Comparative Analysis Report on Broken Access Control (BAC) Detection Using Neural Network-Based Machine Learning Model

Introduction

This report provides a comparative analysis of the detection approach for *Broken Access Control (BAC)* vulnerabilities using a *Random Forest* model with existing research on *machine learning (ML)* applications in access control. The focus of this analysis is on various *BAC vulnerabilities*, including **Vertical Access Control (VAC)**, **Horizontal Access Control (HAC)**, **Missing Function-Level Access Control (MFAC)**, and **Insecure Direct Object Reference (IDOR)**, which are detected through a structured approach involving a CSV input with specified fields. This comparative analysis highlights the advantages, limitations, and novel aspects of the proposed method compared to other studies in the field.

1. Detection of Multiple Vulnerability Types

The model developed for *BAC detection* incorporates multiple vulnerability types, including **VAC, HAC, MFAC,** and **IDOR**, with individual classifications for each type. This comprehensive approach differs from many existing studies, which generally focus on one or two vulnerabilities. For instance, research conducted by *Barabanov et al.* titled "**Automatic detection of access control vulnerabilities via API specification processing**" concentrates on detecting **IDOR/BOLA vulnerabilities** through API specification analysis. Their approach, while effective for endpoint-specific vulnerabilities, is limited in scope as it does not encompass other types of *BAC vulnerabilities* (Barabanov et al., 2022).

By contrast, this project's model leverages a broader classification system, providing a more holistic view of *access control violations*. This inclusivity allows for detecting multiple *BAC vulnerabilities* within one framework, thereby enhancing the scope of application across varied access scenarios. The integration of such multiple vulnerability types aligns with emerging needs in *cybersecurity* for adaptable, multi-vulnerability detection systems.

2. Application of the Random Forest Model

A *Random Forest model* has been employed, an approach well-suited to handle high-dimensional data and effectively distinguish between multiple classes of vulnerabilities. In studies such as **"Toward Deep Learning Based Access Control"**, researchers have utilized

different *ML techniques*, including **Decision Trees**, **Recurrent Neural Networks**, and **Restricted Boltzmann Machines**, to handle *access control mining* and violation detection.

The *Random Forest model*, known for its robustness and interpretability, is an advantageous choice as it manages categorical data efficiently and maintains transparency in the decision-making process, which is beneficial for *access control* analysis.

Unlike *deep learning methods*, which are explored in recent studies for increased accuracy, the *Random Forest model* allows the present project to maintain interpretability—a key consideration for practical application where detection reasoning is required. While *deep learning models* such as **Recurrent Neural Networks (RNN)** have been utilized to mine policies from logs, the interpretability trade-off associated with these models was avoided in the current project, favoring the transparency of *decision trees* for vulnerability detection.

3. Severity and Priority Scoring System

An additional feature in the proposed model is its **severity** and **priority** scoring system, which assigns vulnerability classifications based on risk. Existing literature generally focuses on the binary classification of vulnerabilities without a multi-tiered risk assessment. The report titled **"Machine Learning in Access Control: A Taxonomy and Survey"** emphasizes that a model with the ability to assess vulnerabilities by severity or risk could aid in triaging security responses. However, the implementation of such a scoring system is rarely observed in academic prototypes.

The *priority* (ranging from 2 to 10 for vulnerabilities) and *severity* (1 to 10) scores in this project are an innovative addition that enables a more nuanced classification of threats. This scoring system is expected to facilitate a more structured approach in vulnerability management, aligning with operational needs for categorizing and prioritizing security incidents based on risk.

4. Feature Engineering and Input Structure

The model employs a *feature set* designed to capture nuanced details related to *BAC vulnerabilities*, such as `client_ip`, `user_role`, `requested_resource`, and `is_id_match`. These features are specifically engineered to capture critical *access patterns* that can reveal *BAC violations*. In comparison, the study **"Automatic Detection of Access Control Vulnerabilities via API Specification Processing"** emphasizes the use of API specifications and endpoint attributes to detect vulnerabilities, illustrating a similar emphasis on *feature engineering* for detecting nuanced vulnerability patterns.

The use of fields like `user_id`, `session_token`, and `is_manipulated` in the proposed model contributes to accurately identifying access control anomalies, distinguishing between legitimate and unauthorized access. These features are not always included in standard *policy mining* methods, suggesting that the current project's data structure captures the complexity of real-world *access control environments* more effectively.

5. Real-Time Detection Capability

This model was designed with *real-time detection capabilities*, an area often underexplored in academic research. Real-time monitoring is critical for practical applications in *cybersecurity*, where immediate identification of *BAC violations* is necessary to prevent exploitation. Research such as **"Toward Deep Learning Based Access Control"** explores *policy mining* but often lacks implementation-ready solutions for real-time applications.

This real-time monitoring feature in the current model positions it for practical deployment in *security systems*, addressing a gap in the transition from theoretical detection models to real-world applicability. The existing literature supports the integration of *OpenAPI specifications* for endpoint detection, which could serve as an enhancement to the current model for API-specific vulnerabilities as well.

Conclusion

The *Random Forest-based approach* developed in this project presents a comprehensive solution for *BAC detection* by incorporating a multi-vulnerability classification system, severity and priority scoring, advanced feature engineering, and real-time detection capabilities. While studies in *access control vulnerability detection* provide essential groundwork, this project's unique features demonstrate an innovative enhancement over conventional approaches. Areas such as vulnerability scoring and multi-vulnerability classification mark this model as a significant advancement, aligning with real-world requirements for dynamic, high-risk *access control environments*.

References:

1. Barabanov, A., et al. "Automatic Detection of Access Control Vulnerabilities via API Specification Processing." arXiv, 2022
2. Alohaly, M., et al. "Machine Learning in Access Control: A Taxonomy and Survey." arXiv, 2022
3. Various Authors. "Toward Deep Learning Based Access Control." arXiv, 2023

Comparative Analysis Report on SQL Injection Detection Using Neural Network-Based Machine Learning Model

Introduction

This report provides a comparative analysis of an SQL injection (SQLi) detection model trained with neural networks, focusing on its feature engineering, loss function optimization, cross-validation, and user interface (UI) capabilities. The model receives SQL query files as input and processes them to classify vulnerabilities, suggesting mitigation strategies and generating visual reports. By comparing it with existing models and research, the report highlights the model's strengths, potential improvements, and unique features.

1. Feature Engineering and Model Training

In the current model, feature engineering is used to enhance detection accuracy, with features extracted from SQL queries based on attributes relevant to SQLi detection. Neural networks are known to be effective for complex patterns in input data, as evidenced by recent studies using TextCNN and LSTM for SQLi detectionMDPI

. The model's training process, which incorporates loss function optimization and cross-validation, aims to increase reliability and prevent overfitting.

Comparison:

- In **"Detection of SQL Injection Attack Using Machine Learning Techniques"**, feature engineering was similarly emphasized for high detection accuracy, with support from algorithms like SVM and Decision Trees, although neural networks were less frequently utilized due to their complexity in handling SQL query data.
- The model's cross-validation strategy strengthens its accuracy by testing across subsets of data, an approach also seen in adaptive models that use neural networks and ensemble methods for enhanced resilience to varied attack types (ResearchGate). However, integrating cross-validation with neural networks in SQLi detection is less common, setting this model apart in terms of robust training.

2. Loss Function Optimization and Performance

The model's loss function optimization is a key step in reducing false positives and enhancing accuracy. Optimized loss functions are critical in SQLi detection to handle the dynamic nature of SQL queries and prevent overfitting, particularly in neural network models. Research, such as that by Krishnan et al., also highlights how specific loss functions and backpropagation help improve detection of complex SQLi patterns by minimizing classification errors (MDPI).

Comparison:

- Traditional models often rely on predefined rules or simpler ML algorithms like Decision Trees, which do not always require intensive loss function optimization due to their inherent interpretability. However, neural networks, such as those utilizing LSTM layers for sequence modeling, benefit significantly from optimization. This aligns with methods used in other advanced SQLi detection models, but differs by leveraging a deep learning-based approach that allows for fine-tuned learning of SQL query patterns and subtle anomalies.

3. User Interface (UI) and Usability

The model integrates a user interface developed in HTML that enables users to upload `.sql` files, which are then classified by the trained neural network model. This user-friendly interface simplifies the detection process and makes SQLi analysis accessible without extensive technical input, an approach often lacking in traditional detection tools.

Comparison:

- In contrast to many existing SQLi detection systems, which may only offer backend solutions without a dedicated UI, this model provides a front-facing UI that accepts query files and returns a visualized report. Research such as "Detection of SQL Injection Attack Using Machine Learning Techniques" focuses more on backend implementations without integrated user interfaces. The model's approach improves accessibility, aligning with trends in cybersecurity towards making tools user-friendly for a broader audience.

4. Classification and Visualization of Vulnerabilities

After classification, the model tests each query for vulnerability type and provides a report that includes both visualization of vulnerabilities (vulnerable vs. safe queries) and a CSV report with mitigation suggestions. This output facilitates a deeper understanding of vulnerability distribution and offers actionable insights.

Comparison:

- Models like the deep forest method applied by Alghawazi et al. demonstrate high accuracy but typically lack visualization features, focusing instead on raw detection output
  MDPI
  . By including visualizations and mitigation strategies, the current model aligns with real-world application needs, providing clear guidance for end-users on query safety and actionable steps.

5. Mitigation Suggestions and Reporting

The model provides automated mitigation suggestions alongside classification results, which directly addresses SQLi risks by suggesting preventative measures. This is a relatively novel aspect, as many models stop at vulnerability detection and do not extend to suggest mitigations. This feature is aligned with models that incorporate defensive tactics for practical application.

Comparison:

- In traditional detection models, SQLi prevention is often left to manual interpretation by cybersecurity teams, but recent AI-driven models, such as those in adaptive deep learning, are beginning to incorporate automated suggestions for improving security practices
  ResearchGate
  . This model's emphasis on suggesting mitigations enhances its practical value, contributing to proactive security efforts against SQLi vulnerabilities.

Conclusion

The neural network-based SQL injection detection model offers an innovative solution by combining advanced feature engineering, loss function optimization, cross-validation, and user accessibility through an HTML interface. Its comprehensive classification, vulnerability visualization, and mitigation suggestions set it apart from existing detection models, providing a more thorough and accessible approach to SQLi prevention.

References:

1. Deep Learning-Based Detection Technology for SQL Injection. MDPI, 2021
2. Detection of SQL Injection Attack Using Machine Learning Techniques: A Systematic Literature Review. MDPI, 2022.
3. Analysis of AI-based SQL Injection Detection Models. ResearchGate, 2022.