TÉCNICAS DE DETECÇÃO E PREVENÇÃO DE ATAQUES WEB: UMA ABORDAGEM COM SQL INJECTION (SQLi) E CROSS-SITE SCRIPTING (XSS)

Universidade Presbiteriana Mackenzie

Coordenadoria de Fomento à Pesquisa - Pró-Reitoria de Pesquisa e Pós-Graduação

Naoto Ushizaki (IC)[1]

Apoio:PIVIC Mackenzie

Rodrigo Cardoso Silva (Orientador)[2]

Faculdade de Computação e Informática (FCI) [1,2]

<10437445@mackenzista.com.br>

<rodrigoc.silva@mackenzie.br>

2025

RESUMO

O estudo tem como objetivo aprimorar os sistemas de defesa de websites por meio de métodos de aprendizado de máquina não supervisionado para detectar ataques do tipo SQL Injection (SQLi) e Cross-Site Scripting (XSS). Utilizando Python e bibliotecas de inteligência artificial, desenvolveu-se um modelo de autoencoder treinado para identificar padrões anômalos em tráfego web. Foram aplicados datasets de domínio público para treinamento e validação. Os resultados indicaram que o modelo foi eficiente na detecção de SQL Injection, com alta taxa de recall e baixo índice de falsos negativos, embora tenha apresentado elevado número de falsos positivos na detecção de XSS. Conclui-se que a abordagem proposta é promissora, mas necessita de otimizações adicionais para ser aplicada em ambientes de produção.

Palavras-chave: Aprendizagem de Máquina Não Supervisionada de IA, *SQL Injection* e *XSS*, *Pytorch* e *Autoencoder*.

ABSTRACT

The study aims to improve website defense systems using unsupervised machine learning methods to detect SQL Injection (SQLi) and Cross-Site Scripting (XSS) attacks. Using Python and artificial intelligence libraries, an autoencoder model was developed to identify anomalous patterns in web traffic. Public domain datasets were used for training and validation. The results indicated that the model was effective in detecting SQL Injection, with a high recall rate and a low false negative rate, although it presented a high number of false positives in detecting XSS. The conclusion is that the proposed approach is promising but requires further optimization to be applied in production environments.

Keywords: *AI Unsupervised Machine Learning, SQL Injection and XSS, Pytorch and Autoencoder.*

1. INTRODUÇÃO

No início dos anos 2000, houve o desenvolvimento e mais estudos na área de Aprendizagem de Máquina ou *Machine Learning*, um subconjunto de Inteligência Artificial (IA), auxiliou na compreensão dos padrões típicos de tráfego e práticas de usuários em uma organização para poder identificar e responder a atividades incomuns.

A tecnologia vem em crescimento significativo e continua evoluindo em diversas áreas, entre elas, na segurança cibernética. Essa área, tem sido constantemente adaptada e aprimorada com o objetivo de detectar, neutralizar e impedir a invasão de ameaças digitais em sistemas *web* e na proteção de dados.

No entanto, com a constante evolução da IA, sua versatilidade se apresenta como um fator duplamente impactante: ao mesmo tempo em que é promissora para a área de segurança cibernética, também representa uma ameaça para qualquer tipo de ambiente digital. Os agentes maliciosos estão utilizando este recurso de algoritmos inteligentes para tornar seus ataques mais sofisticados, difíceis de serem detectados e automatizados, explorando vulnerabilidades com maior eficiência e em escala cada vez maior. Como por exemplo, para criação de múltiplos ataques de *malware* e maior agilidade no processo de exploração e identificação de vulnerabilidades (*YAMPOLSKIY*, 2024).

Entre esses ataques maliciosos destaca-se o *SQL Injection* e *Cross-Site Scripting* (XSS). Estes são tipos de invasores de websites, redes e banco de dados que utilizam o método de injeção de *script*, manipulando os comandos a favor do atacante. E caso o ataque tenha sucesso, podem resultar no roubo de dados críticos e valiosos. Deste modo, representando um risco significativo à confidencialidade dos dados e da sua integridade.

Neste contexto, torna-se relevante investigar a relação entre o avanço das técnicas de IA e as novas estratégias de ataque e defesa no campo da segurança na *web*, destacando a necessidade de soluções cada vez mais robustas, ágeis e eficientes para mitigar os riscos associados.

1.1. JUSTIFICATIVA DA PESQUISA

Com o crescimento da digitalização de serviços e a constante evolução das tecnologias web, vem sendo frequente o número de ameaças digitais em que causam muitos danos tanto para empresas quanto para usuários. Especialmente ataques como *SQL Injection* e *Cross-Site Scripting (XSS)* que representam riscos graves à integridade, confiabilidade e disponibilidade dos dados em ambientes *webs.*

Nesta situação, é essencial desenvolver e aperfeiçoar soluções capazes de detectar e prevenir essas ameaças de maneira eficiente e ágil. As aplicações de métodos de segurança tradicionais têm mostrado ser limitadas frente as abordagens sofisticadas adotadas por agentes maliciosos atualmente.

Neste contexto, o uso de técnicas de *Machine Learning (ML)* são fundamentais para serem utilizadas como uma abordagem promissora, proporcionando modelos capazes de aprender padrões anômalos, se adaptar à novas situações, realizar análises cruciais e oferecer soluções em tempo hábil.

Portanto, esta pesquisa se justifica pela necessidade de aprimorar mecanismos de segurança na web, através do uso da abordagem de aprendizado de máquina, com foco na detecção e prevenção de ataques *SQL Injection* e *XSS.*

1.2. MOTIVAÇÃO DA PESQUISA

O motivo para se realizar esta pesquisa foi para desenvolver um modelo não supervisionado que aplique técnicas de Machine Learning para detectar ataques do tipo *SQL Injection* e *Cross-Site Scripting* (XSS).

Nos últimos anos, houve um aumento de relatos de ataques do tipo *Cross-Site Scripting* (XSS) e *SQL Injection.* Um exemplo revelado pela Microsoft (CISO ADVISOR, 2023; CSA, 2023), a companhia da *Azure* relatou ter observado um atentado de captura de dados de seus ambientes de nuvem através de *SQL Servers* os quais são vulneráveis à técnica de *SQL Injection.* Mediante a essa estratégia, isso concede aos invasores o acesso a instâncias do *SQL Server* com permissões elevadas, sendo possível executar comandos *SQL* e a extração de dados valiosos, como permissões de acesso, estrutura de banco de dados e configurações de rede. E, outra vulnerabilidade que podem explorar é a identidade de nuvem da instancia do *SQL Server* concedendo assim acesso a serviços de metadados instantâneos (IMDS) e adquirir chave de acesso da identidade de nuvem.

Um caso ocorreu em 2023, onde um grupo de segurança chamada *Akamai Security Intelligence Group (SIG)* esteve analisando uma atividade de tentativa de ataque em uma vulnerabilidade crítica em um *plugin* de campos personalizados do *WordPress* que afetava mais de 2 milhões de *websites*. A falha permitia ataques de *Cross-Site scripting (XSS)* refletida, nos quais códigos são injetados em websites vulneráveis e repassados diretamente aos visitantes.

Outro exemplo ocorreu com o comprometimento de websites de recrutamento e varejo. Houve o roubo de mais de 2 milhões de endereços de *e-mail* e outras informações pessoais de pelo menos 65 websites, segundo *Group-IB* (CISO ADVISOR, 2024). A invasão foi feita a partir do uso de *scripts XSS* injetados em *websites* legítimos de procura de empregos, com o intuito de exibir formulários *phishing* e obter credenciais administrativas.

Esses ataques não se limitaram somente à plataforma Azure e a *websites* de recrutamento. Em outro contexto, também foi relatado um caso na Romênia, em que a *Romanian Intelligence Service* (SRI) descreve ter sofrido por volta de 85.000 tentativas de ataques com o objetivo de ganhar acessos para a infraestrutura da eleição, que ocorria no país naquele momento, para alterar informações da eleição para o público e negando acessos ao sistema. Foi reportado também que houve tentativas de invasão de sistemas através da exploração de vulnerabilidades com *SQL Injection* e *Cross-Site Scripting* (XSS) de dispositivos em mais de 33 países (BBC, 2024; BLEEPING COMPUTER, 2024).

Considerando os ataques mencionados anteriormente e constante evolução de técnicas pelos atacantes, foi necessário investigar modelos de aprendizado de máquina não supervisionado, que permitem identificar padrões e estruturas anômalos e novas formas de ataque sem a necessidade de dados previamente rotulados. Esse método é vantajoso em ambientes que possui ataques bem variados ou em rápidas mudanças. Isto possibilita uma resposta mais escalável e flexível na detecção e prevenção de ameaças do tipo *SQL Injection* e *Cross-Site Scripting* (XSS).

Em conclusão, os casos citados anteriormente, reforçam a gravidade e o impacto que técnicas como *SQL Injection* e *Cross-Site Scripting (XSS)* podem causar, representando sérias ameaças tanto para aplicação web quanto para infraestruturas críticas.

1.3. OBJETIVOS DA PESQUISA

Desenvolver um modelo de aprendizado de máquina não supervisionado para detectar ataques de SQL Injection e Cross-Site Scripting (XSS), visando aprimorar a eficiência dos

sistemas de segurança, a confiabilidade e integridade dos dados, bem como o tempo de resposta dos mecanismos de defesa em ambientes e serviços web.

2. REFERENCIAL TEÓRICO

A análise de ameaças de aplicações web, especialmente relacionadas a SQL *Injection* e *Cross-Site Scripting (XSS),* têm sido discutidos na literatura técnica e científica. Ambas as falhas, segundo a OWASP (2023), estão entre as 10 principais vulnerabilidades de segurança em aplicações web com potencial de comprometimento de dados e violação das políticas de segurança.

Em 2002, um dos primeiros estudos feitos relacionado a ataques SQL Injection foi realizado por Mcdonald (2002). No artigo, o autor executa experimentos práticos para analisar e demonstrar como comandos maliciosos podem ser injetados em consultas SQL e manipulados para obter dados sensíveis. Além de mapear tipos de injeção mais comuns e avançados, foi proposto estratégias de defesa e o uso de instruções parametrizadas e validação de entradas.

Além desse artigo, o estudo conduzido por Tramontana et al. (2005), apresenta detalhes e análises sobre a identificação de vulnerabilidades de *Cross-Site Scripting (XSS)* em aplicações web, com o objetivo de sanitização de entrada de dados em aplicações web. A abordagem proposta consistiu na análise do código-fonte da aplicação, na realização de testes automatizados com entradas maliciosas e verificação de respostas de aplicação que refletissem scripts sem a devida sanitização, confirmando uma vulnerabilidade XSS. O estudo desenvolveu uma ferramenta para automatizar parte do processo de detecção da vulnerabilidade e testes foram realizados em sistemas de universidades e empresas. Com isso, foi revelado que muitas aplicações apresentavam falhas, mesmo utilizando validações básicas.

Mais recentemente, abordagens utilizando o aprendizado de máquina (ML) têm sido utilizadas como alternativa promissora para detecção eficiente e automatizada de ameaças web. Em relação ao ataque SQL *Injection,* Krishnan et al. (2021) aplicou 5 técnicas de ML, tanto supervisionada quanto não supervisionada. Como resultado, o estudo revelou as pontuações de cada método e evidenciou que, por contarem com dados rotulados e tendo uma otimização guiada, alcançaram métricas de desempenho elevadas. Porém, essa abordagem pode ser desvantajosa caso os dados estejam induzindo a reconhecer apenas padrões específicos de brechas, tornando-o menos eficaz na detecção de ataques que divergirem do padrão aprendido.

Conclui-se, portanto, que embora abordagens supervisionadas tenham demonstrado um elevado desempenho na detecção de ameaças conhecidas, sua eficácia depende da qualidade e

diversidade dos dados de treinamento. Nesse contexto, é importante considerar a possibilidade de ameaças *Zero Day, (0-day)* que não seguem padrões previamente analisados. Assim, isso reforça a importância de desenvolver e aprimorar um modelo de aprendizado não supervisionada, de forma a ampliar a cobertura na detecção das ameaças web *SQL Injection* e *Cross-Site Scripting,* mesmo diante de ameaças desconhecidas.

2.1. REVISÃO DA LITERATURA

O artigo *Prevention of SQL Injection attack using unsupervised Machine Learning Approach* (KHAVITHA et. al, 2021) aplicou um método de aprendizado de máquina não supervisionado utilizando o algoritmo de agrupamento de padrões de entrada em clusters, conhecida como *K-Means Clustering.* O estudo coletou dados contendo padrões de ataque do tipo *SQL Injection* que foram usadas para treinar o modelo. Além *K-Means*, foi utilizado o NLTK (*Natural Language Tool Kit*) para extrair e selecionar partes essenciais das *queries* para o aprendizado da máquina. Como resultado, os *clusters* conseguiram identificar com sucesso *que*ries maliciosos, atingindo uma precisão ótima.

O artigo *SQL Injection detection Using Machine Learning* (KHISHNAN et. al., 2021) pesquisa sobre abordagens da aplicação de algoritmos de aprendizado de máquina com o objetivo de identificar ataque de injeção de SQL. Este estudo implementa e compara diversos algoritmos de aprendizagem de máquina supervisionado com a função de classificar tráfego como malicioso ou legítimo. Através desta abordagem, destacou-se a habilidade de identificar padrões complexos no tráfego de dados e foi oferecido uma aproximação promissora para aperfeiçoar a segurança de aplicações de web.

O artigo *Detection of SQL Injection Attack Using Machine Learning Techniques: A Systematic Literature Review* (ALGHAWAZI et. al., 2022) realizou uma revisão sistemática sobre a utilização de estratégias de máquina voltada na identificação de padrão de ataques do SQL Injection. Esta pesquisa inclui análise e comparação de modelos algorítmicos e tipos de datasets utilizados em outros artigos tecnológicos. Exibe os resultados da eficiência dos algoritmos, a importância da seleção dos parâmetros de avaliação, desafios encontrados e tendências futuras.

O artigo *Deep Learning Architecture for Detecting SQL Injection Attacks Based on RNN Autoencoder Model* (ALGHAWAZI et al., 2023) propõem uma abordagem baseada em aprendizado profundo para detector ataques de injeções SQL. Seu objetivo foi desenvolver e avaliar um modelo baseado em RNN Autoencoder capaz de identificar padrões maliciosos em

comandos SQL com alta precisão e baixo índice de falsos positivos. O modelo atingiu uma precisão de 94% e F1-score de 92%. Um resultado que não só evidencia sua eficiência na detecção de ataques SQL, mas também mostra o potencial superior das arquiteturas Autoencoders em relação às abordagens tradicionais.

O artigo *Machine Learning Approaches for Detection of SQL Injection Attacks* (ANWAR et.al., 2025) realiza a avaliação da eficácia de modelos de aprendizado de máquina na detecção de *SQL Injection*. Neste estudo foram avaliados os modelos *Support Vector Machines* (SVM*), Deep Neural Networks* (DNN) e *Random Forest* (RF). E, foi utilizado um conjunto de dados contendo requisições normais e maliciosas relacionadas a *SQL Injection*. A performance dos modelos é analisada em relação à métricas como precisão, acurácia, *recall* e *F1-score.* Os resultados demonstraram que os três modelos apresentaram um desempenho satisfatório, sendo considerados fortes candidatos para a detecção de *SQL Injection*. A escolha entre elas deve ser baseada em fatores como nível de interpretabilidade, disponibilidade de recursos computacionais e relevância do poder discriminativo para a aplicação específica.

O artigo *Critical Evaluation of SQL Injection Security Measures in Web Applications* (ATHAB, 2025) oferece uma análise das vulnerabilidades do *SQL Injection* com um foco em vetores de ataque via URL o qual contribui para a o campo de segurança na aplicação *web*. Esta pesquisa classifica e define padrões de ataques comuns - como manipulação de consultas (*queury manipulation*) e XSS (*Cross-Site Scripting*) – com o intuito de demonstrar como ataques maliciosos podem explorar parâmetros de entrada contra a integridade das aplicações. Além disso, são feitas críticas sobre as medidas de segurança como o *Boyer-Moore*, que é um algoritmo de correspondência de padrões, e as técnicas de validação de entrada no lado do servidor, assim avaliando sua eficácia na defesa contra ataques de *SQL Injection.*

O artigo *Malicious Query Recognition Using Chosen Machine Learning Techniques* (DARAMOLA, C.Y. et. al., 2025) realiza uma pesquisa voltada à implementação e treinamento de um algoritmo de aprendizado de máquina com o objetivo de detectar e classificar *queries* maliciosos em aplicações *web* e sistemas de banco de dados utilizando técnicas de *machine learning*. O foco principal é a identificação de padrões associados aos ataques do tipo *SQL Injection*. Para isso, o estudo realizou a coleta de dados de vários *sites* e fontes (*MySQL, PostgreSql, Sqlite, Github*) e foi montado um *dataset* contendo *queries* avançados. Após a modelagem de dados, foram aplicadas técnicas como o SVM (*Support Vector Machine*), *Naïve Bayes Classifier*, *Random Forest* e *Multi-Layer Perceptron* (MLP) e foram avaliados segundo sua eficiência em detecção automática de *queries* maliciosos.

3. MÉTODO APLICADO

A metodologia adotada para esta pesquisa foi a realização de testes e no treinamento de um algoritmo de aprendizado de máquina não supervisionada com o objetivo de detectar *queries* maliciosos, como *SQL Injection* e *Cross-Site Scripting* (XSS), em aplicações webs. Para isso, foram utilizados dados provenientes de datasets disponíveis, os quais serviram como base para o treinamento e avaliação do modelo.

3.1 POR QUE UM MODELO MÁQUINA NÃO SUPERVISIONADA?

Diferentemente dos modelos supervisionados, cuja eficácia está ligada à qualidade e diversidade dos dados rotulados utilizados no treinamento, a técnica de aprendizado não supervisionado explora padrões fora do comum, sem depender de exemplos previamente classificados. E segundo Augustine et al. (2024), essa capacidade de explorar comportamentos fora do padrão torna possível detectar tipos de ataques ainda não descobertos ou previamente analisados. Sendo assim, uma abordagem promissora e ágil para detectar invasão de ataques, como SQL Injection e XSS, mesmo quando se apresentam de forma incomum.

3.2. MODELO UTILIZADO

O modelo foi *o Autoencoder.* Segundo Pius (2023) e Barnan (2022) o *Autoencoder* é um tipo de rede neural artificial utilizado para aprender representações dos dados de forma não supervisionada. Seu objetivo é aprender uma representação de menor dimensão (*encoder*) de dados de maior dimensionalidade, geralmente para reduzi-la, treinando a rede para capturar partes mais importantes da entrada sem a necessidade de rótulos. Após essa compressão, o algoritmo reconstrói os dados originais, buscando minimizar a perda de reconstrução. Dessa forma, é capaz de destacar aspectos mais relevantes dos dados, permitindo a diferenciação entre amostras normais e potenciais ataques.

3.3 PROCESSAMENTO E SEPARAÇÃO DE DADOS

Foi utilizado o *StandardScaler* para processar os dados dos *datasets* que, segundo Pedregosa et al. (2011), padroniza os recursos removendo a média e dimensionando a variância unitária. Ou seja, transforma valores de cada *feature* para que tenham média zero e desvio padrão igual a um.

A separação de dados foi utilizada do pacote *sklearn.model_selection* o *train_test_split.* Um método que divide *arrays* ou matrizes em subconjuntos aleatórios de treinamento e teste. Para este experimento, foi utilizado 80% dos dados para treino e 20% para testes.

4. RESULTADO E DISCUSSÃO

Esta pesquisa foi realizada por conta do aumento significativo dos crimes cibernéticos, especialmente de ataques do tipo *SQL Injection* e *XSS*, que têm afetado aplicações *web* nos últimos anos (SECURITY LEADERS, 2024). Com o uso crescente de inteligência artificial por atacantes para aperfeiçoar seus métodos e ferramentas de invasão, os sistemas de segurança tradicionais tornam-se cada vez mais obsoletos, incapazes de detectar ameaças em tempo hábil, o que pode resultar em prejuízos operacionais e comprometimento de dados.

Dessa forma, o objetivo da pesquisa foi desenvolver um modelo aprendizado de máquina não supervisionado que detecte ameaças do tipo *XSS* e *SQL Injection*. E com a utilização de *datasets* coletados em *Githubs* e *Kaggle que* possibilitou a realização de treinamentos e testes.

Por fim, destaca-se que o objetivo de alcançar resultados promissores em termos de precisão e *F1-score,* na detecção de ameaças foi parcialmente atingido. Entretando, este estudo fornece dados, métodos e resultados detalhados que podem subsidiar pesquisas e trabalhos futuros, além de apresentar uma margem significativa para aprimoramentos e aprofundamentos.

4.1. AMBIENTE DE TESTE PARA SQL INJECTION

Os experimentos foram realizados no *Google Colab*. O conjunto de dados SQL que foram utilizados são *SLQ Injection Attack for training (D1)* e *SQL Injection Attack for Test (D2)* **(**CAMPAZAS; CRESPO, 2022**).**

Abaixo estão as bibliotecas que foram utilizadas em ambos os modelos:

- Importar arquivos/*datasets*: *files* do *google.colab*
- Abrir arquivos CSV: *pandas*
- Métricas de avaliação: *classification_report* e *confusion matrix*
- Separação dos dados: *train_test_split*
- Pré-processamento: *StandardScaler*
- Do *tensorflow.keras.models*: *Model*
- Do *tensorflow.keras.layers*: *Input*, *Dense*

4.1.1. MANIPULAÇÃO DO DATASET

As colunas são separadas em X e y para organizar quais serão os dados de treinamento e testes, tanto para o eixo x, quanto ao eixo y do *dataset.* E neste caso, a coluna que apresenta dados maliciosos ou normais é a coluna *Label* que será armazenada em y. E em X será armazenado o resto das colunas, que são as colunas numéricas, essenciais para o processo de avaliação e treinamento ao modelo. E os dados () são separados em treinos e teste para ambos os eixos (X e y) e o *test_size* organiza-os para isolar 20% do conjunto de informações para testes e o resto para treinamento. Esta abordagem está disponível em: https://github.com/mackcoder/SQL-Injection-Detection-using-Autoencoder-Vanila-in-Python.-/commit/ab5a69c986b0a598407bfcc903d15e2dc25e486f#diff-7d990de1d6767cb02db983227bfd61e2e864b3eef1c999ad6b7b5cf66d7e9b88.

4.1.2. CRIANDO E TREINANDO AUTOENCODER

Antes de testar o modelo a dados mistos, somente os dados normais são utilizados para o treinamento do modelo e permite que o modelo aprenda padrões legítimos de comportamento. Para preservar a integridade estatística dessas informações, aplicou-se o processo de normalização utilizando o pré-processador *StandardScaler*, da biblioteca *sklearn.preprocessing*.

Para a etapa de detecção de anomalias, foi implementado um modelo de *autoencoder* com arquitetura simétrica e função de ativação *ReLU (*VARTOUNI, M. et al., 2018*)*. A estrutura do autoencoder consistiu em duas camadas de codificação (com 16 e 8 neurônios, respectivamente) e duas camadas de decodificação (com 16 neurônios e uma camada final com dimensão igual à entrada). Nesta parte, foi possível realizar somente o treinamento do modelo com dados normais, com o objetivo de o modelo analisar os dados originais e iniciar o processo de reconstrução desses mesmos dados. E durante este processo, o erro quadrático médio (MSE) foi calculado e serviu como métrica para identificar desvios em relação ao comportamento aprendido.

Para definir o limiar de detecção (*threshold*), foi realizada uma varredura entre as porcentagens entre 70 e 98 do erro de reconstrução das amostras normais (MAC, H. et al., 2018). Para cada valor de *threshold*, foi gerada uma classificação binária: amostras com erro acima do limiar foram rotuladas como ataques, e abaixo como normais. A métrica de *recall* para a classe "Ataque" foi utilizada como critério principal para selecionar o melhor *threshold*. A implementação pode ser consultada no seguinte repositório do *GitHub*:

https://github.com/mackcoder/ML-models-AutoencoderVanila-in-Python.-/blob/main/Main-SQL-Autoencoder

4.1.3. RESULTADOS DA ML NA DETECÇÃO DE SQL INJECTION

A Tabela 1 demonstra que os resultados obtidos são bem expressivos na detecção de ataques. A classe "Normal" apresentou uma precisão de 1.00, o que significa que todas as amostras classificadas como normais realmente pertencem a essa categoria. Na classe *Recall* exibiu uma nota de 0.7 apontando que 0.3 das amostras normais foram equivocadamente identificas como ataques. Por outro lado, a classe "Ataque" obteve *recall* máximo (1.00), ou seja, todos os ataques foram corretamente detectados, embora com uma precisão de 0.77, indicando que algumas amostras foram classificadas como ataques. Já o equilíbrio entre essas métricas se baseia nos valores de *F1-score*: 0.82 para Normal e 0.87 para Ataque, evidenciando que o modelo priorizou a detecção completa de ameaças, mesmo ao custo de alguns falsos positivos.

	Precision	***Recall***	***F1-score***	***Support***
Normal	**1.00**	**0.70**	**0.82**	**40000**
Ataque	**0.77**	**1.00**	**0.87**	**40001**

Tabela 1

4.2. AMBIENTE DE TESTE PARA XSS

Os experimentos foram realizados no *Google Colab*. O conjunto de dados XSS que foram utilizados são **(**MEREANI, F. A.; HOWE, J. M. 2018**)**. E para o XSS foram utilizados *XSStraining e XSStesting* ()

Abaixo estão as bibliotecas que foram utilizadas em ambos os modelos:

- Importar arquivos/*datasets*: *files* do *google.colab*
- Abrir arquivos CSV: *pandas*
- Métricas de avaliação: *classification_report* e *confusion matrix*
- Separação dos dados: *train_test_split*
- Importou *numpy*
- Importou *tensorflow*

- Pré-processamento: *StandardScaler*
- Do *tensorflow.keras.models*: *Model*
- Do *tensorflow.keras.layers*: *Input*, *Dense*

4.2.1. CRIANDO E TREINANDO AUTOENCODER

Neste modelo, a manipulação de dados e o experimento realizado foi semelhante ao que foi feito com o modelo detector de *SQL Injection*, utilizando o mesmo pré-processador e separação de dados.

O processo para a detecção de anomalias em ataques XSS foi implementado o modelo de *autoencoder* com arquitetura simétrica e função de ativação *ReLU (*VARTOUNI, M. et al., 2018*)*. A estrutura do modelo contou com duas camadas de codificação (com 16 e 8 neurônios, respectivamente) e duas camadas de decodificação (com 16 neurônios e uma camada final com dimensão igual à entrada). O treinamento foi realizado exclusivamente com amostras normais, permitindo que o modelo aprendesse os padrões originais de entrada e iniciasse o processo de reconstrução dessas informações. E durante o processo de ciclos (*Epoch*), o erro quadrático médio (MSE) foi utilizado como métrica para quantificar desvios em relação ao comportamento aprendido. Para definir o limiar de detecção (*threshold*) (MAC, H. et al., 2018), foi criado um *loop* para verificar e testar os percentuais entre 70 e 98 do erro de reconstrução das amostras normais. Para cada valor testado, foi gerada uma classificação binária: amostras com erro acima do limiar foram rotuladas como ataques, e abaixo como normais. A métrica de *recall* da classe Ataque foi adotada como critério principal para selecionar o melhor *threshold*. A implementação completa pode ser consultada no seguinte repositório do GitHub: https://github.com/mackcoder/ML-models-AutoencoderVanila-in-Python.-/blob/main/Main-XSS-Autoencoder

4.2.2. AVALIAÇÃO DOS RESULTADOS DA ML NA DETECÇÃO DE XSS

A Tabela 2 representa a avaliação do conjunto de teste original. Neste conjunto, o modelo atingiu um *recall* de 1.00 na classe Ataque, porém com uma precisão de 0.56, por conta de uma taxa alta de falsos positivos (807). A acurácia geral foi de 79% com destaque para o *F1-score* da classe Ataque (0.72).

	Precision	***Recall***	***F1-score***	***Support***
Normal	**1.00**	**0.71**	**0.83**	**2795**
Ataque	**0.56**	**1.00**	**0.72**	**1030**

Tabela 2

A Tabela 3 representa a avaliação do conjunto de teste do *dataset XSSTesting* contendo amostras maliciosos. O modelo manteve o recall de 0.999, detectando quase todos os ataques, porém com uma taxa considerável de falsos positivos (6087) o que impactou a precisão da classe Ataque.

	Precision	***Recall***	***F1-score***	***Support***
Normal	**1.00**	**0.57**	**0.72**	**14096**
Ataque	**0.62**	**1.00**	**0.77**	**10000**

Tabela 3

4.3. RECOMENDAÇÕES PARA MELHORES RESULTADOS

Embora as métricas de precisão, *recall*, *F1-score* e acurácia tenham exibido um desempenho razoável na detecção, tanto de *SQL Injection e XSS.* os resultados ainda não são suficientemente robustos para que o modelo seja considerado confiável em ambientes de produção *web*.

Portanto, para melhorar os resultados é recomendável que:

- Aprimorar o pré-processamento dos dados com a remoção de ruídos e atributos irrelevantes, junto com a melhoria na normalização das *features.*
- Testar diferentes tamanhos e profundidades no *Autoencoder* (camadas/ *neurons* mais profundas podem aprender representações mais complexas).
- Aplicar métodos ROC *curve* ou análise de distribuição para definir o limiar ideal, ao invés de usar percentis fixos no *threshold*.
- Realizar testes do modelo em tráfego real com o objetivo de medir o tempo de execução e analisar seus resultados.
- Criação de uma *pipeline* para retreinar o modelo com dados novos.

- Procurar métodos e monitoração de métricas e algoritmos que ajustem o modelo conforme o necessário para obter resultados concretos.

5. CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

O estudo realizado demonstrou a viabilidade da utilização de modelos de aprendizado não supervisionado, como *autoencoders*, para a detecção de ataques do tipo *SQL Injection e XSS*. Ao treinar o modelo com os *datasets* disponibilizados, manipular os dados e treinar o algoritmo de máquina, foi possível identificar padrões legítimos de comportamento e, com base no erro de reconstrução, distinguir amostras maliciosas com alto grau de sensibilidade. Portanto os resultados obtidos evidenciaram um desempenho razoável em termos de *recall* para a classe Ataque, o qual é importante em cenários de segurança, embora a precisão tenha sido impactada pela ocorrência de falsos positivos.

Apesar de os testes não tenham sido realizados com tráfego real, a limitação de tempo impediu uma maior diversificação dos conjuntos de dados e uma otimização mais refinada dos modelos, o que poderia ter contribuído para resultados mais robustos. Por fim, com os experimentos realizados e dados exibidos nesta pesquisa, espera-se que este trabalho sirva de base para estudos futuros voltados ao aprimoramento de técnicas de detecção de ataques cibernéticos, especialmente aquelas baseadas em ML e IA para a segurança cibernética.

6. REFERÊNCIAS

ESTER, Martin. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. kdd, v. 96. n. 34, 1996. Disponível em: https://cdn.aaai.org/KDD/1996/KDD96-037.pdf. Acesso em: 10 ago. 2025.

FORD, Vitaly; SIRAJ, Ambareen. Applications of machine learning in cyber security. Disponível em: https://vford.me/papers/Ford%20Siraj%20Machine%20Learning%20in%20Cyber%20Security%20final%20manuscript.pdf. Acesso em: 10 set. 2024.

ZHOU, Qianru; PEZAROS, Dimitrios. Evaluation of machine learning classifiers for Zero-Day intrusion detection – An analysis on CIC-AWS-2018 dataset. mai. 2019. Disponível em: <Evaluation-of-Machine-Learning-Classifiers-for-Zero-Day-Intrusion-Detection--An-Analysis-on-CIC-AWS-2018-dataset.pdf>. Acesso em: 17 set. 2024.

JUNG, Wookhyun et al. Poster: Deep learning for Zero-day Flash malware detection. Disponível em: https://www.covert.io/research-papers/deep-learning-security/Poster%20-%20Deep%20Learning%20for%20Zero-day%20Flash%20Malware%20Detection.pdf. Acesso em: 07 out. 2024.

RAO, Delip; MCMAHAN, Brian. Natural language processing with PyTorch. Sebastopol: O'Reilly, fev. 2019. Disponível em: https://www.oreilly.com/library/view/natural-language-processing/9781491978221/. Acesso em: 10 out. 2024.

ALI, Shamshair et al. Comparative evaluation of AI-based techniques for zero-day attacks. Electronics, Basel, v. 11, n. 23, p. 3934, nov. 2022. Disponível em: https://www.mdpi.com/2079-9292/11/23/3934. Acesso em: 02 out. 2024.

PATTEWAR, Tareek. Detection of SQL injection using machine learning: a survey. International Research Journal of Engineering and Technology, nov. 2019. Disponível em: https://www.academia.edu/download/61273198/IRJET-V6I114220191119-123579-856wj2.pdf. Acesso em: 05 nov. 2024.

KRISHNAN, S. S. Anandha et al. SQL injection detection using machine learning. Revista GEINTEC, 2021. Disponível em: https://revistageintec.net/old/wp-content/uploads/2022/02/1939.pdf. Acesso em: 13 fev. 2024.

LEHR, J. et al. Supervised learning vs. unsupervised learning: a comparison for optical inspection applications in quality control. IOP Conference Series: Materials Science and Engineering, v. 1140, 2021. Disponível em: https://iopscience.iop.org/article/10.1088/1757-899X/1140/1/012049/pdf. Acesso em: 14 fev. 2024.

ALGHAWAZI, Maha. Detection of SQL injection attack using machine learning techniques: a systematic literature review. Digital, Basel, v. 2, n. 4, p. 39, set. 2022. Disponível em: https://www.mdpi.com/2624-800X/2/4/39. Acesso em: 16 fev. 2025.

BANNISTER, Adam. Business email platform Zimbra patches Memcached injection flaw that imperils user credentials. The Daily Swig, jun. 2022. Disponível em: https://portswigger.net/daily-swig/business-email-platform-zimbra-patches-memcached-injection-flaw-that-imperils-user-credentials. Acesso em: 05 mar. 2025.

JEMAL, Ines et al. SQL injection attack detection and prevention techniques using machine learning. Disponível em: https://www.researchgate.net/profile/Omar-Cheikhrouhou/publication/342734749_SQL_Injection_Attack_Detection_and_Prevention_Te

chniques_Using_Machine_Learning/links/5f0415d44585155050 91b1ec/SQL-Injection-Attack-Detection-and-Prevention-Techniques-Using-Machine-Learning.pdf. Acesso em: 25 jul. 2025.

TASEER, Muhammad; GHAFORY, Hamayoon. SQL injection attack detection using machine learning algorithm. Mesopotamian Journal of Cybersecurity, v. 2, n. 1, fev. 2022. Disponível em: https://journals.mesopotamian.press/index.php/CyberSecurity/article/download/24/35. Acesso em: 10 mar. 2025.

SENOUCI, Oussama; BENAOUDA, Nadjib. Advanced deep learning framework for detecting SQL injection attacks based on GRU model. ResearchGate, mar. 2022. Disponível em: https://www.researchgate.net/publication/359468095_Deep_Neural_Network-Based_SQL_Injection_Detection_Method. Acesso em: 08 mai. 2025.

Hackers atacam Azure por meio de servidores SQL violados. CisoAdvisor, dez. 2023. Disponível em: https://www.cisoadvisor.com.br/hackers-atacam-azure-por-meio-de-servidores-sql-violados/. Acesso em: 14 mai. 2025.

Milhões de registros são roubados de sites por injeção SQL. CisoAdvisor, fev. 2024. Disponível em: https://www.cisoadvisor.com.br/milhoes-de-registros-sao-roubados-de-sites-com-injecao-de-sql/. Acesso em: 16 abr. 2025.

TADHANI, J. R. et al. Securing web applications against XSS and SQLi attacks using a novel deep learning approach. Scientific Reports, [s.l.], jan. 2024. Disponível em: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10799887/. Acesso em: 18 abr. 2025.

YAMPOLSKIY, Aleksandr. What does 2024 have in store for the world cybersecurity? World Economic Forum – WeForum.org. 2024. Disponível em: https://www.weforum.org/stories/2024/02/what-does-2024-have-in-store-for-the-world-of-cybersecurity/. Acesso em: 09 jun. 2025.

HUONG LE, Thi-Thu et al. Enhancing SQL injection detection with trustworthy ensemble learning and boosting models using local explanation techniques. Preprints.org, out. 2024. Disponível em: https://www.preprints.org/manuscript/202410.1878/download/final_file. Acesso em: 18 abr. 2025.

DASARI, Naga Sai. Enhancing SQL injection detection and prevention using generative models. arXiv, fev. 2025. Disponível em: https://arxiv.org/abs/2502.04786. Acesso em: 19 abr. 2025.

MALLAIAH, Shivamurthaiah. Emerging research trends in computer science and information technology. ResearchGate, fev. 2025. Disponível em: https://www.researchgate.net/profile/Joyanto-Roychoudhary/publication/389660477_Emerging_Research_Trends_in_Computer_Science_and_Information_Technology_ISBN_978-93-48620-71-2/links/67cbc88bcc055043ce6f45eb/Emerging-Research-Trends-in-Computer-Science-and-Information-Technology-ISBN-978-93-48620-71-2.pdf#page=54. Acesso em: 28 jun. 2025.

ANWAR, Ican. Machine learning approaches for detection of SQL injection attacks. *JAFOTIK* – Jurnal Farmasi dan Teknologi Informasi Kesehatan, v. 1, n. 1, fev. 2025. Disponível em: https://journal.lenterailmu.com/index.php/jafotik/article/download/50/57. Acesso em: 29 jun. 2025.

DARAMOLA, C. Y. et al. Malicious query recognition using chosen machine learning techniques. SN Computer Science, v. 6, 281, 2025. Disponível em: https://doi.org/10.1007/s42979-025-03745-4. Acesso em: 08 jul. 2025.

ZHANG, Wei et. al. Deep Neural Network-Based SQL Injection Detection Method. *Wiley Online Library.* Disponível em: https://onlinelibrary.wiley.com/doi/pdf/10.1155/2022/4836289. Acesso em: 21 jul. 2025.

DARAMOLA, C.Y., AKINPELU, S.A., AKINYEMI, E.O. et al. Malicious Query Recognition Using Chosen Machine Learning Techniques. SN COMPUT. SCI. **6**, 281 (2025). Disponível em: https://doi.org/10.1007/s42979-025-03745-4

MEREANI, F. A.; HOWE, J. M. (2018). Detecting Cross-Site Scripting Attacks Using Machine Learning. In Advanced Machine Learning Technologies and Applications, volume 723 of AISC, pages 200–210. Springer. Disponível em: https://github.com/fmereani/Cross-Site-Scripting-XSS/tree/master/XSSDataSets. Acesso em: 28 jul. 2025.

CAMPAZAS, A.; CRESPO, I. SQL Injection Attack for Training (D1). Zenodo, 26 jul. 2022. Disponível em: https://zenodo.org/records/6906893. Acesso em: 19 jul. 2025.

CAMPAZAS, A.; CRESPO, I. SQL Injection Attack for Test (D2). Netflow. Zenodo, 26 jul. 2022. Disponível em: https://zenodo.org/records/6907252. Acesso em: 19 jul. 2025.

BERGMANN, Dave; STRYKER, Cole. O que é um autoencoder (autocodificador)? IBM, 2023. Disponível em: https://www.ibm.com/think/topics/autoencoder. Acesso em: 29 jul. 2025.

PEDREGOSA, Fabian et al. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, v. 12, p. 2825–2830, 2011. Disponível em: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html. Acesso em: 01 ago. 2025.

HOPHR. How to implement and scale up autoencoders in TensorFlow for anomaly detection or data generation tasks. HopHR, s.d. Disponível em: https://www.hophr.com/tutorial-page/implement-and-scale-up-autoencoders-in-tensorflow-for-anomaly-detection-or-data-generation-tasks. Acesso em: 03 ago. 2025.

VISHWAKARMA, Swapnil. How to understand Sigmoid Function in Artificial Neural Networks? Analytics Vidhya. 09 dez. 2024. Disponivel em: https://www.analyticsvidhya.com/blog/2023/01/why-is-sigmoid-function-important-in-artificial-neural-networks/#h-what-sigmoid-function-do. Acesso em: 01 ago. 2025.

THAKUR, Ayush. ReLU vs. Sigmoid Function in deep Neural Networks. Weights & Biases. 19 ago. 2022. Disponível em: https://wandb.ai/ayush-thakur/dl-question-bank/reports/ReLU-vs-Sigmoid-Function-in-Deep-Neural-Networks--VmlldzoyMDk0MzI. Acesso em: 01 ago. 2025.

ALGHAWAZI, Maha.; ALGHAZZAWI, Daniya.; ALARIFI, Suaad. Deep Learning Architecture for Detecting SQL Injection Attacks Based on RNN Autoencoder Model. Mathmatics. 2023. Disponível em: https://www.mdpi.com/2227-7390/11/15/3286. Acesso em: 10 ago. 2025.

CLOUDERA FAST FORWARD LABS. Deep Learning for anomaly detection. 2020. Disponível em: https://ff12.fastforwardlabs.com/. Acesso em: 08 ago. 2025.

AUGUSTINE, Nwabudike.; SULTAN, A.B.M.; OSMAN, M.H.; SHARIF, K.Y. Application of Artificial Intelligence in detecting SQL Injection Attacks. Internacional Journal On Informatics Visualization. 22 out. 2024. Disponível em: https://joiv.org/index.php/joiv/article/download/3631/1136. Acesso em: 14 ago. 2025.

CLOUDERA. Deep learning for anomaly detection. Blog, 28 jan. 2020. Disponível em: https://www.cloudera.com/blog/technical/deep-learning-for-anomaly-detection.html. Acesso em: 13 ago. 2025.

SECURITY LEADERS. Ataques contra aplicações e APIs cresceram 49% no último ano. 14 ago. 2024. Disponível em: https://securityleaders.com.br/ataques-contra-aplicacoes-e-apis-cresceram-49-no-ultimo-ano/. Acesso em: 02 ago. 2025.

PIUS, Abish. Introduction to all types of Autoencoders with python. Medium. 21 mar. 2023. Disponível em: https://medium.com/chat-gpt-now-writes-all-my-articles/introduction-to-all-types-of-autoencoders-with-python-ec0e47b5e1b9. Acesso em: 15 ago. 2025.

MAC, Hieu; TRUONG, Dung; NGUYEN, Lam; NGUYEN, Hoa; TRAN, Hai Anh; TRAN, Duc. Detecting Attacks on Web Applications using Autoencoder. In: The Ninth International Symposium on Information and Communication Technology (SoICT 2018), Danang City, Viet Nam, 6 dez. 2018. Disponível em: https://admin.tvrijakartanews.com/uploads/Detecting_Attackson_Web_Applicationsusing_Autoencoder_dab6406334.pdf. Acesso em: 14 ago. 2025.

VARTOUNI, Ali Moradi; KASHI, Saeed Sedighian; TESHNEHLAB, Mohammad. An Anomaly Detection Method to Detect Web Attacks Using Stacked Auto-Encoder. In: 6th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS), 2018. Disponível em: https://netman.aiops.org/~peidan/ANM2022/13.Security/ReadingLists/2018CFIS_SAE.pdf. Acesso em: 15 ago. 2025.