Internship Report

Implementation of Web Application Firewall using Machine Learning

Prateek Shrivastava

220031101611032

B.Tech CSE with Specialization in Cyber-Security

Rashtriya Raksha University, Gandhinagar

Table of Contents

Acknowledgments

I would like to extend my deepest gratitude to the Central Research Laboratory, GAD - Bharat Electronics Limited, for affording me the great opportunity to complete this internship on "Implementation of Web Application Firewall using Machine Learning," which took place from 1st August 2024 to 30th September 2024.

I am indeed very grateful to my internship mentor, Mr. Gaurav Damri, for his unwavering support, encouragement, and guidance throughout these two months of the project. His expertise and valuable feedback played a pivotal role in shaping this work.

I would also like to express my heartfelt appreciation to all the team members and staff at the Central Research Laboratory who supported me through this project, fostering a cooperative and enriching learning environment that greatly enhanced my technical skills and deepened my understanding of the subject.

Lastly, I would like to thank my family and friends for their constant support and encouragement throughout my journey as an intern.

Prateek Shrivastava
220031101611032

Abstract

This project concerns the design and implementation of a WAF by means of machine learning techniques to enhance the detection of web-based attacks. Traditional rule-based WAFs are faltering with increased sophistication due to threats such as SQL injection, cross-site scripting (XSS), and other vulnerabilities because it is quite challenging for them to adapt dynamically with new emerging patterns that become part of the attacks. The purpose of this project was to build a machine learning WAF that could detect and mitigate known and unknown real-time threats.

Many machine learning algorithms were used in this internship. They include Random Forest, Support Vector Machines (SVM), and Decision Trees. The main goal is to classify the web requests as malicious or legitimate. The chosen dataset is a labeled HTTP request log from which critical features are extracted and engineered to increase model accuracy. The project involves pre-processing, training, testing, and testing all performance metrics: accuracy, precision, recall, and F1-score.

The results show that the machine-learning-based models provide better performance: almost no false positives, hence the improved efficiency of the WAF in general. The solution is adaptive and allows scalability to safeguard web applications from increasingly diverse sets of cyber threats.

Introduction

Web Application Attacks

Web applications are essential for modern businesses, enabling communication, transactions, and access to critical services. However, this reliance increases the attack surface, making web applications vulnerable to threats that exploit weaknesses in their design, databases, and user inputs.

Some of the major web application security risks identified in the OWASP Top 10 list are as follows:

1. **Broken Access Control:** Where attackers bypass access restrictions to gain unauthorized access.
2. **Cryptographic Failures:** Which expose sensitive data due to weak encryption.
3. **Injection Attacks:** Like SQL injection, where malicious code is inserted into inputs to manipulate databases.
4. **Insecure Design:** stemming from inadequate security measures during development.
5. **Security Misconfiguration:** Where improper settings leave applications vulnerable.
6. **Vulnerable and Outdated Components:** Introduce risks due to unpatched software.
7. **Identification and Authentication Failures:** Allow attackers to impersonate users.
8. **Software and Data Integrity Failures:** Expose applications to attacks via insecure updates.
9. **Logging and Monitoring Failures:** Delay attack detection.
10. **Server-Side Request Forgery (SSRF):** Tricks the server into making unauthorized requests, potentially exposing internal systems.

To mitigate these risks, Web Application Firewalls (WAFs) are crucial. They filter and monitor web traffic, ensuring that only legitimate requests reach the server. WAFs configured with the OWASP Core Rule Set (CRS) effectively prevent many of these vulnerabilities through predefined rules and anomaly detection, enhancing overall web security.

The Role of OWASP Core Rule Set (CRS)

Given the widespread vulnerabilities of the web, OWASP developed a set known as the OWASP ModSecurity Core Rule Set (CRS). CRS is a set of open-source rules designed to detect and prevent common web application attacks, providing a baseline defense for WAFs. It protects

against OWASP's Top 10 security risks, such as SQL injection, XSS, and CSRF, by examining incoming web requests for known attack patterns. Key features of the CRS include attack detection using pattern matching and anomaly scoring, targeted prevention of OWASP Top 10 vulnerabilities, and an anomaly scoring system that prioritizes threats. The CRS is customizable to meet specific organizational needs, reducing false positives and enhancing detection accuracy.

Enhancing WAFs: Integrating Machine Learning with OWASP Core Rule Set

The OWASP Core Rule Set (CRS) is installed on a Web Application Firewall (WAF) and provides the first layer of defense against common web application attacks. While it uses signature-based detection techniques and anomaly scoring to identify known vulnerabilities and suspicious behavior, integrating machine learning alongside OWASP CRS can significantly enhance attack detection, especially for zero-day attacks. Machine learning models can analyze large volumes of traffic data, learn normal patterns, and detect deviations in real-time, identifying potential threats that traditional signature-based systems may miss. By continuously learning from new traffic and attack patterns, machine learning enables WAFs to detect novel or evolving attack vectors without relying solely on predefined signatures, thus providing a more adaptable, proactive defense. This hybrid approach leverages the strengths of both signature-based detection and behavioral analysis, offering robust protection against both known and unknown threats.

Project Methodology and Implementation

Dataset

The classification experiments were conducted on the CSIC 2010 dataset, a widely used benchmark for testing web application firewalls and intrusion detection systems. Created by the Spanish Research National Council, the dataset contains over 36,000 labeled HTTP requests—both normal and malicious—mimicking realistic web traffic for an e-commerce application. It includes various attack types, such as SQL injection, buffer overflow, and information leakage, making it ideal for training classifiers to differentiate between normal and anomalous requests. Its diversity and structure provide a strong foundation for evaluating security models.

Request Parser for Web Traffic Analysis

In this project, I developed a custom request parser to extract and analyze key features from web traffic logs, specifically tailored for HTTP requests. The parser processes raw log files and converts them into a structured format suitable for machine learning classification. It works by reading line-by-line from the input file, identifying valid HTTP requests, and extracting important components such as the HTTP method, full URL, URL path, and query parameters.

For each request, the parser calculates additional features, including the number of query parameters, content length, body length, and counts of special characters. Special characters are often associated with web application attacks, so identifying their occurrence helps in detecting potentially malicious requests. The parser also captures various request headers such as User-Agent, Content-Type, Host, and others, allowing for a comprehensive analysis of each request.

The extracted data is then written to a CSV file, with one row representing each request, containing over 20 features, such as the HTTP method, query parameters, and special character counts. This structured format enables efficient training and testing of machine learning classifiers to distinguish between normal and anomalous requests. The parser is designed to handle large datasets and ensures the extraction of relevant features critical for Web Application Firewall (WAF) development and testing.

Data Classifier for Anomaly Detection

Initial Random Forest Classifier

I implemented a Random Forest classifier to train and test on the parsed HTTP request data, with the goal of distinguishing between normal and anomalous requests. The classifier is based on the Random Forest algorithm, a powerful ensemble learning method that constructs multiple decision trees and aggregates their predictions for improved accuracy and robustness.

The process begins by loading the pre-processed dataset, which contains a variety of features extracted from HTTP requests, including headers, body parameters, and special character counts. The target variable for classification, labeled as request_type, indicates whether a request is valid or malicious.

To ensure the model handles categorical data effectively, One-Hot Encoding is applied to any categorical features. This transforms these columns into a binary format, which is necessary for the Random Forest classifier to work efficiently. The dataset is then split into training and testing sets using an 80-20 split, ensuring a proper evaluation of the model's performance.

A Random Forest classifier is initialized with 100 decision trees to balance accuracy and computational efficiency. The model is trained on the training set, after which predictions are made on the testing set. To evaluate its performance, key metrics such as accuracy and a classification report (which includes precision, recall, and F1-score) are generated.

This model serves as a critical component in identifying suspicious or malicious web traffic, supporting the goal of enhancing web application security through the development of an effective Web Application Firewall (WAF). The results from this model will be analyzed to fine-tune its parameters for better performance in future iterations.

Result

```
Accuracy: 0.8956030459346598
              precision    recall  f1-score   support

           0       0.86      0.90      0.88      5011
           1       0.93      0.90      0.91      7202

    accuracy                           0.90     12213
   macro avg       0.89      0.90      0.89     12213
weighted avg       0.90      0.90      0.90     12213
```

The Random Forest classifier was trained and tested on the parsed HTTP request data to differentiate between normal and malicious requests. After splitting the data into training and testing sets, the model was able to achieve an accuracy of 89.5%, indicating its ability to correctly classify requests most of the time.

Iterative Fine Tuning with Misclassification Handling

To improve the performance of the previous Random Forest model, several refinements were made. In this fine-tuned approach, categorical features were one-hot encoded to better represent the dataset's structure. The dataset was split into training and testing sets with a 70/30 split to allow for more comprehensive testing.

Additionally, an iterative approach was implemented to improve model performance by identifying and incorporating misclassified samples back into the training set over several iterations. This refinement allowed the model to learn from its mistakes, progressively enhancing accuracy and reducing misclassifications.

The results of this fine-tuned model showed notable improvements over the previous one, reflected in a higher accuracy score and better overall classification metrics, demonstrating the effectiveness of these adjustments in enhancing the model's detection of anomalous and valid requests.

Result

```
Iteration 1:
[[6778  752]
 [1104 9686]]
              precision    recall  f1-score   support

           0       0.86      0.90      0.88      7530
           1       0.93      0.90      0.91     10790

    accuracy                           0.90     18320
   macro avg       0.89      0.90      0.90     18320
weighted avg       0.90      0.90      0.90     18320

Accuracy: 0.8987
```

```
Iteration 2:
[[ 6556   974]
 [  626 10164]]
              precision    recall  f1-score   support

           0       0.91      0.87      0.89      7530
           1       0.91      0.94      0.93     10790

    accuracy                           0.91     18320
   macro avg       0.91      0.91      0.91     18320
weighted avg       0.91      0.91      0.91     18320

Accuracy: 0.9127
```

```
Iteration 3:
[[ 6957   573]
 [  575 10215]]
              precision    recall  f1-score   support

           0       0.92      0.92      0.92      7530
           1       0.95      0.95      0.95     10790

    accuracy                           0.94     18320
   macro avg       0.94      0.94      0.94     18320
weighted avg       0.94      0.94      0.94     18320

Accuracy: 0.9373
```

```
Iteration 4:
[[ 6965   565]
 [  445 10345]]
              precision    recall  f1-score   support

           0       0.94      0.92      0.93      7530
           1       0.95      0.96      0.95     10790

    accuracy                           0.94     18320
   macro avg       0.94      0.94      0.94     18320
weighted avg       0.94      0.94      0.94     18320

Accuracy: 0.9449
```

```
Iteration 5:
[[ 7017   513]
 [  530 10260]]
              precision    recall  f1-score   support

           0       0.93      0.93      0.93      7530
           1       0.95      0.95      0.95     10790

    accuracy                           0.94     18320
   macro avg       0.94      0.94      0.94     18320
weighted avg       0.94      0.94      0.94     18320

Accuracy: 0.9431

Training complete.
```

The fine-tuned classifier demonstrated significant improvements in accuracy and performance over multiple iterations. Starting with an initial accuracy of 89.87% in Iteration 1, the model progressively improved, achieving a final accuracy of 94.31% by Iteration 5. The iterative refinement process, where misclassified samples were added back to the training set, helped the model improve its understanding of complex patterns in the dataset.

Final Model with Enhanced Generalization

The final model implemented for this project builds upon the previous Random Forest Classifier by incorporating more advanced techniques such as GridSearchCV for hyperparameter tuning and iterative training with misclassified samples. These refinements have significantly improved the classifier's performance.

The previous model, although functional, did not include advanced techniques like GridSearchCV for tuning and lacked dynamic feature handling like class weights. While it also iterated over misclassified samples, the performance improvements were more limited compared to the fine-tuned model.

Overall, the fine-tuned Random Forest classifier represents a more robust, precise, and efficient approach to classifying requests, with an enhanced focus on optimizing model performance and mitigating class imbalances.

Result

```
Iteration 1:
[[655 111]
 [ 81 985]]
              precision    recall  f1-score   support

           0       0.89      0.86      0.87       766
           1       0.90      0.92      0.91      1066

    accuracy                           0.90      1832
   macro avg       0.89      0.89      0.89      1832
weighted avg       0.90      0.90      0.89      1832

Accuracy: 0.8952
```

```
Iteration 2:
[[ 705   61]
 [  57 1009]]
              precision    recall  f1-score   support

           0       0.93      0.92      0.92       766
           1       0.94      0.95      0.94      1066

    accuracy                           0.94      1832
   macro avg       0.93      0.93      0.93      1832
weighted avg       0.94      0.94      0.94      1832

Accuracy: 0.9356
```

```
Iteration 3:
[[ 713   53]
 [  41 1025]]
              precision    recall  f1-score   support

           0       0.95      0.93      0.94       766
           1       0.95      0.96      0.96      1066

    accuracy                           0.95      1832
   macro avg       0.95      0.95      0.95      1832
weighted avg       0.95      0.95      0.95      1832

Accuracy: 0.9487
```

```
Iteration 4:
[[ 732   34]
 [  27 1039]]
              precision    recall  f1-score   support

           0       0.96      0.96      0.96       766
           1       0.97      0.97      0.97      1066

    accuracy                           0.97      1832
   macro avg       0.97      0.97      0.97      1832
weighted avg       0.97      0.97      0.97      1832

Accuracy: 0.9667
```

```
Iteration 5:
[[ 737   29]
 [  29 1037]]
              precision    recall  f1-score   support

           0       0.96      0.96      0.96       766
           1       0.97      0.97      0.97      1066

    accuracy                           0.97      1832
   macro avg       0.97      0.97      0.97      1832
weighted avg       0.97      0.97      0.97      1832

Accuracy: 0.9683
```

The final model of Random Forest classifier demonstrated significant improvements in performance compared to the earlier version. In Iteration 1, the model achieved an accuracy of 87.06%, with a recall of 0.95 for class 0 and 0.82 for class 1.
By Iteration 5, the accuracy had increased to 96.83%, with both classes showing significantly high precision (class 1 at 0.98) and recall (class 1 at 0.97). This demonstrates the classifier's improved ability to correctly predict both normal and anomalous requests.

Conclusion

In this internship project, the focus was on developing a robust web application firewall using machine and deep learning techniques to effectively classify web requests as normal or anomalous. The project involved utilizing the CSIC 2010 dataset, a well-established benchmark for evaluating intrusion detection systems, which provided a comprehensive set of labeled HTTP requests.

The initial phase of the project involved the development of a data parser, which successfully extracted essential features from raw log data and transformed them into a structured format suitable for analysis. This preprocessing step laid the groundwork for building a machine learning classifier.

Subsequently, a Random Forest classifier was employed to train and evaluate the model on the parsed dataset. Initial results demonstrated promising accuracy; however, further fine-tuning through an iterative approach led to significant performance enhancements. By incorporating misclassified samples back into the training set and by employing hyperparameter tuning using GridSearchCV, the model progressively refined its ability to distinguish between normal and anomalous requests, culminating in a final accuracy of 96.83%.

The results highlight the effectiveness of the implemented approach, showcasing how iterative training can significantly improve classifier performance. Overall, this project successfully demonstrated the potential of machine learning techniques in enhancing web application security, paving the way for future advancements in the field of cybersecurity. The skills and knowledge acquired during this internship will undoubtedly contribute to ongoing efforts in developing secure web applications and robust intrusion detection systems.

References

Web application firewall using character-level convolutional neural network. (2018, March 1). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/abstract/document/8368694

Web application attacks detection using machine learning techniques. (2018, December 1). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/abstract/document/8614199

Dawadi, B., Adhikari, B., & Srivastava, D. (2023). Deep Learning Technique-Enabled Web Application firewall for the detection of web attacks. *Sensors*, *23*(4), 2073. https://doi.org/10.3390/s23042073

OWASP CRS | OWASP Foundation. (n.d.). https://owasp.org/www-project-modsecurity-core-rule-set/

OWASP Top Ten | OWASP Foundation. (n.d.). https://owasp.org/www-project-top-ten/

RandomForestClassifier. (n.d.). Scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html