 Ferramentas prevenção
  - 3.3 Técnicas para contornar problemas já detectados em aplicações.

# O QUE É SEGURANÇA?

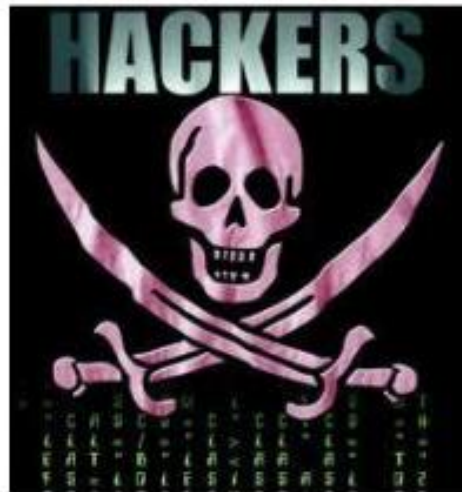
**Segurança** é a percepção de se estar protegido de riscos, perigos ou perdas. (Dicionário)

**Segurança Pública** é sistema integrado e otimizado envolvendo instrumento de prevenção, coação, justiça, defesa dos direitos, saúde e social. Se inicia pela prevenção e finda na reparação do dano. (Wikipedia)

Em software, **Segurança** mede a capacidade do sistema de proteger as informações do usuário e fornecê-las apenas (e sempre) às pessoas autorizadas. (ISO 9126)

# O QUE É SEGURANÇA?

- Segurança de software é um assunto complexo que deve ser tratado antes mesmo de se pensar em testes



# O QUE É SEGURANÇA DA INFORMAÇÃO?

---

- **Proteção de um conjunto de dados**
  - No sentido de preservar o seu valor para um indivíduo/organização.
- **Características básicas:**
  - Confidencialidade
  - Integridade
  - Disponibilidade
  - Autenticidade:
- **O conceito se aplica a todos os aspectos de proteção de informações e dados e também a sistemas.**

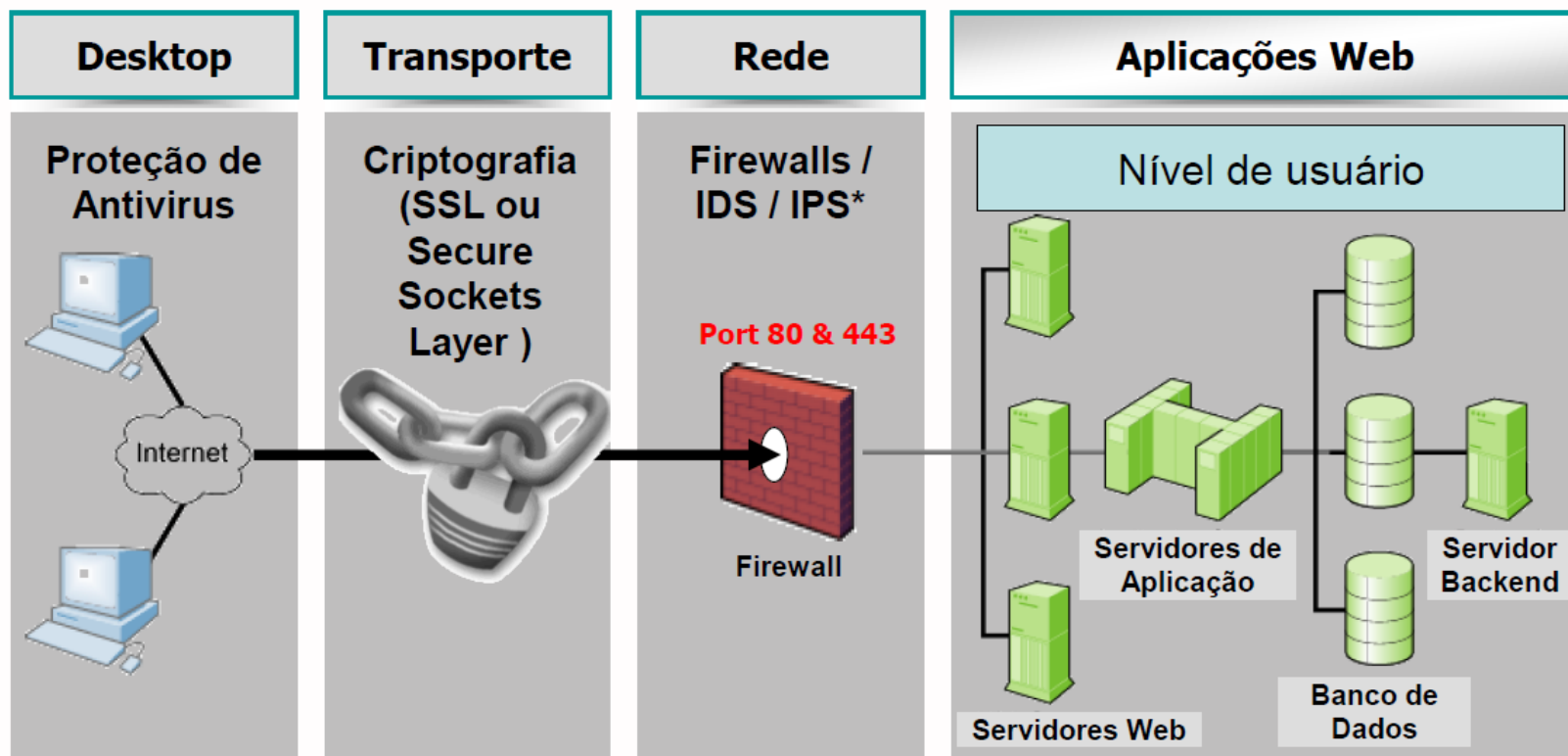
# O QUE É SEGURANÇA

---

- **Software são alvos atrativos para invasores (hackers), empregados insatisfeitos, competidores desonestos e qualquer outra pessoa que deseje...**
  - **Roubar informação confidencial**
  - **Modificar conteúdo com má intenção**
  - **Degradar desempenho**
  - **Desmontar funcionalidades**
  - **Embaraçar uma pessoa, organização ou negócio**

# CONHECENDO O PROBLEMA

## Cenário de Informação de Segurança



\* Sistema de detecção de intrusão ou o sistema de prevenção de intrusões



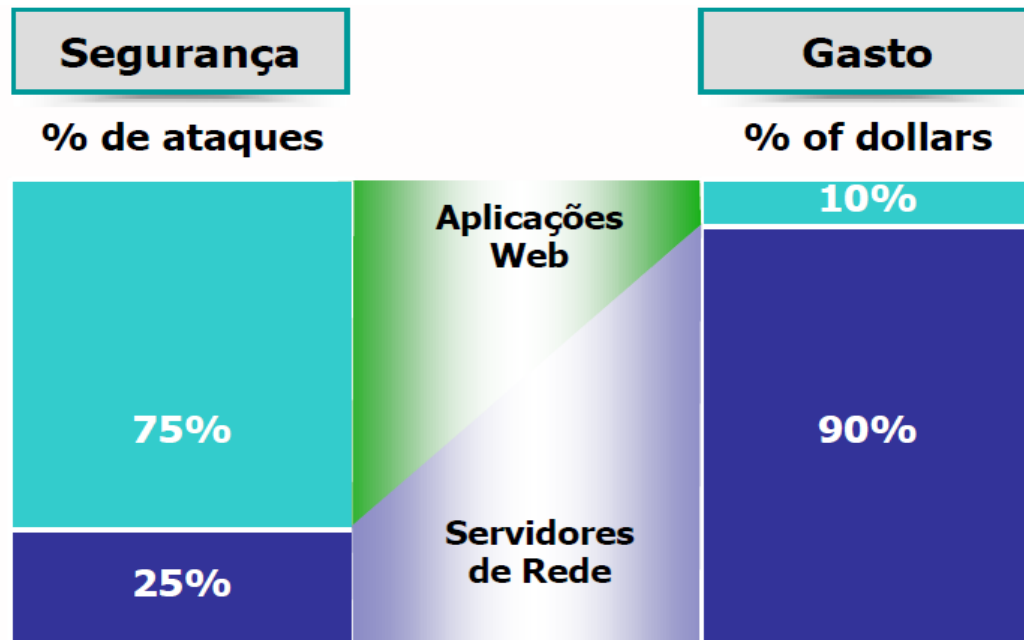
# CONHECENDO O PROBLEMA

---

- O Mito: ***“Nosso site é seguro”***
- ***“Temos firewalls no lugar”***
  - Portas 80 e 443 estão abertas por razões de direito
- ***“Nós criptografamos os dados com SSL”***
  - Isso só protege os dados entre site e usuários, não o próprio aplicativo Web



# CONHECENDO O PROBLEMA



**75%** De todos os ataques nas informações de segurança são direcionados para a camada de aplicação Web

**2/3** De todas as aplicações Web são vulneráveis

# SEGURANÇA DE APLICAÇÕES WEB: O QUE PODE ACONTECER?

- **Vazamento de dados confidenciais**
  - Clientes, Parceiros ou dados da empresa



# SEGURANÇA DE APLICAÇÕES WEB: O QUE PODE ACONTECER?

- Roubo de Identidade
  - Hacker passa por um usuário confiável



# SEGURANÇA DE APLICAÇÕES WEB: O QUE PODE ACONTECER?

- **Desfiguração: alteração de conteúdo**
  - Fere a marca, engana clientes e assim por diante



# SEGURANÇA DE APLICAÇÕES

## WEB: O QUE PODE ACONTECER?

---

- **Desligamento de Aplicações (indisponível)**
  - Falta de acesso pode causar perdas graves





# SEGURANÇA DE APLICAÇÕES WEB: O QUE PODE ACONTECER?

- **Execução Remota**
  - Executar código arbitrário no servidor



# CUSTOS DE UMA FALHA NA SEGURANÇA

---

- **Atenção da Mídia/Danos a marca**
- **Acompanhamento dos custos de serviços**
- **Honorários legais**
- **Penalidades de órgãos de regulamentação**
- **Auditorias**
- **Novos gastos de segurança**
- **Processos de Clientes**
- **Perda de clientes**

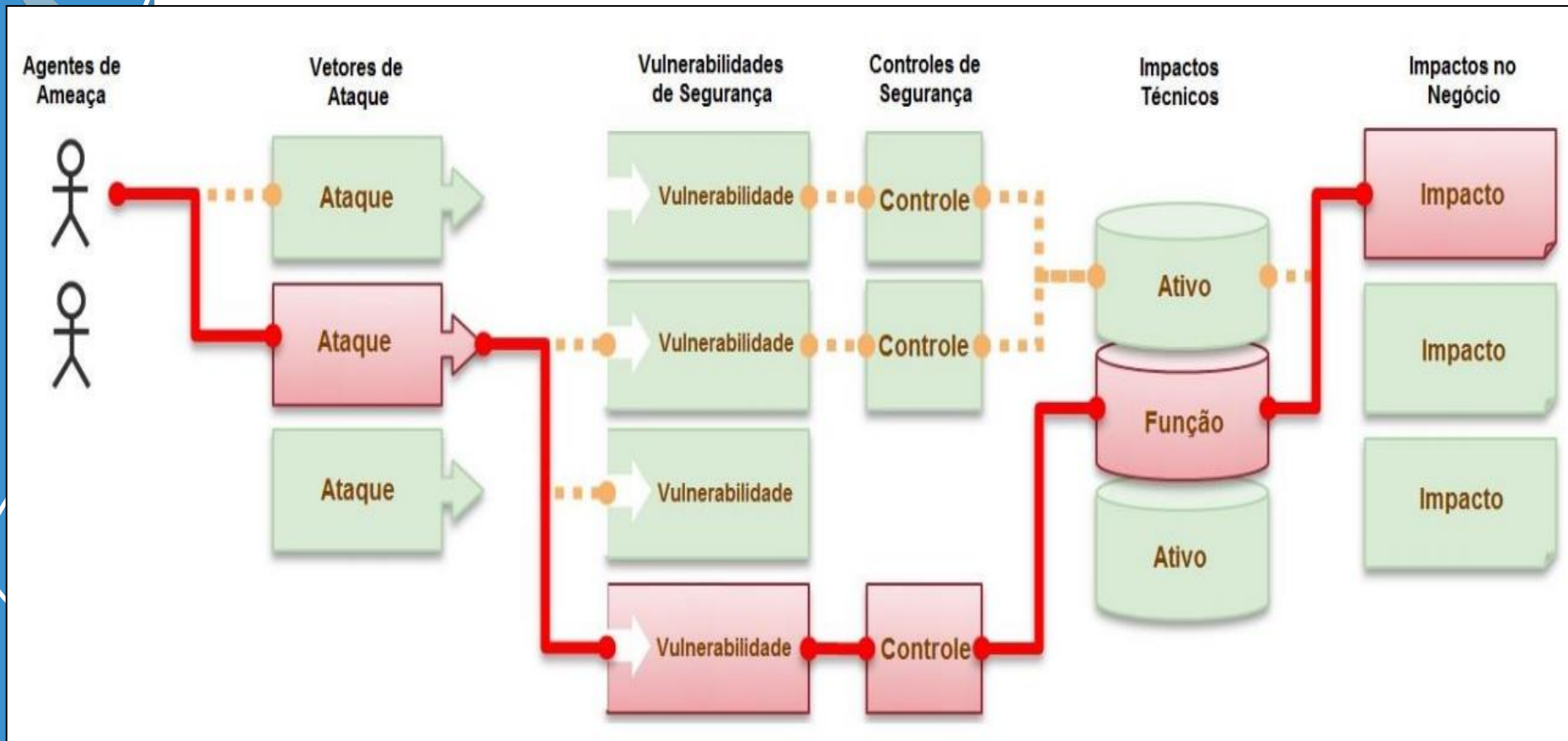


# A REALIDADE

---

- **Dois terços de todas as aplicações Web tem algum tipo de vulnerabilidade**
- **Existem várias organizações que tentam ajudar**
  - OWASP – Open Web Application Security Project
  - [https://www.owasp.org/index.php/Main\\_Page](https://www.owasp.org/index.php/Main_Page)
  - O projeto OWASP Top Ten
  - [http://owasptop10.googlecode.com/files/OWASP\\_Top\\_10\\_-\\_2013\\_Brazilian\\_Portuguese.pdf](http://owasptop10.googlecode.com/files/OWASP_Top_10_-_2013_Brazilian_Portuguese.pdf)

# O QUE SÃO RISCOS DE SEGURANÇA EM APLICAÇÕES?



# OWASP Top 10 - 2013

## A1 - Injeção

- Dados não confiáveis são enviados para um interpretador como parte de um comando ou consulta.

## A2 – Quebra de Autenticação e gerenciamento de Sessão

- Falhas de implementação comprometem senhas, chaves e tokens de sessão.

## A3 – Cross-Site Scripting (XSS)

- scripts são executados no navegador e podem “sequestrar” sessões do usuário, desfigurar sites, ou redirecionar o usuário para sites maliciosos

# OWASP Top 10 - 2013

## A4 – Referência Insegura e Direta a Objetos

- Exposição de uma referência à implementação interna de um objeto, como um arquivo, diretório, ou registro da base de dados.

## A5 – Configuração Incorreta de Segurança

- Configuração insegura implementada na aplicação, frameworks, servidor de aplicação, servidor web, banco de dados e plataforma.

## A6 – Exposição de Dados Sensíveis

- Dados sensíveis, tais como cartões de crédito, IDs fiscais e credenciais de autenticação não são devidamente criptografados.

# OWASP TOP 10 - 2013

## A7 – Falta de Função para Controle do Nível de Acesso

- Falha na Restrição de Acesso a URL
- Falta de verificação dos direitos de acesso antes de tornar a funcionalidade visível na interface do usuário

## A8 – Cross-Site Request Forgery (CSRF)

- Permite ao atacante forçar o navegador da vítima a criar requisições HTTP forjadas, que a aplicação vulnerável aceite como requisições legítimas.

## A9 – Utilização de Componentes Vulneráveis Conhecidos

- Componentes, tais como bibliotecas, frameworks, e outros módulos de software quase sempre são executados com privilégios elevados e conhecidos

# OWASP TOP 10 - 2013

## A10 – Redirecionamentos e Encaminhamentos Inválidos

- os atacantes podem redirecionar as vítimas para sites de *phishing* ou *malware*, ou usar encaminhamentos para acessar páginas não autorizadas



UFAM

# SQL INJECTION





# O QUE É SQL?

- SQL (*Structured Query Language*) que significa **Linguagem de Consulta Estruturada**, é uma linguagem padrão de gerenciamento de dados.
- Principais comandos:
  - INSERT (inserção),
  - SELECT (consulta),
  - UPDATE (atualização),
  - DELETE (exclusão).



# E SQL INJECTION?

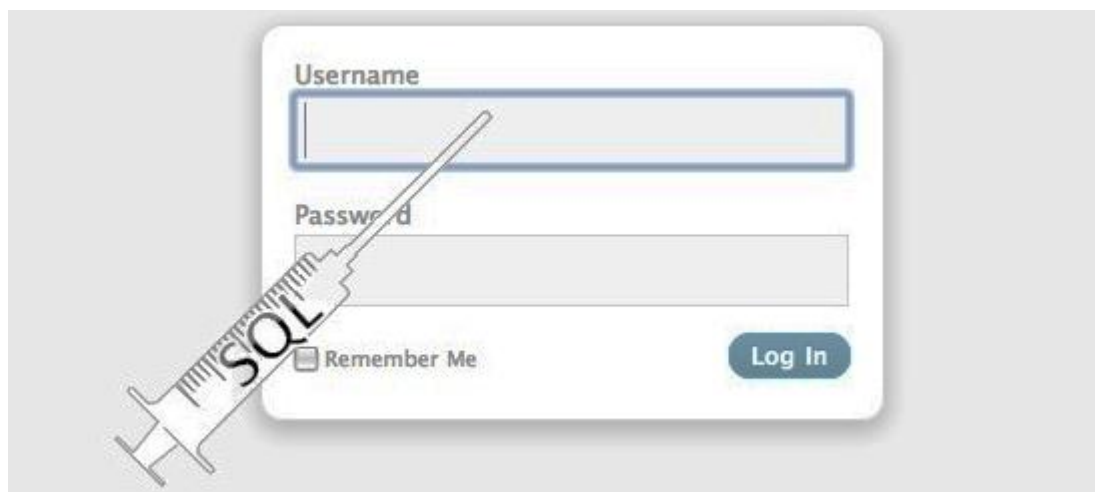
- É um tipo de ameaça de segurança que se aproveita de falhas em sistemas que interagem com bases de dados via SQL.

## SQL INJECTION



# QUANDO ACONTECE?

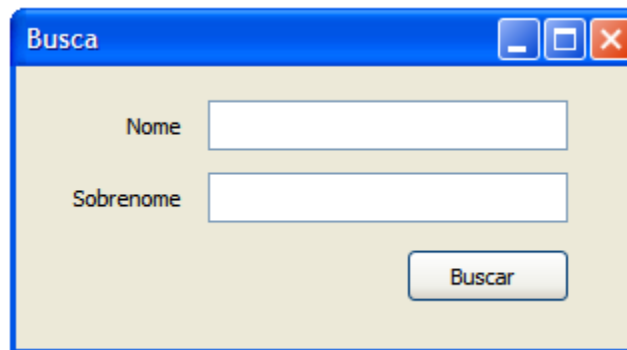
- Quando o atacante consegue inserir uma série de instruções SQL dentro de uma consulta (query), que são executados no banco de dados.
- A inserção acontece através da manipulação das entradas de dados de uma aplicação.



# EXEMPLO

- Consulta que retorna o id, nome e sobrenome dos autores a partir do filtro pelo seu nome e sobrenome

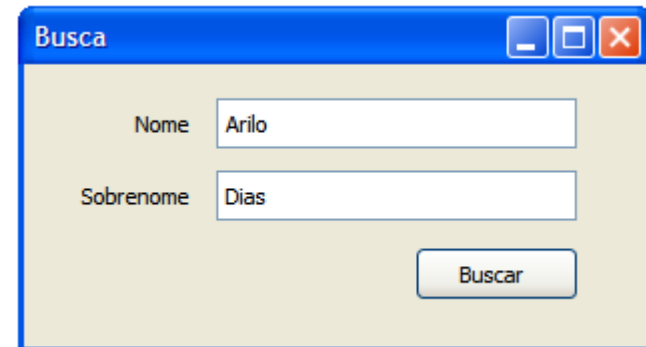
```
SELECT id, nome, sobrenome  
FROM autores  
WHERE nome = '$nome' AND sobrenome = '$sobrenome';
```



A screenshot of a Windows-style window titled "Busca". It contains two text input fields: the first is labeled "Nome" and the second is labeled "Sobrenome". Below these fields is a button labeled "Buscar". The window has standard Windows window controls (minimize, maximize, close) in the top right corner.

# EXEMPLO

- Consulta que retorna o id, nome e sobrenome dos autores a partir do filtro pelo seu nome e sobrenome
  - Exemplo de instanciação:
    - Nome = Arilo
    - Sobrenome = Dias



Busca

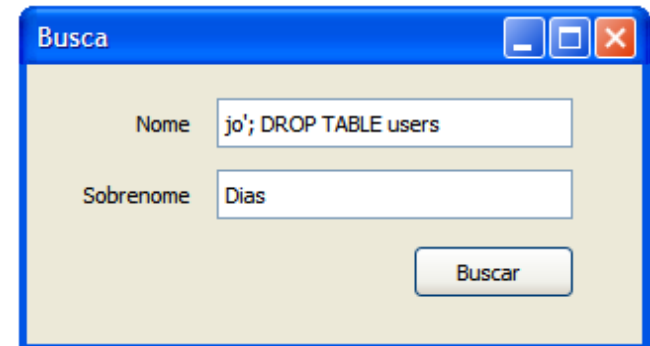
Nome

Sobrenome

```
SELECT id, nome, sobrenome  
FROM autores  
WHERE nome = 'Arilo' AND sobrenome = 'Dias';
```

# EXEMPLO

- **Problema: permite a entrada de código malicioso**
  - Nome = jo'; DROP TABLE autores ; --
  - Sobrenome = Dias



Busca

Nome jo'; DROP TABLE users

Sobrenome Dias

Buscar

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

- A tabela users seria apagada do banco de dados.



UFAM

# TÉCNICAS PARA EVITAR ATAQUES



# REJEIÇÃO DE ENTRADAS

- Sempre que puder, rejeite entrada que contenha os caracteres específicos de comandos SQL.

Caractere de entrada	Significado em Transact-SQL
;(ponto e virgula)	Delimitador de consulta.
'(apostrofo)	Delimitador de cadeia de dados de caractere.
-- (dois hifens)	Delimitador de comentário.
/* ... */	Delimitadores de comentário. Texto entre / * e * / não é avaliado pelo servidor.

- Alguns programadores usam a função **str\_replace()** para remover palavras como **SELECT**, **DELETE**, **UPDATE**, **TRUNCATE**, entre outras.

# VALIDAÇÃO DE ENTRADAS

- Verifique se uma entrada qualquer, tem o tipo de dados esperado.
- Sempre que possível, aceite apenas bons valores, ao invés de tratá-los.
  - Se a aplicação espera por entradas numéricas, considere verificar os dados com a função *is\_numeric()*.

```
bool is_numeric ( mixed $var )
```

# EXEMPLOS

---

- **Exemplo de funções em PHP**

- ctype\_digit() - Verifica se os caracteres são numéricos
- is\_bool() - Verifica se a variável é um booleano
- is\_null() - Informa se a variável é NULL
- is\_float() - Informa se a variável é do tipo float
- is\_int() - Informa se a variável é do tipo inteiro
- is\_string() - Informa se a variável é do tipo string
- is\_object() - Informa se a variável é um objeto
- is\_array() - Verifica se a variável é um array

# MODELO DE SEGURANÇA POSITIVO

---

- **Define o que é de entrada aceitável e rejeita todo o resto.**
  - Pode rejeitar dados válidos (falso positivo) se a definição não for suficientemente ampla;
  - Pode proteger de ataques conhecidos e desconhecidos (futuros);
  - Podem afetar a utilização;

# MODELO DE SEGURANÇA NEGATIVO

---

- **Define o que é de entrada inaceitável e o rejeita.**
  - Ataques atuais (conhecidos) a endereços devem ser atualizados para lidar com ataques futuros
  - Não é possível endereçar o desconhecido(falso negativo)
  - Todos os modelos de segurança negativas são potencialmente fracos

# SUPERUSUÁRIO DE BD

- **Nunca conecte ao banco de dados como um superusuário ou como o dono do banco de dados.**
  - Use sempre usuários personalizados com privilégios bem limitados.



- Muitos preferem usar superusuário ou administrador por lhe dar mais privilégios. **Os hackers também pensam assim.**

# TRATAMENTO DE STRING

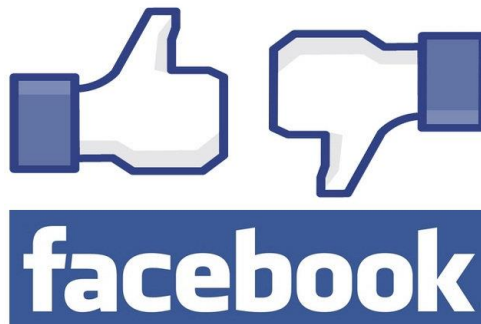
---

- **Adicione aspas para cada valor não numérico especificado pelo usuário que será passado para o banco de dados com as funções de caracteres de escape.**
- **Exemplos:**
  - `mysql_real_escape_string()`
  - `sqlite_escape_string()`



# INFORMAÇÕES PRIVADAS DO BD

- Não imprima qualquer informação específica do banco de dados, especialmente sobre o esquema, custe o que custar.



# NUNCA USE CONSULTAS DINÂMICAS

---

- **Use apenas APIs parametrizadas**
  - Essas APIs codificam a entrada do usuário, e certificam-se que ele não quebre as instruções SQL
- **Use procedimentos armazenados**
  - Eles geralmente estão seguros contra SQL injection
  - **Nota:** concatenando argumentos ou usando `exec ( )` dentro de um procedimento armazenado pode torná-lo vulnerável

# FUNÇÕES PRÉ-DEFINIDAS

---

- Você pode usar *stored procedures* e *cursores* previamente definidas para abstrair acesso aos dados para que os usuários não acessem tabelas ou *views* diretamente, mas essa solução pode ter outros impactos.



UFAM

# UM POUCO DE PRÁTICA

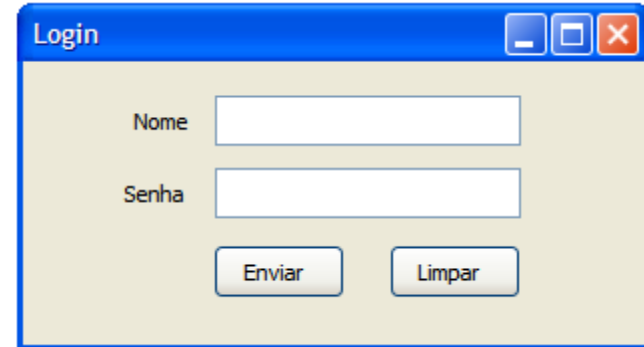
# SITE PARA TESTES

---

- **Abra o site que está no endereço**
- <http://www.altoromutual.com/bank/login.aspx>

# EXEMPLO PRÁTICO

- **Tela de Login**
  - Um formulário típico de login.



Login

Nome

Senha

```
form name="frmLogin" action="login.asp" method="post">  
  Nome : <input type="text" name="nomeUsuario">  
  Senha: <input type="password" name="senhaUsuario">  
  <input type="submit" name="Enviar" >  
</form>
```

- Ao clicar no botão **Enviar** será executado um script para efetuar a validação dos dados informados.

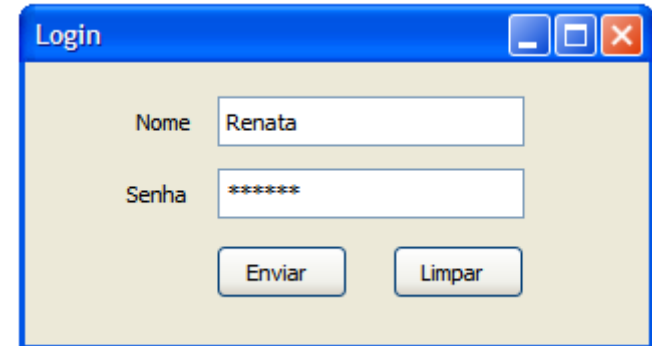
# SCRIPT DE VALIDAÇÃO

```
<%  
dim nomeUsuario, senhaUsuario, consulta  
dim conn, rs  
nomeUsuario = Request.Form("nomeUsuario")  
senhaUsuario = Request.Form("senhaUsuario")  
  
set conn = server.createObject("ADODB.Connection")  
set rs = server.createObject("ADODB.Recordset")  
  
consulta = "select count(*) from usuarios where nomeUsuario='" &  
nomeUsuario & "' and senhaUsuario='" & senhaUsuario & "'"   
  
conn.Open "Provider=SQLOLEDB; Data Source=(local);Initial  
Catalog=myDB; User Id=sa; senhaUsuario="   
  
rs.activeConnection = conn  
rs.open consulta  
if not rs.eof then  
    response.write "Acesso Concedido"  
else  
    response.write "Acesso Negado"  
end if  
%>
```



# EXEMPLO PRÁTICO

- **Tela de Login**
  - Exemplo de instanciação:
    - Nome = Renata
    - Sobrenome = 123456



A screenshot of a Windows-style login window titled "Login". It features two input fields: "Nome" with the text "Renata" and "Senha" with masked characters "\*\*\*\*\*". Below the fields are two buttons labeled "Enviar" and "Limpar". The window has standard Windows window controls (minimize, maximize, close) in the top right corner.

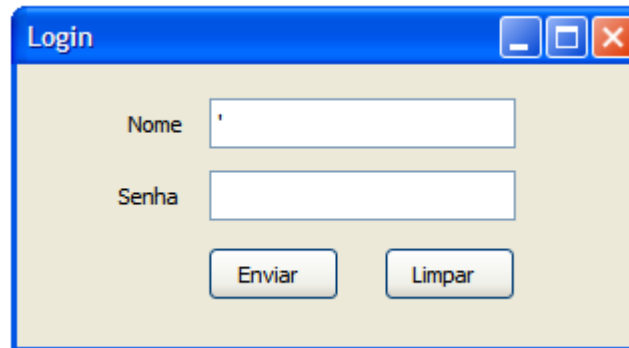
```
select count(*) from usuarios where  
nomeUsuario=' ' & nomeUsuario & ' ' and  
senhaUsuario=' ' & senhaUsuario & ' '
```

```
select count(*) from usuarios where  
nomeUsuario= 'Renata' and  
senhaUsuario='123456'
```

# EXEMPLO PRÁTICO

- **Problema: permite a entrada de código malicioso**

- Nome = '
- Senha = '



A screenshot of a web application's login window. The window has a blue title bar with the text 'Login' and standard window control buttons (minimize, maximize, close). The main area is light beige. It contains two text input fields: the first is labeled 'Nome' and contains a single apostrophe character; the second is labeled 'Senha' and is empty. Below the fields are two buttons: 'Enviar' and 'Limpar'.

- Se o apóstrofo (aspas simples) não for tratado acontecerá erro de SQL

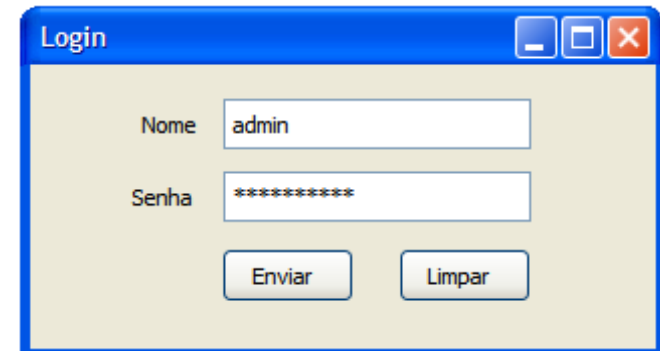
**Erro:** Syntax error

- .

# EXEMPLO PRÁTICO

- **Problema: permite a entrada de código malicioso**

- Nome = admin
- Senha = ' or 1=1--



A screenshot of a web application's login window. The window has a blue title bar with the text 'Login' and standard window control buttons (minimize, maximize, close). The main area is light beige. It contains two input fields: 'Nome' with the value 'admin' and 'Senha' with masked characters '\*\*\*\*\*'. Below the fields are two buttons: 'Enviar' and 'Limpar'.

- A consulta vai verificar se existe o usuário **admin** e se a senha é **vazio ou 1=1**

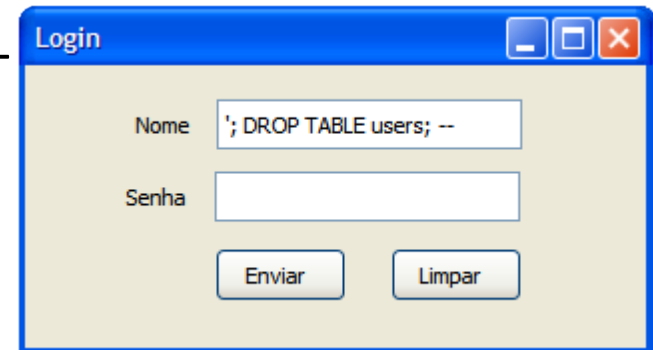
```
select count(*) from usuarios where  
nomeUsuario= 'admin' and  
senhaUsuario= '' or 1=1 --
```

VERDADE

# EXEMPLO PRÁTICO

- **Problema: permite a entrada de código malicioso**

- Nome = `' ; DROP TABLE users; --`
- Senha =



A screenshot of a web application's login window. The window has a blue title bar with the text "Login" and standard window control buttons (minimize, maximize, close). The main area is light beige. It contains two input fields: "Nome" and "Senha". The "Nome" field contains the text `' ; DROP TABLE users; --`. The "Senha" field is empty. Below the fields are two buttons: "Enviar" and "Limpar".

- Esse comando vai excluir a tabela **users**

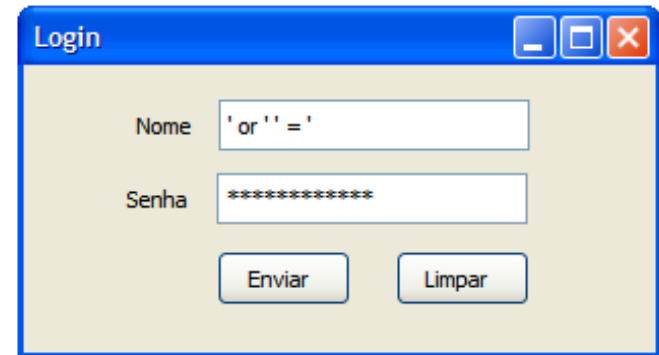
```
select count(*) from usuarios where  
nomeUsuario= ' ' ; DROP TABLE users; -- '  
and senhaUsuario=' '
```

- .

# EXEMPLO PRÁTICO

- **Problema: permite a entrada de código malicioso**

- Nome = ' or ' ' = '
- Senha = ' or ' ' = '



The screenshot shows a 'Login' window with two input fields. The 'Nome' field contains the text ' or ' ' = ' and the 'Senha' field contains '\*\*\*\*\*'. Below the fields are two buttons: 'Enviar' and 'Limpar'.

- A consulta vai verificar se existe o usuário **vazio ou vazio = vazio** e se a senha é **vazio ou vazio = vazio**

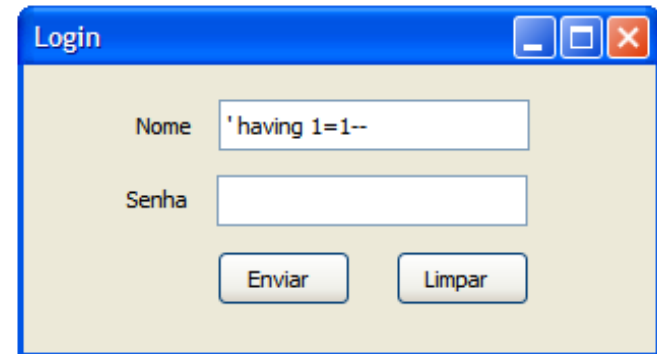
```
select count(*) from usuarios where  
nomeUsuario= ' or ' ' = ' and  
senhaUsuario= ' or ' ' = ' '
```

- .

# EXEMPLO PRÁTICO

- **Problema: permite a entrada de código malicioso**

- Nome = ' having 1=1--
- Senha =



A screenshot of a web application's login window. The window has a blue title bar with the text 'Login' and standard window control buttons (minimize, maximize, close). The main area is light beige. It contains two text input fields: the first is labeled 'Nome' and contains the text ' having 1=1--'; the second is labeled 'Senha' and is empty. Below the fields are two buttons: 'Enviar' (Send) and 'Limpar' (Clear).

- Utiliza esse comando para saber nome das tabelas e campos do banco de dados, essa consulta gera um erro.

**Erro:** Microsoft OLE DB Provider for ODBC Drivers error '80040e14'

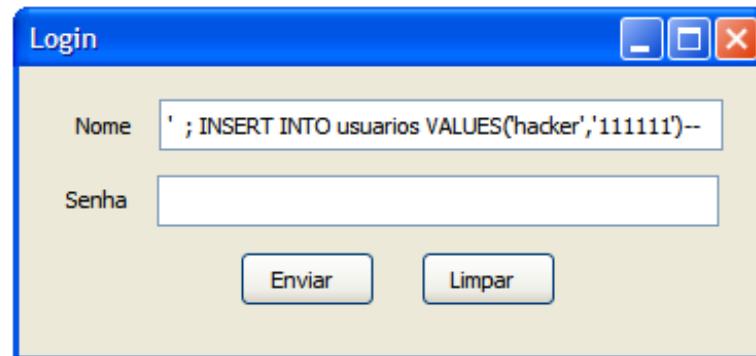
[Microsoft] [ODBC SQL Server Driver] [SQL Server] **Column**  
'usuarios.codigo' is invalid in the select list because it is not contained in an  
aggregate function and there is no GROUP BY clause.

- .

# EXEMPLO PRÁTICO

- **Problema: permite a entrada de código malicioso**

- Nome = ' ; INSERT INTO usuarios VALUES('hacker','111111')--  
Senha =



The screenshot shows a 'Login' window with two input fields: 'Nome' and 'Senha'. The 'Nome' field contains the text: ' ; INSERT INTO usuarios VALUES('hacker','111111')--'. The 'Senha' field is empty. Below the fields are two buttons: 'Enviar' and 'Limpar'.

- Esse comando vai incluir o usuário **hacker** com a senha **111111**





UFAM

# FERRAMENTA DE APOIO A IDENTIFICAÇÃO DE VULNERABILIDADE

# VISÃO GERAL

---

- Web sites estão ficando mais e mais complexos a cada dia e quase não há mais sites estáticos sendo desenvolvidos.
- Mesmo se o site é 100% “feito à mão”, quando nós confiamos em nosso próprio trabalho e pensamos que tudo está seguro, é possível que um caractere especial não tenha sido tratado ou não tenhamos conhecimento sobre novas técnicas de ataque.

# HAVIJ

- É uma ferramenta automatizada de injeção SQL que ajuda a executar testes de penetração, encontrar e explorar vulnerabilidades de injeção SQL em páginas web.
- Software pago, mas possui versão grátis.

# HAVIJ

- **Utilidades:**

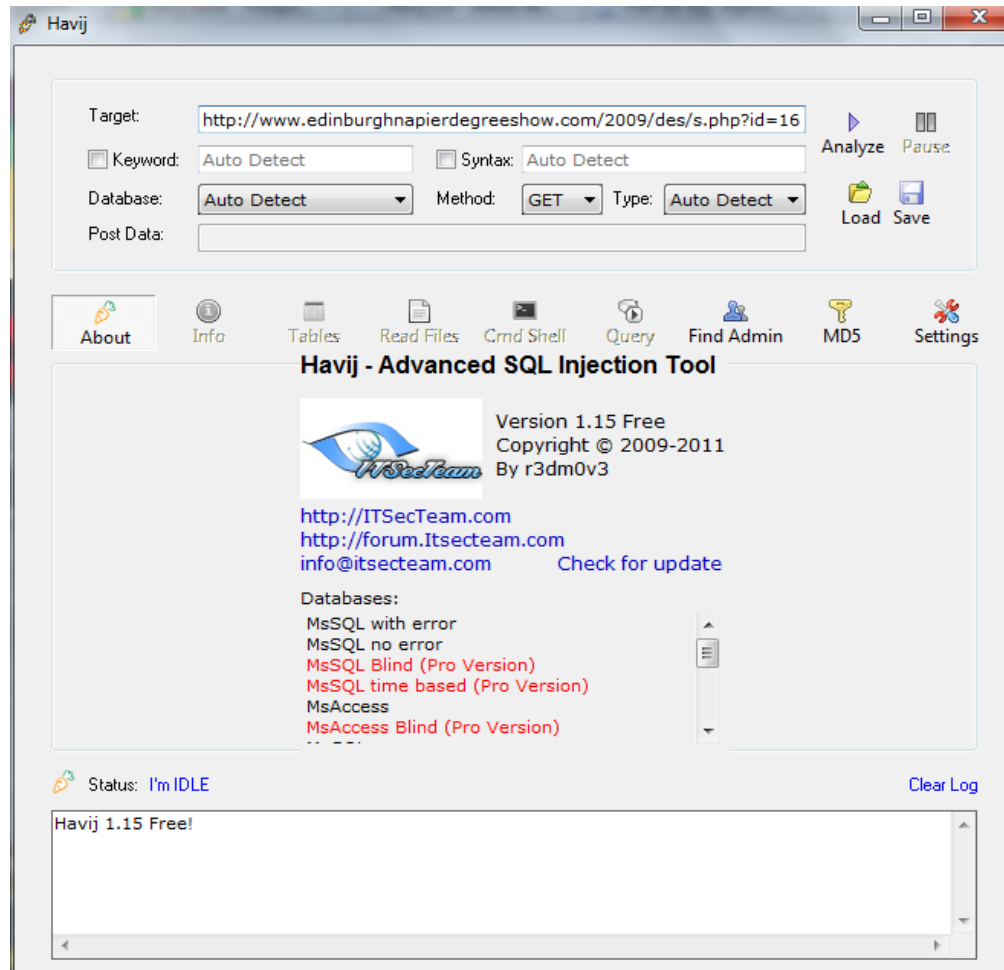
- Executar software de back-end de banco de dados de impressões digitais.
- Recuperar os usuários DBMS e os hashes de senha, tabelas e colunas de despejo.
- Buscar dados do banco de dados.
- Acessar o sistema de arquivos subjacente e executa comandos no sistema operacional

# HAVIJ - INSTALAÇÃO

---

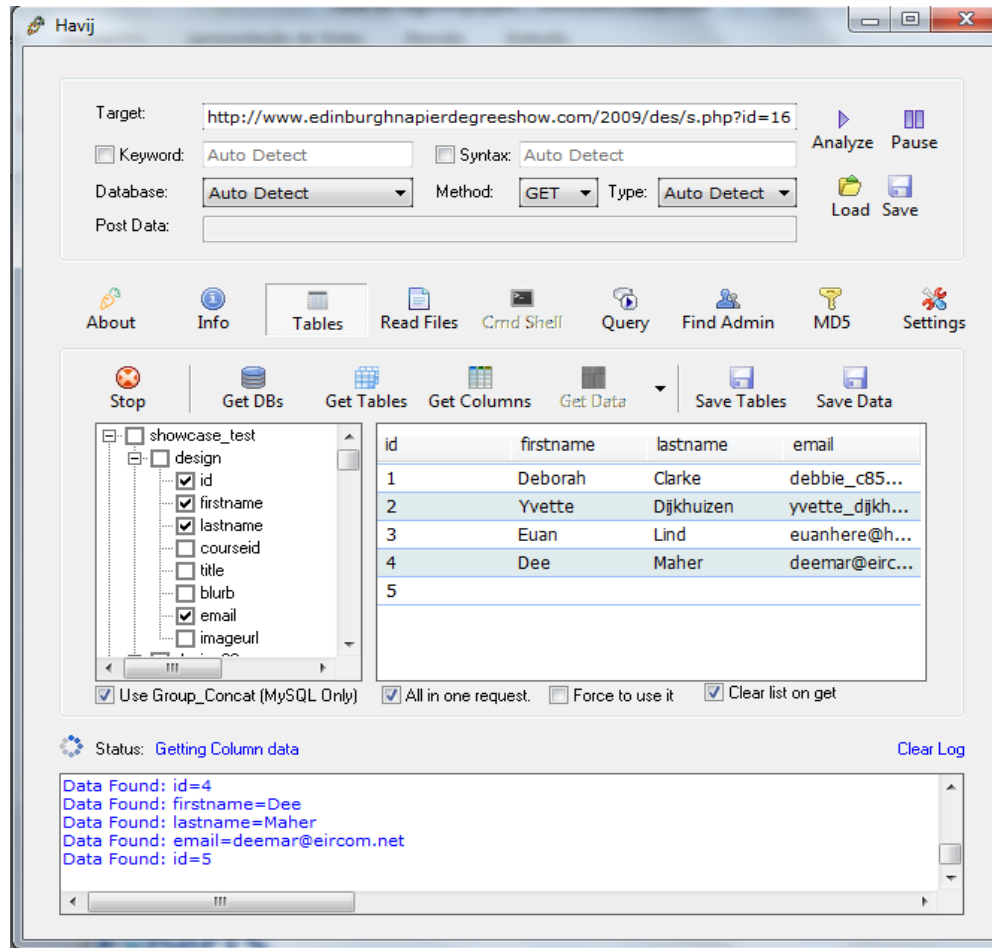
- Baixe de:  
<http://www.itsecteam.com/products/havij-advanced-sql-injection/>
- Não requer nenhuma configuração avançada durante a instalação.

# HAVIJ – COMO USAR



- Ferramenta bem intuitiva, onde somente analisando o “alvo” já é possível descobrir falhas e invadir um sistema “frágil”.

# HAVIJ – COMO USAR



- Depois de analisado, caso encontra uma falha e acesse o banco de dados é possível acessar todos os dados deste e confirma a fragilidade do sistema.





UFAM

# PRÁTICA COM HAVIJ

# SITES PARA TESTES

---

- **Abra o site que está no endereço**
- <http://testphp.vulnweb.com/search.php?test=query>

# REFERÊNCIAS

---

- **IBM Security AppScan disponível em** <http://www.ibm.com/developerworks/downloads/r/appscan/>
- **OWASP – Open Web Application Security Project disponível em** [https://www.owasp.org/index.php/Main\\_Page](https://www.owasp.org/index.php/Main_Page)
- **As 10 vulnerabilidades de segurança mais críticas em aplicações WEB – OWASP Top 10 2013 disponível em** [http://owasptop10.googlecode.com/files/OWASP\\_Top\\_10\\_-\\_2013\\_Brazilian\\_Portuguese.pdf](http://owasptop10.googlecode.com/files/OWASP_Top_10_-_2013_Brazilian_Portuguese.pdf)
- **Itsecteam – Havij Advanced SQL Injection disponível em** <http://www.itsecteam.com/products/havij-advanced-sql-injection/>
- **Macoratti.net – Previna-se contra a Injeção SQL disponível em** [http://www.macoratti.net/sql\\_inj.htm](http://www.macoratti.net/sql_inj.htm)

# **GARANTINDO A SEGURANÇA DE APLICAÇÕES WEB CONTRA SQL INJECTIONS**

**Arilo Claudio Dias Neto (ExperTS)**

Coordenador do Grupo ExperTS - [ariloclaudio@gmail.com](mailto:ariloclaudio@gmail.com)

**Renata Magalhães Rêgo (ExperTS)**

Mestranda em Informática - [renatamagalhaesrego@gmail.com](mailto:renatamagalhaesrego@gmail.com)