Journal of Technology & Information

Mitigação de Ataques de SQL Injection em um Projeto de Tecnologia da Informação em Ambiente WEB

Mitigation of SQL Injection Attacks in an Information Technology Project in a WEB Environment

Recebido: 20/10/2022 | Revisado: 23/10/2022 | Aceito: 25/10/2022 | Publicado: 26/10/2022

Irapuan Glória Júnior
Fatec Santana de Parnaíba
https://orcid.org/0000-0003-2973-3470
ijunior@ndsgn.com.br

Resumo

As empresas utilizam cada vez mais sistemas computacionais em ambiente WEB, ocasionando a possibilidade de diversos tipos de ataques, com o SQL *Injection*. Este tipo de ataque consiste em enviar códigos maliciosos em determinados objetos do sistema e, desta forma, obter acesso e informações privilegiadas. Diante deste contexto, a pesquisa apresentou uma sugestão de mitigação de ataques de SQL *Injection* em um projeto de Tecnologia da Informação em Ambiente WEB. O artigo possui natureza qualitativa e utilizou a metodologia de *Design Science Research*. Os resultados foram uma estrutura de uma classe que realiza o tratamento dos dados recebidos, a recomendação de usar o método *parameters*() ao criar as chamadas e o uso de *Stored Procedures* nas operações com os SGBD. A contribuição para a teoria foi de apresentar uma sugestão de mitigação dos ataques e que carecem de mais estudos. A contribuição para a prática é a de que analistas e gestores de TI possam se beneficiar das recomendações desta pesquisa.

Palavras-chave: Cybersecurity. SQL *Injection*. Segurança da Informação. Sistemas WEB. MSSQL

Abstract

Companies are increasingly using computer systems in a WEB environment, causing the possibility of various types of attacks, with SQL Injection. This type of attack consists of sending malicious code into certain system objects and, in this way, obtaining access and privileged information. Given this context, the research presented a suggestion to mitigate SQL Injection attacks in an Information Technology project in a WEB Environment. The article has a qualitative nature and used the Design Science Research methodology. The results were a structure of a class that handles the data received, the recommendation to use the parameters() method when creating the calls and the use of Stored Procedures in operations with the DBMS. The contribution to the theory was to present a suggestion of mitigation of attacks and that need further studies. The contribution to the practice is that programmers, analysts and IT managers can benefit from the recommendations of this research.

Keywords Cybersecurity. SQL Injection. Information Security. WEB Systems. MSSQL

1. Introdução

Atualmente com o advento de novas tecnologias as empresas necessitam de sistemas específicos para suprir suas necessidades corporativas (Kotler et al., 2016), utilizando equipes de desenvolvimento de sistemas que utilizam linguagens computacionais para criarem artefatos que atendam aos requisitos impostos pelos gestores (Pressman & Maxim, 2016; Sommerville, 2015).

O uso da internet está sendo cada vez mais utilizado como forma de crescimento das vendas, tornando esse ambiente propício para os novos modelos de negócio (Kotler, 2021; Kotler et al., 2021), ficando vulnerável a vários tipos de ataques, com o SQL *Injection*.

Um ataque de SQL *Injection* consiste no envio de comandos maliciosos em campos alfanuméricos em sistemas WEB com o intuito de obter acesso, senhas e demais informações sigilosas (Microsoft, 2022e; Sadeghian et al., 2013; Shar & Tan, 2013).

Diante deste cenário, a Questão de Pesquisa (QP) deste trabalho é: "Como evitar um ataque de SQL *Injection* em um projeto de Tecnologia da Informação em Ambiente WEB?", com o objetivo de apresentar uma sugestão computacional para mitigar esse tipo de ataque.

2. Referencial Teórico

2.1. Linguagens de Programação

As empresas desenvolvem sistemas utilizando diversas linguagens computacionais (Pressman & Maxim, 2016; Sommerville, 2015) para suprir necessidades específicas para a prestação de serviços ou criação de produtos (Kotler et al., 2016).

Dentre as linguagens utilizadas no mercado computacional é o ASPx que é empregada para o desenvolvimento de sistemas no ambiente WEB e possui um

mecanismo, o *Engine*, que possa executar a solução computacional diretamente em um banco de dados ou interconectada a uma outra linguagem, como o C# (Microsoft, 2022a).

Outra linguagem usada no desenvolvimento é o C# que pode ser utilizado pelas empresas de soluções computacionais para o desenvolvimento de sistemas, jogos e sistemas especialistas (Microsoft, 2022b).

Os sistemas normalmente utilizam um repositório de dados, denominado Sistema Gerenciador de Banco de Dados (SGBD), que além do armazenamento, possuem criações de perfis de usuários, criptografia, sistema de backup, integridade de dados, relacionamento de informações e recursos de otimização de consulta e desempenho de atividades (Microsoft, 2022f).

2.2. Segurança da Informação

As empresas estão empregando cada vez mais as Tecnologias de Informação e Comunicação (TIC) para disponibilizarem serviços e outras transações em tempos em que a digitalização dos negócios é vital para a maioria das empresas (Kotler et al., 2021).

Em decorrência disso, os dados coletados, tratados e armazenados devem possuir dispositivos digitais que promovam a segurança contra tentativas de ataques e invasões na empresa e para os usuário que utilizam seu ambiente (Fernandes, 2013).

Os princípios de desenvolvimento de aplicações seguras são apresentados pelo triângulo CIA (Hintzbergen et al., 2018), que contém os conceitos de confidencialidade, integridade e disponibilidade dos dados relativos a toda instalação de software, análise de dados ou fornecimento de acesso a algum dado ou outra informação (Tipton et al., 2016).

O triângulo CIA, acrônico de Confidencialidade/*Confidentiality*, Integridade/*Integrity* e Disponibilidade/*Availability*, é citado na norma ISO 27.001 ao especificar os requisitos para estabelecer, implementar, manter e melhorar continuamente qualquer sistema de gerenciamento de segurança da informação (ISO, 2013).

A **confidencialidade** está relacionada com os limites de acessos às informações dos usuários em um ambiente computacional por meio de medidas protetivas como a criptografia, *traffic padding, etc* (Hintzbergen et al., 2018).

A garantia de que uma informação seja acessada por indivíduos autorizados por meio de autenticação é vital em um sistema, como os casos das atribuições de perfis para os usuários para bloquear o acesso de um funcionário a informações estratégicas (Grassi et al., 2017).

Em relação a **integralidade,** compreende a garantia de que uma informação continue completa e inalterada (Fernandes, 2013), e que qualquer modificação não autorizada de dados, que tenha sido realizada intencionalmente ou acidentalmente, seja considerada uma violação (Hintzbergen et al., 2018).

A **disponibilidade** visa garantir que a informação esteja disponível para aqueles que possuem autorização para acessá-la, de acordo com as políticas estipuladas pela empresa (Diana et al., 2016).

A necessidade de estabelecer a Tríade CIA no Brasil está atrelado, dentre outras coisas, a Lei Geral de Proteção de Dados (LGPD) que está em vigor desde 2020 e regulamenta o uso e manipulação de dados no país, criando um ambiente seguro para empresas e usuários e, caso seja contemplada, poderá causar prejuízos financeiros e macular a imagem das empresas (Barcelos et al., 2021; Botelho & Camargo, 2021; Raes, 2022).

2.3. SQL *Injection*

O SQL *Injection*, ou injeção de SQL, é um ataque em um sistema WEB em que um código mal-intencionado é inserido em um campo de entrada alfanumérica com instruções em SQL que, ao ser executada, resultará em uma manipulação de dados não autorizada (Microsoft, 2022e).

Esse tipo de ataque não é identificado nos firewalls, pois o ambiente interpreta como uma consulta comum (Macoratti, 2008), desta forma, no desenvolvimento do sistema deverá ser considerado estratégias para mitigar esse tipo de ataque (Macoratti, 2008; Sadeghian et al., 2013; Shar & Tan, 2013).

O ataque ocorre considerando que o sistema utilizará da concatenação de *string*s para gerar um comando SQL para fazer a busca ou qualquer manipulação em um banco de dados conforme apresentado no Código 1 (Macoratti, 2008), em que o comando SQL será criado, utilizando ASPx, na variável mMySQL com um comando SELECT consultando a tabela TB_Usuario para obter o valor "1" caso exista algum usuário que possua o valor do campo Ds_Login e Ds_Senha idênticos aos que foram informados no formulário (*form*) pelos objetos *textbox* (Microsoft, 2022a) txtDs_Login e txtDs_Senha. No Código 2 é apresentado um exemplo com o login JOAO e a senha 1234.

Código 1 – Comando SQL baseado em concatenação de *String*s baseado em MACORATTI (2008)

```
mMySQL = "SELECT 1 FROM TB_Usuario" & _
          WHERE Ds_Login = '" & txtDs_Login & "'" & _
         " AND   Ds_Senha = '" & txtDs_Senha.Text & "';"
```

Código 2 – Comando SQL com os dados baseado em MACORATTI (2008)

```
SELECT 1 FROM TB_Usuario
      WHERE Ds_Login = 'JOAO'
      AND   Ds_Senha = '1234';
```

O atacante poderá informar nos objetos *textbox* os valores para login e senha respectivamente: "sa" e '" or 1=1 –" (Macoratti, 2008). Neste caso o comando irá atribuir "sa" para o nome do usuário, que corresponde ao administrador padrão do banco de dados (Microsoft, 2022f), e também indicará no outro objeto de texto um apóstrofe, o operador lógico OR, seguido de uma condição verdadeira, a "1 = 1", e a sequência de dois traços, fazendo com que o resto da linha seja considerada um comentário e, desta forma, retornando o resultado como *True* na consulta e resultado na permissão de entrada, conforme apresentado no Código 3.

Código 3 – Comando SQL com código malicioso baseado em MACORATTI (2008) e Microsoft (2022f)

```
SELECT 1 FROM TB_Usuario
      WHERE Ds_Login = 'sa'
      AND   Ds_Senha = '' OR 1=1 --';
```

3. Metodologia

Este trabalho possui natureza qualitativa (Theophilo & Martins, 2016), com uso da metodologia *Design Science Research*, que constitui de um framework científico com a análise do problema, especificações de técnicas e apresentação da solução (Nunamaker Júnior et al., 1990). Em relação a coleta de dados foi realizada a partir da compilação de documentos digitais e informações relacionadas na empresa. A unidade de análise foram os sistemas WEB da empresa conforme apresentado na Tabela 1.

Tabela 1 - Características Metodológicas

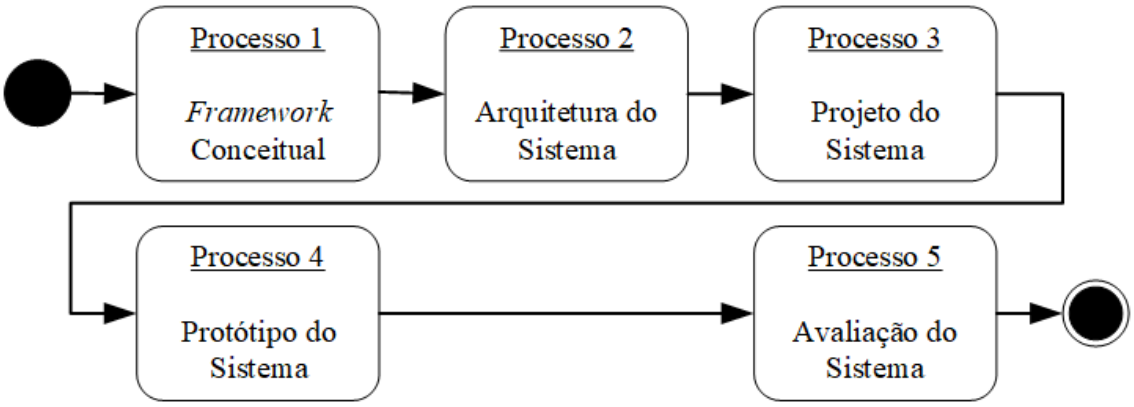
Item	Conteúdo	Autores
Natureza	Qualitativa	Theophilo e Martins (2016)
Metodologia	*Design Science Research*	Nunamaker Júnior, Chen & Purdin (1990)
Coleta de Dados	- Documentos Digitais - Informações provenientes da empresa	
Unidade de Análise	Sistemas WEB Existentes	

3.1. Procedimentos Metodológicos

Os procedimentos para o desenvolvimento desta pesquisa, conforme apresentadas na Figura 1 e pautadas na metodologia escolhida, são:

- **Processo 1: *Framework* Conceitual.** Descrição dos requisitos para a solução computacional;
- **Processo 2: Arquitetura do Sistema.** Apresentação de inter-relações do artefato e sua arquitetura de software;
- **Processo 3: Projeto do Sistema.** Representação do sistema por meio de diagramas de UML;
- **Processo 4: Protótipo do Sistema.** Apresentação e detalhes das telas do sistema;
- **Processo 5: Avaliação do Sistema.** Apresentação de uma tabela com as relações entre os requisitos e os processos que os contemplam.

Figura 1 - Procedimentos Metodológicos baseado em NUNAMAKER JR; CHEN; PURDIN (1990).



3.2. Objeto de Estudo

Uma empresa do setor de desenvolvimento de sistemas, denominada na pesquisa como Empresa-Alpha, está localizada no estado de São Paulo, possui mais de 20 anos no mercado e atua com empresas de grande porte nos setores de telecomunicações, bancária e química.

Possui sistemas de controle específicos em seus clientes, todos constituídos em ambiente WEB, nas linguagens ASPx (Microsoft, 2022a) e C# (Microsoft, 2022b) e necessita de uma rotina que torne seguro uma *string* recebida pelo sistema proveniente de um usuário.

4. Análise e Interpretação dos Resultados

4.1. Framework Conceitual

A empresa necessita que todas as suas interações com o usuário tenham validações para evitar um ataque *SQL Injection*, a começar pela autenticação ao sistema (1.0), que ao ser criada a rotina, os programadores irão propagar em todo o sistema.

O procedimento de autenticação consiste em obter o Login (**RF01**) e Senha (**RF02**) do usuário que deverá informar via caixas de texto, que correspondem a um objetivo que recebe qualquer tipo de conteúdo alfanumérico (Microsoft, 2022b).

Após a obtenção das informações, será realizado o tratamento dos dados (**RF03**) para evitar caracteres que promoverão um ataque malicioso, resultando em um texto tratado e que poderá ser utilizado para acessar o banco de dados.

Neste ponto o sistema deverá ter as informações tratadas e prontas para verificar se existe uma correlação entre o login e senha digitados com a mesma informação em um registro no banco de dados (**RF04**).

A busca retornará dois tipos de status: (1) Negativo (*False*) caso não tenha encontrado correspondente e então o sistema deverá avisar o usuário da falha (**RF05**); ou (2) Positivo (*True*), deverá acessar o sistema (**RF06**).

Tabela 2 – Requisitos Funcionais (RF) do Sistema baseado nos requisitos de Sistemas da Empresa

#	Rótulo	Descrição
RF01	Login	O usuário irá digitar o login
RF02	Senha	O usuário irá digitar a senha conforme as regras vigentes na empresa
RF03	Tratamento *SQL-Injection*	O sistema deverá realizar o tratamento dos dados para evitar ataque
RF04	Verificar a Autenticação do Usuário	O sistema deverá verificar se o Login+Senha no Banco de Dados
RF05	Resultado da Busca: Negativa	Avisar o usuário de que não encontrou
RF06	Resultado da Busca: Positiva	Acessar o sistema

A empresa necessita que para cada entrada de texto, proveniente de uma página WEB, tenha a verificação de entradas maliciosas do SQL *Injection*.

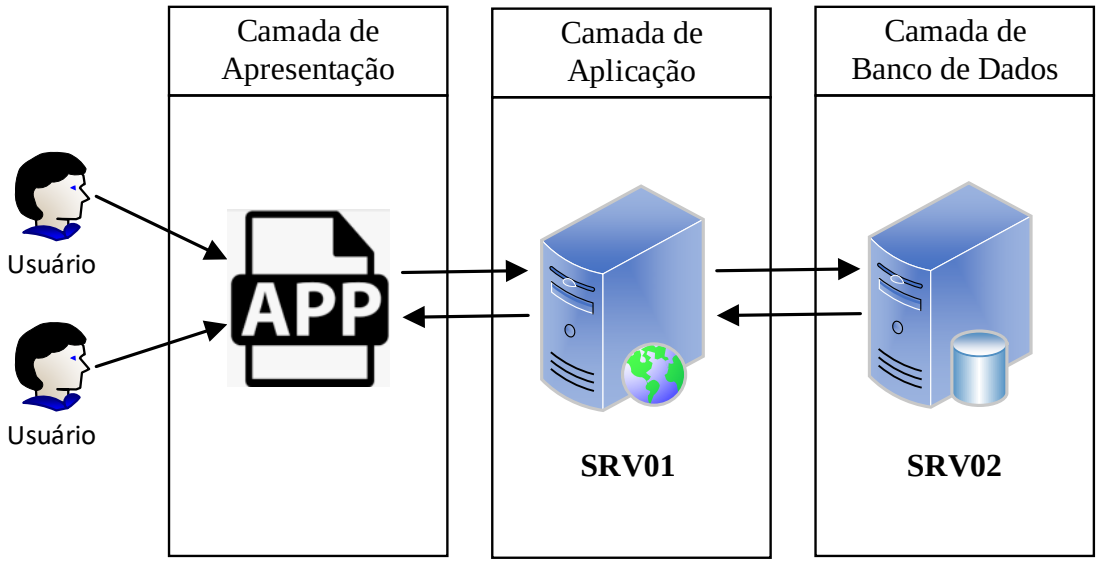
4.2. Arquitetura do Sistema

O sistema será desenvolvido em ambiente WEB e composto por *n-tier* (Microsoft, 2022c): (1) Camada de apresentação, possui o acesso às telas do sistema; (2) Camada de Aplicação, utilização do *Engine* Internet Information Service para a execução do código das telas, o chamado *Code Behind*; (3) Camada de Banco de Dados, com o Microsoft SQL Server como sistema gerenciador de banco de dados.

A Figura 2 apresenta a arquitetura do sistema, em que o acesso realizado pelos usuários nos formulários ASPx (*Form*) do sistema que estão na **Camada de Apresentação/SRV01**. Assim que é realizada alguma requisição de dados, como uma

consulta ou envio dos dados para um cadastro, as informações são enviadas para a **Camada de Aplicação/SRV01**. Nesta camada, existem os componentes (DLL) do sistema que ao ser executado poderá retornar alguma mensagem de erro ou os dados, mas também pode solicitar ou enviar informações para ler ou gravar dados na **Camada de Banco de Dados/SRV02** que contém o SGBD.

Figura 2 – Arquitetura do Sistemas baseada em Microsoft (2022c)



4.3. Projeto do Sistema

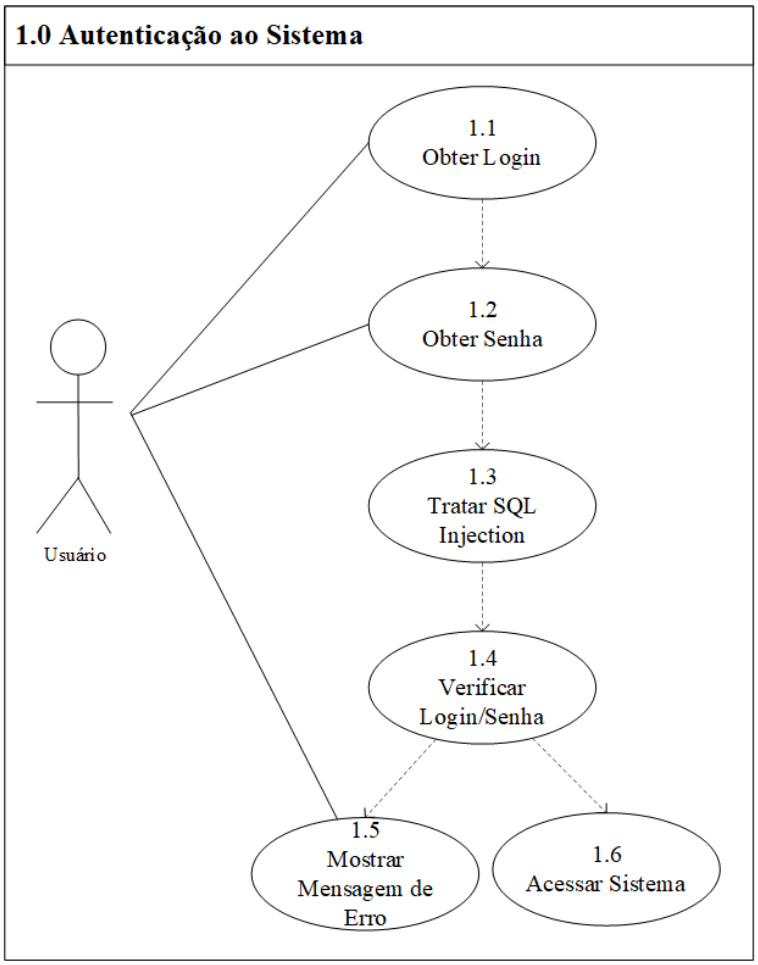
As ações previstas no sistema foram representados (Figura 3) utilizando um Diagrama de UML, sendo o Diagrama de Caso de Uso (DCU) aquele que utiliza representa os processos do sistema (Pressman & Maxim, 2016; Sommerville, 2015).

No diagrama foi considerado genericamente que aquele que irá transacionar informações com o sistema será denominado **Usuário** e representado pelo símbolo do *pivot-man*/homem-palito.

O primeiro processo é o **obter login (1.1)** em que o usuário irá informar o login para realizar o acesso. A digitação da senha é em **obter senha (1.2)** que é informada ao sistema. Baseado nas informações digitadas, irá **tratar o SQL Injection (1.3)** retornando a *string* sem códigos maliciosos.

Após a limpeza das entradas, é realizada a consulta no SGBD com os dados informados no processo **Verificar Login/Senha (1.4)** em que podem ocorrer duas situações: os dados são inválidos, em que o sistema irá **mostrar mensagem de erro (1.5)**; Caso seja válido, o usuário irá **acessar sistema (1.6)**.

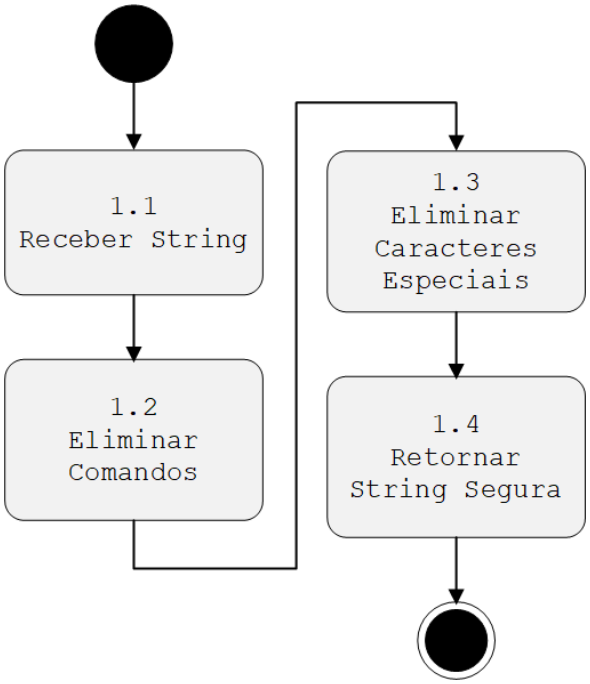
Figura 3 – Diagrama de Caso de Uso do Login do Sistema



O Diagrama de Atividade (DAtiv) apresenta os processos a serem desenvolvidos pelos programadores em um nível mais detalhado (OMG, 2015), além de poderem indicar quais as *function* ou *procedures* que deverão ser elaboradas (Pressman & Maxim, 2016).

No DAtiv (Figura 4) a rotina proposta para tratamento do SQL *Injection*, a SQLInjection(), consiste em receber a *string* a ser verificada **(1.1)**, eliminar os comandos (**1.2**) e caracteres especiais (**1.3**) que possam servir de comentários ou concatenação, como o "&". Após a retirada, devolver a *string* segura (**1.4**) ao processo que requisitou o tratamento.

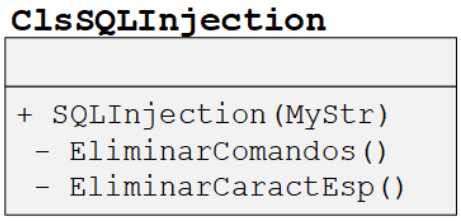
Figura 4 – Diagrama de Atividade da rotina de tratamento do SQL *Injection*



A arquitetura *n-tier* promove o uso de componentes, desta forma, utiliza classes para serem instanciadas e consumidas (MICROSOFT, 2022c; OMG, 2015) culminou na criação da classe **ClsSQLInjection()** conforme apresentado na Figura *5*. Ao ser instanciada, o objeto não possui atributos e apenas métodos.

O método público SQLInjection recebe o parâmetro MyStr do tipo alfanumérico, que faz a chamada para os métodos privados EliminarComandos() e EliminarCaractEsp() que retiram da *string* informada os comandos SQL e os caracteres especiais, respectivamente. Ao final, retorna a informação segura para a instância.

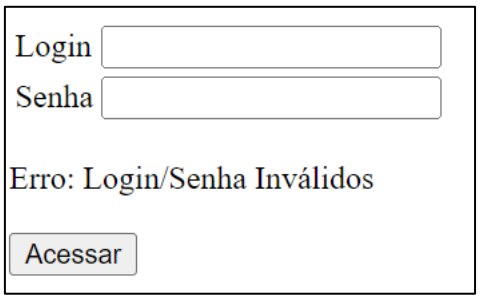
Figura 5 – Diagrama de Classes da rotina de tratamento do SQL *Injection*



4.4. Protótipo do Sistema

A Figura 6 representa o *Form* de acesso do sistema da Empresa-Alpha, em que o usuário informa o login/senha e depois clica no objeto *Button* (Microsoft, 2022a) com o rótulo Acessar. Caso haja algum erro, será exibido no objeto *Label* (MICROSOFT, 2022a) que contém o texto de exemplo "Erro: Login/Senha Inválidos".

Figura 6 – Tela de Login



4.5. Avaliação do Sistema

A validação consiste em verificar se os requisitos indicados pela Empresa-Alpha foram contemplados na sugestão de solução demonstrado no Diagrama de Caso de Uso. A constatação da relação entre os requisitos funcionais e os processos do DCU estão na Tabela 3.

Tabela 3 – Requisitos Funcionais (RF) do Sistema e os processos no DCU

#	Rótulo	Processo DCU
RF01	Login	1.1
RF02	Senha	1.2
RF03	Tratamento *SQL-Injection*	1.3
RF04	Verificar a Autenticação do Usuário	1.4
RF05	Resultado da Busca: Negativa	1.5
RF06	Resultado da Busca: Positiva	1.6

4.6. Formas de Mitigação de Ataques SQL *Injection*

O uso de uma classe para realizar a limpeza dos dados coletados é uma forma de realizar a mitigação, conforme apresentado na ClsSQLInjection(), a ser aplicado em todos os objetos que recebam um conteúdo alfanumérico nos *Forms* do sistema.

Uma outra forma é não utilizar o formato de concatenação de *String*s para gerar o comando SQL (Macoratti, 2008), mas enviar o conteúdo na forma de parâmetros (Microsoft, 2022d), como indicado na inserção do parâmetro Ds_Login no Código 4 em C#.

Código 4 – Uso do comando *parameter* em C# baseado em MACORATTI (2008) e Microsoft (2022d)

```
Dim mParam() As SqlParameter
...
SqlParameter mParam = new SqlParameter();
    parameter.ParameterName = "@Ds_Login";
    parameter.IsNullable = false;
    parameter.SqlDbType = SqlDbType.VarChar;
    parameter.Direction = ParameterDirection.Output;
    parameter.Size = 50;
…
    command.Parameters.Add(mParam);
```

4.7. Discussão

Foi possível identificar que a eliminação de possíveis comandos ou caracteres especiais é uma forma de mitigar os ataques de SQL *Injection*, mas não indica ser a forma completa. A alteração na forma de programação, com o uso de parâmetros, para a execução das consultas, as *queries*, corrobora para a solução.

O uso de procedimentos armazenados, as *Stored Procedures*, são recomendadas pelo fabricante que, tendo os comandos já pré-compilados, além da melhoria no desempenho, trazem a rigidez em relação aos dados informados pelo sistema.

5. Conclusões

O SQL *Injection* é um tipo de ataque que consiste em digitar códigos maliciosos em objetos que aceitam informações alfanuméricas em sistemas WEB, visando obter o login/senha de acesso e, desta forma, obter todas as informações da empresa.

A pesquisa apresentou a estrutura de uma classe, a ClsSQLInjection(), que visa o tratamento dos dados recebidos, a recomendação de usar o método *parameters()* ao criar as chamadas e o uso de *Stored Procedures* nas operações com os SGBD.

Em futuras trabalhos serão abordados o desenvolvimento de *Stored Procedures* para a retirada de caracteres especiais e a utilização diretamente no código ASPx dos *Forms*. A contribuição para a teoria foi de apresentar uma sugestão de mitigação dos ataques e que carecem de mais estudos. A contribuição para a prática é a de que programadores, analista e gestores de TI possam se beneficiar das recomendações desta pesquisa em sistemas em ambiente WEB.

Referencial Bibliográfico

Barcelos, A. K., Souza, C. L. S. de, Carmo, J. P. M. do, Faria, M. C., Alcântara, M., & Camargos, P. A. D. (2021). Lei Geral de Proteção de Dados e o Papel do DPO. *Revista Projetos Extensionistas*, *1*(2), 87–92.

Botelho, M. C., & Camargo, E. P. do A. (2021). O TRATAMENTO DE DADOS PESSOAIS PELO PODER PÚBLICO NA LGPD. *Revista Direitos Sociais e Políticas Públicas (UNIFAFIBE)*, *9*(3), 549–580.

Diana, A., Tojeiro, C., Cardoso, T. M., Lucas, T. J., & Moraes, E. A. (2016). COMPUTAÇÃO EM NUVEM - DISPONIBILIDADE: PESQUISA APLICADA NA FACULDADE DE TECNOLOGIA DE OURINHOS. *RETEC - Ourinhos*, *9*(2), 75–79.

Fernandes, N. O. C. (2013). *Segurança da Informação*. e-TEC. https://www.fatecourinhos.edu.br/retec/index.php/retec/article/view/214

Grassi, P. A., Garcia, M. E., & Fenton, J. L. (2017). *Digital Identity Guidelines* (p. 1–104). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-63-3

Hintzbergen, J., Hintzbergen, K., Smulders, A., & Baars, H. (2018). *Fundamentos de Segurança da Informação: Com base na ISO 27001 e na ISO 27002.* (3º ed). Brasport.

ISO. (2013). International Standard Organization—ISO/IEC 27001. Em *ISO*. International Standard Organization. https://www.iso.org/standard/54534.html

Kotler, P. (2021). *Marketing para o século XXI: como criar, conquistar e dominar mercados*. Alta Books.

Kotler, P., Kartajaya, H., & Setiawan, I. (2016). *Marketing 4.0: Moving from traditional to digital*. John Wiley & Sons.

Kotler, P., Kartajaya, H., & Setiawan, I. (2021). *Marketing 5.0: Tecnologia para a humanidade*. Sextante.

Macoratti, J. C. (2008). *Previna-se contra a Injeção SQL*. https://www.macoratti.net/sql_inj.htm

Microsoft. (2022a). *ASP.NET*. https://dotnet.microsoft.com/en-us/apps/aspnet

Microsoft. (2022b). *Documentação do C#*. https://docs.microsoft.com/pt-br/dotnet/csharp/

Microsoft. (2022c). *Estilo de arquitetura de N camadas*. https://learn.microsoft.com/pt-br/azure/architecture/guide/architecture-styles/n-tier

Microsoft. (2022d). *SqlParameter Construtores*. https://learn.microsoft.com/pt-br/dotnet/api/system.data.sqlclient.sqlparameter.-ctor?view=dotnet-plat-ext-6.0

Microsoft. (2022e, setembro 26). *Injeção de SQL*. Injeção de SQL. https://learn.microsoft.com/pt-br/sql/relational-databases/security/sql-injection?view=sql-server-ver16

Microsoft. (2022f, outubro 12). *Microsoft Lean—Install SQL Server from the Installation Wizard (Setup)*. https://learn.microsoft.com/en-us/sql/database-engine/install-windows/install-sql-server-from-the-installation-wizard-setup?view=sql-server-ver16

Nunamaker Júnior, J. F., Chen, M., & Purdin, T. D. M. (1990). Systems development in information systems research. *Journal of management information systems*, *7*(3), 89–106.

OMG. (2015). *OMG Unified Modeling Language TM (OMG UML)*. https://www.omg.org/spec/UML/About-UML/

Pressman, R., & Maxim, B. (2016). *Engenharia de Software Uma Abordagem Profissional* (8º ed). McGraw Hill Brasil.

Raes, A. (2022). *Procon/MS autua Leroy Merlin, Privália, James e Centauro por infração a LGPD*. Procon MS. https://www.procon.ms.gov.br/procon-ms-autua-leroy-merlin-privalia-james-e-centauro-por-infracao-a-lgpd/

Sadeghian, A., Zamani, M., & Manaf, A. Abd. (2013). A Taxonomy of SQL Injection Detection and Prevention Techniques. *2013 International Conference on Informatics and Creative Multimedia*, 53–56. https://doi.org/10.1109/ICICM.2013.18

Shar, L. K., & Tan, H. B. K. (2013). Defeating SQL Injection. *Computer*, *46*(3), 69–77. https://doi.org/10.1109/MC.2012.283

Sommerville, I. (2015). *Software Engineering* (10º ed). Pearson.

Theophilo, C. R., & Martins, G. de A. (2016). *Metodologia Da Investigação Cientifica* (3ª). Atlas.

Tipton, S. J., Forkey, S., & Choi, Y. B. (2016). Toward Proper Authentication Methods in Electronic Medical Record Access Compliant to HIPAA and C.I.A. Triangle. *Journal of Medical Systems*, *40*(4), 100. https://doi.org/10.1007/s10916-016-0465-x