Machine Learning for Detection of Distributed Denial-of-Service Attacks from Queries Executed in DBMS

Danilo A. M. Chagas[1], Geraldo P. Rocha Filho[2], Rodolfo I. Meneguette[3]
Rodrigo Bonacin[4], Vinícius P. Gonçalves[1]

[1]Universidade de Brasília (UnB) – Brasília – DF – Brazil

[2]Universidade Estadual do Sudoeste da Bahia (UESB) – Vitória da Conquista – BA – Brazil

[3]Universidade de São Paulo (USP) – São Carlos – SP – Brazil

[4]CTI Renato Archer and Unifaccamp – Campinas – SP – Brazil

danilo.chagas@aluno.unb.br, geraldo.rocha@uesb.edu.br, meneguette@icmc.usp.br

rodrigo.bonacin@cti.gov.br, vpgvinicius@unb.br

Abstract. *Denial-of-Service (DoS) attacks have been extensively studied in the literature, especially in their most dangerous form, the Distributed Denial-of-Service (DDoS). Database, a critical infrastructure for services, has mechanisms for recording information (logs) of SQL queries and sessions. Although they are vulnerable to DDoS, they are not entirely covered by commercial tools or research on such a detection. Machine Learning (ML) techniques are highly effective in identifying patterns in data such as database SQL logs. Thus, this work proposes the application of ML to detect DDoS attacks on a database from the logs of queries executed on it. As a result, the classification obtained an F1-score of 94.44%, which indicates the effectiveness of the proposed approach.*

1. Introduction

Denial-of-Service (DoS) attacks intend to exhaust the resources of the infrastructure of a computational service, making it unavailable to legitimate users [Brooks et al. 2021]. A consummated DoS attack violates the availability pillar of the organization's Information Security [Haider et al. 2020]. The attack on GitHub in 2018 illustrates a massive DoS in its distributed form, Distributed Denial-of-Service (DDoS), demonstrating its potential damage. This attack had a 1.35 Tbps of volume, making it the most significant DDoS attack ever seen then [Chadd 2018, Haider et al. 2020].

DoS attacks can be targeted at protocols from lower layers of the Open Systems Interconnect (OSI) and TCP/IP models, such as network and transport layers, up to the highest (application) layer. Attacks on the network and transport layers - layers 3 and 4 of both models - are more common and, due to this, most frequently investigated. Considering application layer attacks, the most frequent ones – and equally studied – are variants of HTTP protocol attacks [Tripathi and Hubballi 2021, Vedula et al. 2021].

The most common attacks on Database Management Systems (DBMSs) are SQL Injection (SQLIA – SQL Injection Attack) [Hashem et al. 2021]. SQLIA is intended for unauthorized access to the database, recovery of sensitive data, or the unavailability of such services [Alwan and Younis 2017, Varshney and Ujjwal 2019, Medeiros et al. 2019, Aliero et al. 2020].

Machine Learning (ML) is a field of artificial intelligence that allows computers to learn from data without being explicitly programmed to do so. ML has been used for solving problems that cannot be resolved easily by traditional approaches, or algorithms still need to be developed to solve them [Géron 2017]. As non-exhaustive examples of its use, we have speech and image recognition, classification of unwanted messages (spam), product recommendation systems, and pricing assets such as real estate or stocks. Another area that hugely benefits from ML is detecting and preventing cyber-attacks, in which both DoS and DDoS are found [Gormez et al. 2020, Kaur et al. 2019, Berman et al. 2019].

ML can rely on data collected from various parties of an infrastructure [Souza et al. 2021]. The models are first trained on this collected data [Cavalcante et al. 2022]. Based on this trained model, it is possible to identify patterns (or mathematical relations) representing information. For example, DDoS attacks tend to have the same (or very close) pattern depending on the data analyzed. However, the efficiency of an ML algorithm relies heavily on how well it can represent the input data. Low-quality training data leads to imperfect representations and, consequently, lower performance of ML techniques [Pouyanfar et al. 2018].

The literature has given greater focus to attacks related to communication protocols, using analysis of packets captured in a network [Akgun et al. 2022, Aliero et al. 2020, Sofi et al. 2017, Mittal et al. 2022]. However, attacks directed at DBMS may not be detected through such examinations since queries are forwarded to it encapsulated in legitimately formed packets. This way, apparently legitimate queries – which present normal behavior in communication protocols – can be executed solely to exhaust the DBMS resources [Gurina and Eliseev 2019, Lima Filho et al. 2019].

Relational DBMSs typically record every session opened and every query performed in log tables. These tables have valuable data about the number of query executions, the number of records returned, memory consumption, usage time, processor waiting time, and the sessions opened [Togatorop et al. 2022]. Their information includes the SQL command executed, query performance parameters, the user who ran it, and the session's state. Thus, the DBMS records quantitative parameters regardless of their nature, whether they are legitimate queries and sessions or with malicious purposes – albeit, in this case, having a coherent syntax.

Therefore, this work aims to use ML to detect DDoS attacks on databases. Our approach uses DBMS log tables with all session data for training supervised ML algorithms, which are used to classify them in terms of their legitimacy regarding service availability. The queries themselves are not analyzed in this work – there is no need for that since all the necessary information can be extracted from the sessions logged.

It should be noted that this classification uses information about session states generated whenever a SQL query is executed in the DBMS. In this work, we carry out a study with data from legitimate sessions and attacks on an Oracle DBMS of a Brazilian public institution of the judiciary system. Our work focuses on determining if it is possible to detect Denial-of-Service attacks (including DDoS) on DBMS from the data collected from the DBMS internal logs. We add that it is outside the scope of this work to deal with their prevention. Three ML techniques were investigated. XGBoost presented the best results, with accurate detection of DDoS attacks directed to databases, without False

Positives and with a low rate of False Negatives, reaching an F1-score of 94.44% for our test dataset.

The remainder of this paper is organized as follows: Section 2 presents recent works on detecting database attacks; Section 3 presents our approach; Sections 4 and 5 present the results and discussion, respectively; Section 6 wraps up the findings and points out further research.

2. Related Work

DoS is a harmful threat to any organization, particularly in its distributed variant (DDoS). Many studies on its detection have been conducted, most addressing services communications issues.

Alkasassbeh et al. (2016) used Multilayer Perceptron (MLP), Random Forest, and Naïve Bayes ML techniques to detect DDoS in the application and network layers. They generated a dataset in Network Simulator 2 (NS2) since they believed the available datasets would not allow realistic and practical results in detecting DDoS. This dataset included HTTP flood and SQL Injection DDoS (SIDDoS), among other attacks from the network layer. The results show high accuracy for all investigated attacks in their study. However, they also show lower precision for SIDDoS.

Lima Filho et al. (2019) analyzed sneaky, low-volume DDoS in the application layer. The authors state that security teams often do not even know these attacks are happening because standard tools cannot detect them. They emphasize that these attacks consume less bandwidth, exploit application layer protocols, respect other lower-level protocols, and exhaust the victim's resources. They also explained that ML offers high flexibility in the classification process, which improves malicious traffic detection rates. The paper presents a tool called Smart Detection. They used available datasets to detect volumetric attacks on transport and application layers and stealth attacks on the HTTP protocol (application layer). The Random Forest algorithm presented the best results for their dataset.

Hashem et al. (2021) propose a mechanism to simultaneously detect DoS and SQLIA. It uses a publicly available dataset called NSL-KDD that contains DDoS attack records and is based on packet capture. The proposed system has two parts: the first is responsible for detecting DDoS in the network, transport, and application layers, and the second is responsible for detecting SQLIA. Their work uses tokenization to train an ML model for query patterns to detect SQLIA. The system then checks every query executed and classifies them into a suspicious pattern or not.

Sofi et al. (2017) analyze modern DDoS attacks, such as HTTP flood and SIDDoS. Their work explains that detecting DDoS is difficult since illegitimate packets can be indistinguishable from legitimate ones. They resorted to a new dataset containing modern DDoS types to carry out the work since they found out that there is no dataset available that includes the studied attacks. It is worth noting that this new dataset was collected from network traffic (packet capture). The algorithms Naïve Bayes, Random Forest, and MLP were used, with the latter presenting the best overall performance.

Medeiros et al. (2019) noticed that the DBMS is an interesting place to add protection against DDoS because DBMS has unequivocal knowledge about clauses, predicates,

Table 1. Related works characteristics

Reference	DBMS DDoS Detection	Analysis at the DBMS Level	Maintains DBMS Architecture
[Alkasassbeh et al. 2016]	✓	✗	✓
[Hashem et al. 2021]	✓	✗	✓
[Sofi et al. 2017]	✓	✗	✓
[Medeiros et al. 2019]	✓	✓	✗
Our work	✓	✓	✓

and expressions in a SQL query. The work highlights that no mechanism outside the DBMS would have such an awareness. They proposed a tool called SEPTIC, which deals with two categories of attacks on databases: SQL Injection and storage injection. It was implemented by a module inside the DBMS that checks every query executed for attacks. The tool has the disadvantage of altering the DBMS architecture. However, it presents low rates of false negatives and false positives.

As presented above, we identified that most works rely on analyzing and detecting DDoS from packet captures. Some works make assumptions about HTTP packets, functioning similarly to a Web Application Firewall (WAF). Only one work dealt directly with catching such an attack from inside the DBMS. Hence, this points out the need to explore the matter further.

Thus, to the best of our knowledge, this is a novel study because it uses a different approach to the problem of identifying DDoS in databases. Instead of analyzing such an attack using packet captures, we propose to verify the logs of the queries executed in a database. This traffic was deemed legitimate by the network-level controls, thus bypassing them, and making their way to the intended destination but causing a Denial-of-Service to the DBMS. Table 1 presents key characteristics of the related works as well as this work.

3. Proposed Approach and Study

Firstly, this section presents the adopted ML methodology used in our approach to detecting DDoS attacks on a DBMS (subsection 3.1), which implements an analysis of the sessions created in the DBMS whenever a query is executed. Subsections 3.2 to 3.7 present the application of our approach in a study with an Oracle DBMS.

3.1. Overview of the Approach

The anatomy of a DoS/DDoS attack on a DBMS is illustrated in Figure 1. An organization can provide its service via publicly available software accessed by HTTP/HTTPS. Users utilize the interface provided and send HTTP/HTTPS requests to the software's backend, that then forms queries and sends them to its database. The DBMS executes the action, registers the query and its parameters in its internal logs, and responds with the data needed to the backend. Under normal usage (identified by the number "1"), regular users (presented by the letter "A") make an access and receive the desired response. In this situation, the whole infrastructure exhibits acceptable loads. However, in an attack scenario (identified by the number "2"), malicious users (letter "B") send a significant

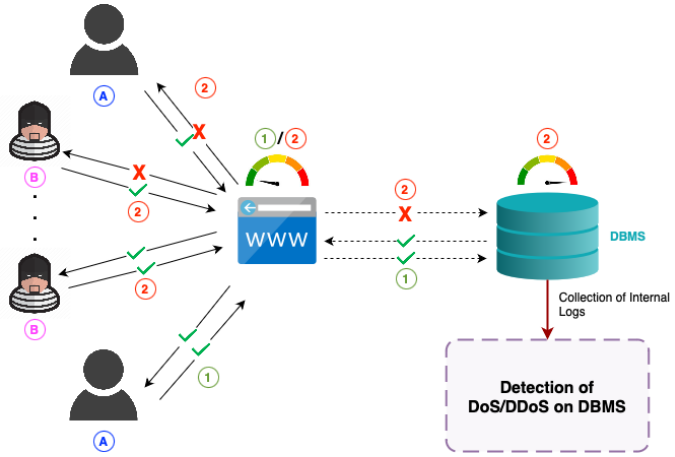

Figure 1. Anatomy of a DoS/DDoS attack on a DBMS

number of requests from one (DoS) or many (DDoS) origins [Brooks et al. 2021]. These requests intend to cause strain on the DBMS infrastructure. It is noticeable, though, that the application server (or HTTP/HTTPS server) is possibly not affected by these requests in terms of load. In this case, such an attack will be a low-rate, non-volumetric attack in which the traffic resembles a legitimate one [Gümüşbaş et al. 2020, Vedula et al. 2021].

As a consequence of the scenario described, most of the processing required by the malicious users is executed at the DBMS. As a result, the DBMS infrastructure becomes saturated and, later, cannot cope with further requests. When such a situation happens, all subsequent users do not receive a response from the DBMS, although the HTTP/HTTPS portion of the software's infrastructure is still responsive.

We chose to analyze the DMBS internal logs because of a few factors. Despite being possible to detect such an attack from other components in the infrastructure, the DBMS logs can be richer in terms of information likely to obtain in such an analysis. They contain features that register the performance of the operations executed, as well as the resource consumption of the server. These parameters store information known solely inside the DBMS since they are the product of the queries' executions [Oracle 2023]. For this reason, these pieces of data are not present in packets like HTTP/HTTPS requests or responses, nor on database requests initiated at the application server. Thus, we understand that our work brings a complementary approach to detect DDoS attacks on DBMSs, bringing an additional security layer to the infrastructure.

This work uses widely disseminated methods in ML and data mining. Yet, they were instantiated and adapted for the context in focus, composing a five stages approach (Figure 2). The first relates to collecting data from the DBMS, which consists of logs of sessions created in the database every time it executes a query. These logs comprise features that identify each session, including the user that runs it, time spent by the processor, user identification, IP address, wait time, and others. These pieces of information represent the performance parameters of queries the DBMS runs, making it a fingerprint of each query.

The second stage is to label the records of the dataset based on an actual DDoS attack to a production DBMS, in which the attack agents were identified using various

tools.

The third stage consists of feature extraction and feature engineering, in which we excluded unnecessary features and extracted new characteristics from information present in the dataset. The codification of categorical variables is performed during this stage, alongside the creation of new features.

The fourth stage is to train the model to detect attacks using a cross-validation technique. Several ML techniques can be evaluated in this stage. We evaluated three techniques in the study presented in this paper: Logistic Regression, Random Forest, and XGBoost. These techniques were chosen for this study because they showed promising results in the literature and presented acceptable computational costs.

The fifth stage is the application of the trained model to new data. This strategy was preferred, in this study, over the training-test split because the training dataset was relatively small. Considering that our work utilized cross-validation with excellent results in the training phase, this approach to new data would be more realistic than splitting a small dataset. Other testing strategies can be used depending on the amount and characteristics of the dataset.

The following subsections, in sequence, present details of how each stage mentioned above was carried out in our study.

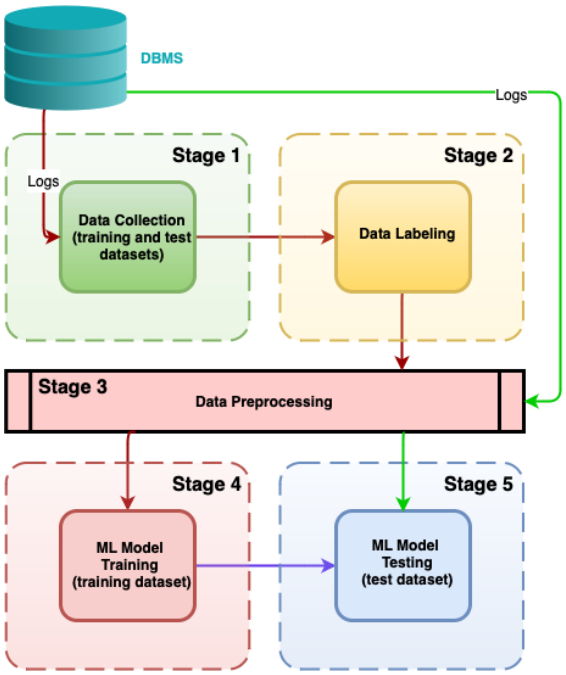

Figure 2. Overview of the proposed stages for detecting DoS/DDoS attacks on a DBMS

3.2. Collecting the Data from an Oracle DBMS

The data used in this project originates from the users' sessions table in an Oracle DBMS. A session in a database represents the state of users' login to a database. It accompanies the user life cycle in the database, from entering in execution until its disconnection from the application. It holds the user's identification (username, IP address, port, logon time)

and some performance markers – such as the state of the connection, time spent waiting or executing, the memory that can be used, concurrency event messages [Oracle 2023].

The initial data was collected at the time of an actual attack on a production database of a Brazilian federal institution of the judiciary system - its name will not be mentioned due to confidentiality agreements. This database holds information for a vastly utilized, in-house developed software in the attacked organization. It is available inside and outside the organization, hence being susceptible to internal and external attacks. Around 100 agents were identified as performing the attack, all coming from different locations (i.e., IP addresses).

The data collection yielded almost 7,000 records generated in a single day. We utilized this first batch of gathered data as the training dataset. For the test dataset, we stored data from the same table on the subsequent day of the start of the attack.

As in a typical system scenario in a real-world situation, only a small fraction of its activity is malicious. Therefore, the data used as the training set obtained was highly imbalanced, with only 0.227% of the records labeled as attack records. In order to bring consistent results to the ML models, an oversampling strategy was used for the training dataset, balancing the row count to a 57/43 ratio – this ratio was chosen discretionarily.

3.3. Labeling DDoS Attack Data

The records in the datasets were labeled based on an actual DDoS attack that happened in May 2022. The malicious activity occurred in a well-known Brazilian federal department, potentially causing damage to its image and its function. An incident response team comprised of developers, application server administrators, database administrators, network administrators, and cybersecurity specialists performed the identification of the attack. This malicious activity was not detected in the TCP/IP protocols' lower layers or cybersecurity appliances, such as IDS/IPS (Intrusion Detection Systems/Intrusion Prevention Systems) or WAF. The suspicion remained because the DBMS server presented poor performance due to high loads spread throughout its components, such as processors and disks.

The investigation took place using tools such as DBMS performance software, DBMS sessions' table/view, and NMS (Network Monitoring System), determining which database users were utilized by the attacker. To do so, the team analyzed each user with a session opened at that moment, narrowing it down to the users performing the attack. The sessions' logs belonging to those users were then manually labeled as a DDoS attack; all other records were marked as non-attack activity. Therefore, it represents a single-label classification problem.

3.4. Feature Selection, Feature Engineering, and Missing Data of DBMS logs

The data obtained comprises 99 features, including 48 literal features, three (3) date and time features, and 48 numeric features. Some literal features include hash or literal values that represent redundant information, making them dispensable. Other literal features included helpful information, such as the IP address, and the name of the user who performed the operations. We then encoded literals and date and time into numeric features to be used by ML techniques. The rest of the literal features were excluded since they are not directly or indirectly linked to possible attack activities. This analysis was based on

consultation of the sessions' table documentation provided by the DMBS vendor referring to the meaning of each feature.

Regarding missing data analysis, we found some features with a missing rate higher than 70% – these are the ones likely to reduce the ML performance [Yu et al. 2022]. In these cases, we discarded the features to avoid this loss in the training metrics. We also analyzed rows with little data – less than 70% of the row populated – and discarded them as a whole [Yu et al. 2022]. This threshold takes care of dropping only a tiny portion of the data, which, in this case, represented less than 0.5% of the row count. At the end of the feature selection process, our dataset required no data imputation.

Based on the documentation, we identified variables with categorical values and performed their codification. Categorical attributes hold nominal values, which makes them qualitative variables or representations of categories. However, the selected ML algorithms do not deal with such features, making it necessary to transform them into numeric attributes - this operation is called codification. The codification chosen was the One-Hot Encoding technique, which transforms each feature category into one new column, giving it the value "0" if it is not present in that particular row and "1" otherwise [Kernbach and Staartjes 2022, Yu et al. 2022]. This codification suits the categories in our dataset since they do not represent any order.

We analyzed the possible values for each category to avoid incurring the so-called curse of dimensionality. This unwanted situation is presented when the number of features (dimensions) grows to high numbers. It makes the model more complex and brings drawbacks, such as high memory consumption and data sparsity, making it challenging to analyze the dataset. One solution, in these cases, could be increasing the training dataset size to reach a sufficient density of training instances. However, this was not feasible in our case since there was no additional data to be collected. Another possible method is to perform dimensionality reduction, which consists in condensing a high-dimensional space into a low-dimensional one [Thudumu et al. 2020]. This transformation retains important properties of the original dimensions, keeping most of the meaning of the data. As described above, we utilized feature selection to keep a low number of features. Thus, it was not necessary to use a dimensionality reduction method.

3.5. Balancing the DBMS logs

Imbalanced data can affect ML algorithms as the models know little about one of the classes. Models trained using imbalanced datasets tend to make a good prediction of the most common class but an inaccurate prediction of the other. While, in some cases, this may result in good predictive metrics, it is generally not desirable in real problems once it is almost useless to predict the classes with fewer instances [Kaur et al. 2019].

This situation must be avoided, mainly because this work intends to emphasize a precise prediction of attacks (that would represent the less numbered class in the dataset). We treated this problem by applying balancing techniques, most notably the oversampling of the smaller class, which consists in synthetically generating new rows of it. The other approach we could have used would be undersampling, i.e., reducing the amount of data of the predominant class to equalize with the smaller one. This choice would lower the number of rows to a small amount, which is inappropriate for training a supervised ML model. For our dataset, we utilized SMOTE (Synthetic Minority Oversampling

Technique) to the oversampling [Kaur et al. 2019].

3.6. Training the Models for Predicting DDoS Attacks

To determine which model was more effective using our data, we performed a 10-fold cross-validation of the three ML techniques. The dataset is divided into ten parts, nine being used as the training dataset and the other as a validation dataset. The purpose of this technique is to avoid overfitting and estimate how well the model will perform with new data. Figure 3 presents the cross-validation process.

One thing we considered before the utilization of cross-validation is how this process should divide our dataset. First, we had to assume that it is a single-label classification problem. In this scenario, random data division could end up with partitions with different statistical distributions for the target class. We applied a stratified split in the cross-validation process to overcome this problem. This ensures that each split has the same statistical distribution regarding the target variable. Thus, we end up with a more precise evaluation of the models [Doan et al. 2022].

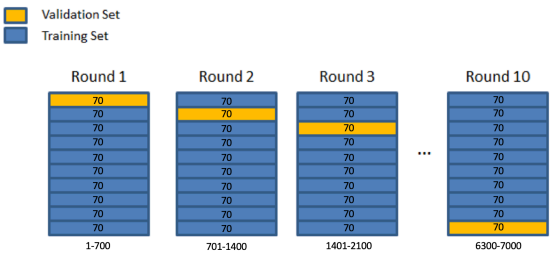

Figure 3. Cross-validation process.

3.7. Testing the Effectiveness of Models in Detecting DDoS Attacks

The most common way to test a model's effectiveness is splitting the dataset into two or three subsets, including training-test or training-validation-test [Kernbach and Staartjes 2022]. However, as aforementioned, we collected the test dataset not in the initial moment, although when the attack was still happening. We elected to test the models on new data instead of splitting the training dataset for a few reasons, as described below.

First, we had to consider that our initial dataset needed to be bigger to split without negatively impacting training. Another considered aspect is that, by collecting a new test dataset, we avoid any possibility of data leakage. Data leakage occurs when the trained model has knowledge of the data present in the test dataset [Kernbach and Staartjes 2022]. When such a situation happens, the trained model will probably not perform as well with production data as with the test dataset.

We also wanted to perceive how practical our proposal was in a real-world situation. However, to use this approach, it is imperative to determine whether the new data has the same statistical distribution as the training dataset. We used Adversarial Validation to this end [Burkov 2020].

Adversarial Validation is a technique applied to investigate if the characteristics of the test dataset are the same as the training dataset. Identifying any changes in the data

distribution that could make the trained model underperform with new data is crucial. This technique starts by excluding the target feature and creating a new one representing whether the data comes from training or test records. Then a classification model makes predictions and the metric ROC-AUC (Receiver Operating Characteristic - Area Under the Curve) is evaluated in this technique. A resultant value close to 50% indicates that both data are indistinguishable and therefore have the same distribution [Burkov 2020]. In our case, we attained a ROC-AUC of 51.31% on an XGBoost model, evidencing that we can employ our new data as a test dataset.

4. Performance Evaluation

Our work uses the libraries Pandas and Numpy to handle the data and the libraries Scikit-Learn and XGBoost to execute the ML techniques. The results were evaluated considering precision, accuracy, recall, ROC-AUC, and F1-score. These measures describe different aspects of the classifier performance as follows.

Precision indicates the accuracy of positive predictions and is obtained by the equation:

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

Accuracy relates to the rate of the correctly classified instances of both classes, and the following equation obtains it:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

The recall, also called true positive rate (TPR), represents the positive instances correctly predicted by the model, and it is obtained by the equation:

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

The F1-score is a common way of comparing classifiers; it combines precision and recall into one metric – it is the harmonic mean of precision and recall:

$$F1\text{-}score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{4}$$

ROC-AUC is another way to compare classifiers, plotting the recall (TPR) against the false positive rate (FPR) [Carrington et al. 2022, Berman et al. 2019, Molina et al. 2022]. The FPR relates to the negative instances classified incorrectly as positive, and it is represented by the equation:

$$FPR = \frac{FP}{FP + TN} \tag{5}$$

In our work, as aforementioned, we initially compared the performance of three classifiers: Logistic Regressor, Random Forest, and XGBoost – we chose the metrics mentioned in this section. The results of the training-validation dataset are presented in Table 2.

Table 2. Performance results for the training-validation dataset

Performance Evaluation - Training Set - % - Average in Cross-Validation			
Metric	Logistic Regression	Random Forest	XGBoost
Accuracy	95.87	**99.97**	99.89
Precision	91.84	**99.92**	99.75
Recall	**100**	**100**	**100**
F1-score	95.6	**99.96**	99.88
ROC-AUC	97.99	**100**	**100**

Table 3. Performance results for the test dataset

Performance Evaluation - Test Set - %			
Metric	Logistic Regression	Random Forest	XGBoost
Accuracy	79.04	87.17	**95.72**
Precision	77.38	**100**	**100**
Recall	68.42	77.38	**89.47**
F1-score	72.63	81.25	**94.44**
ROC-AUC	88.78	81.4	**97.5**

5. Results and Discussion

After running the trained models on the test dataset, we obtained the results shown in Table 3.

Although the best model regarding training-validation results was Random Forest, on new data, XGBoost was the more effective, as it presents higher values for all metrics chosen (Table 3). Figure 4 shows the ROC-AUC curve for all three classifiers with the test dataset. The dashed line in a ROC-AUC graph represents the curve for a purely random classifier. In a visual analysis, the closer a curve is to the upper left corner of the chart, the better the classifier will be. This means that this classifier has a high true positive rate –a low number of false negatives – and a low false positive rate - i.e., a low number of false positives.

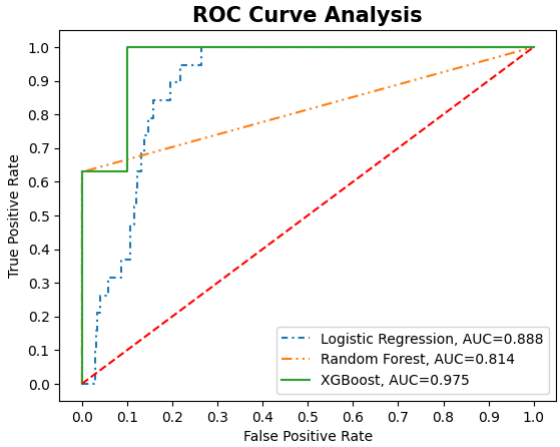

Figure 4. ROC-AUC for the test dataset.

The ROC-AUC values for Logistic Regression and Random Forest classifiers were

lower, which means that the XGBoost is the best of the analyzed classifier for the problem. It is worth noting that both Random Forest and XGBoost classifiers had no False Positives for the testing dataset. Still, the Random Forest model had a higher rate of False Negatives, based on its recall and F1-score results – the same analysis applies to the Logistic Regression classifier. This is far from a desirable situation since the classifier can miss some valuable information in the event of an attack on the DMBS.

Despite utilizing standard ML classification models as its basis, our work is a good baseline for future work. Therefore, its most meaningful contribution is the novelty of using existing internal DBMS mechanisms to increase its security. Our literature review shows that this approach is underexplored in the literature. Our proposal is relevant for those interested in improving their system security since it has a straightforward implementation and brings an additional layer of protection to database infrastructures. As a baseline, this study opens the possibility of further research using several approaches, such as the use of Deep Learning techniques. It also makes it possible to delve into alternative needs, such as improving other information security pillars for such a system.

It is essential to mention that we used a real-world test dataset. This shows that not only is it possible to detect DDoS attacks on a DBMS using its data logs, but also it is possible to achieve high confidence in the predictions.

6. Conclusion

This work presented a novel alternative for identifying DDoS attacks on DBMSs. Classifying these attacks using a traditional packet analysis concept poses different challenges and adds protection to other parts of the infrastructure, such as HTTP servers. Instead of using well-studied approaches based on analyzing packets captured in the network, we looked into the DBMS log table. The analyzed tables include records of the sessions created in the database server and contain information that enabled us to use ML to detect attacks on databases. As a result, attacks were detected with an accuracy of 95.72%, recall of 89.47%, ROC-AUC of 97.5%, and F1-score of 94.44% on our test dataset. This is evidence that our proposal effectively detected this kind of threat in such a dataset.

As for future work, we plan to expand this research by changing the approach from a classification standpoint to detecting anomalous behavior in a database using the same log tables. This anomaly detection will adopt Deep Learning techniques and make it possible to determine other types of database attacks in addition to DDoS attacks.

References

Akgun, D., Hizal, S., and Cavusoglu, U. (2022). A new ddos attacks intrusion detection model based on deep learning for cybersecurity. *Computers & Security*, 118:102748.

Aliero, M. S., Qureshi, K. N., Pasha, M. F., Ghani, I., and Yauri, R. A. (2020). Systematic review analysis on sqlia detection and prevention approaches. *Wireless Personal Communications*, 112(4):2297–2333.

Alkasassbeh, M., Al-Naymat, G., Hassanat, A. B., and Almseidin, M. (2016). Detecting distributed denial of service attacks using data mining techniques. *International Journal of Advanced Computer Science and Applications*, 7(1).

Alwan, Z. S. and Younis, M. F. (2017). Detection and prevention of sql injection attack: A survey. *International Journal of Computer Science and Mobile Computing*, 6(8):5–17.

Berman, D. S., Buczak, A. L., Chavis, J. S., and Corbett, C. L. (2019). A survey of deep learning methods for cyber security. *Information*, 10(4):122.

Brooks, R. R., Yu, L., Oakley, J., Tusing, N., et al. (2021). Distributed denial of service (ddos): A history. *IEEE Annals of the History of Computing*.

Burkov, A. (2020). *Machine Learning Engineering*. Andriy Burkov.

Carrington, A., Manuel, D., Fieguth, P., Ramsay, T., Osmani, V., Wernly, B., Bennett, C., Hawken, S., McInnes, M., Magwood, O., Sheikh, Y., and Holzinger, A. (2022). Deep roc analysis and auc as balanced average accuracy for improved classifier selection, audit and explanation. *IEEE transactions on pattern analysis and machine intelligence*, PP.

Cavalcante, I. C., Meneguette, R. I., Torres, R. H., Mano, L. Y., Gonçalves, V. P., Ueyama, J., Pessin, G., Nze, G. D. A., and Filho, G. P. R. (2022). Federated system for transport mode detection. *Energies*, 15(23):9256.

Chadd, A. (2018). Ddos attacks: past, present and future. *Network Security*, 2018(7):13–15.

Doan, Q. H., Mai, S.-H., Do, Q. T., and Thai, D.-K. (2022). A cluster-based data splitting method for small sample and class imbalance problems in impact damage classification[formula presented]. *Applied Soft Computing*, 120.

Géron, A. (2017). Hands-on machine learning with scikit-learn and tensorflow: Concepts. *Tools, and Techniques to build intelligent systems*.

Gormez, Y., Aydin, Z., Karademir, R., and Gungor, V. C. (2020). A deep learning approach with bayesian optimization and ensemble classifiers for detecting denial of service attacks. *International Journal of Communication Systems*, 33(11):e4401.

Gümüşbaş, D., Yıldırım, T., Genovese, A., and Scotti, F. (2020). A comprehensive survey of databases and deep learning methods for cybersecurity and intrusion detection systems. *IEEE Systems Journal*, 15(2):1717–1731.

Gurina, A. and Eliseev, V. (2019). Anomaly-based method for detecting multiple classes of network attacks. *Information*, 10(3):84.

Haider, S., Akhunzada, A., Mustafa, I., Patel, T. B., Fernandez, A., Choo, K.-K. R., and Iqbal, J. (2020). A deep cnn ensemble framework for efficient ddos attack detection in software defined networks. *Ieee Access*, 8:53972–53983.

Hashem, I., Islam, M., Haque, S. M., Jabed, Z. I., and Sakib, N. (2021). A proposed technique for simultaneously detecting ddos and sql injection attacks. *International Journal of Computer Applications*, 975:8887.

Kaur, H., Pannu, H. S., and Malhi, A. K. (2019). A systematic review on imbalanced data challenges in machine learning: Applications and solutions. *ACM Computing Surveys (CSUR)*, 52(4):1–36.

Kernbach, J. M. and Staartjes, V. E. (2022). Foundations of machine learning-based clinical prediction modeling: Part ii—generalization and overfitting. *Machine Learning in Clinical Neuroscience*, pages 15–21.

Lima Filho, F. S. d., Silveira, F. A., de Medeiros Brito Junior, A., Vargas-Solar, G., and Silveira, L. F. (2019). Smart detection: an online approach for dos/ddos attack detection using machine learning. *Security and Communication Networks*, 2019.

Medeiros, I., Beatriz, M., Neves, N., and Correia, M. (2019). Septic: detecting injection attacks and vulnerabilities inside the dbms. *IEEE Transactions on Reliability*, 68(3):1168–1188.

Mittal, M., Kumar, K., and Behal, S. (2022). Deep learning approaches for detecting ddos attacks: a systematic review. *Soft Computing*, pages 1–37.

Molina, A., Gonçalves, V., Jr., R. S., Giuntini, F., Pessin, G., Meneguette, R., and Filho, G. R. (2022). Weapon: Uma arquitetura para detecção de anomalias de comportamento do usuário. In *Anais do XI Brazilian Workshop on Social Network Analysis and Mining*, pages 121–132, Porto Alegre, RS, Brasil. SBC.

Oracle (2023). Database reference - v$session. `https://docs.oracle.com/en/database/oracle/oracle-database/19/refrn/V-SESSION.html`. [Online; accessed in 03-11-2023].

Pouyanfar, S., Sadiq, S., Yan, Y., Tian, H., Tao, Y., Reyes, M. P., Shyu, M.-L., Chen, S.-C., and Iyengar, S. S. (2018). A survey on deep learning: Algorithms, techniques, and applications. *ACM Computing Surveys*, 51(5).

Sofi, I., Mahajan, A., and Mansotra, V. (2017). Machine learning techniques used for the detection and analysis of modern types of ddos attacks. *Int. Res. J. Eng. Technol.*

Souza, A., Nobre, R., Gonçalves, V., and Filho, G. R. (2021). Uma solução em névoa via objetos inteligentes para lidar com a heterogeneidade dos dados em um ambiente residencial. In *Anais Estendidos do XXXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 257–264, Porto Alegre, RS, Brasil. SBC.

Thudumu, S., Branch, P., Jin, J., and Singh, J. J. (2020). A comprehensive survey of anomaly detection techniques for high dimensional big data. *Journal of Big Data*, 7(1):1–30.

Togatorop, P., Sitorus, H. A. T., Sirait, R. M., and Manurung, T. (2022). Database audit system design and implementation. *Jurnal Mantik*, 5(4):2535–2541.

Tripathi, N. and Hubballi, N. (2021). Application layer denial-of-service attacks and defense mechanisms: a survey. *ACM Computing Surveys (CSUR)*, 54(4):1–33.

Varshney, K. and Ujjwal, R. (2019). Lssqlidp: Literature survey on sql injection detection and prevention techniques. *Journal of Statistics and Management Systems*, 22(2):257–269.

Vedula, V., Lama, P., Boppana, R. V., and Trejo, L. A. (2021). On the detection of low-rate denial of service attacks at transport and application layers. *Electronics*, 10(17):2105.

Yu, L., Zhou, R., Chen, R., and Lai, K. K. (2022). Missing data preprocessing in credit classification: One-hot encoding or imputation? *Emerging Markets Finance and Trade*, 58(2):472–482.