Case-based Explanation of Classification Models for the Detection of SQL Injection Attacks

Juan A. Recio-Garcia[3], Mauricio G. Orozco-del-Castillo[1,2,*] and Jose A. Soladrero[1,2]

[1]Tecnológico Nacional de México/IT de Mérida, Department of Systems and Computing, Merida, Mexico
[2]AAAIMX Student Chapter at Yucatan, Mexico (AAAIMX), Association for the Advancement of Artificial Intelligence, Mexico
[3]Department of Software Engineering and Artificial Intelligence, Instituto de Tecnologías del Conocimiento, Universidad Complutense de Madrid, Spain

Abstract

This paper investigates the interpretability of various machine learning models in the context of detecting SQL injection attacks. The models under investigation include decision trees, multi-layer perceptron, random forests, support vector machines, ADA Boost, naive Bayes, and quadratic discriminant analysis. Our main objective is to gain insights into how these models make decisions when classifying SQL commands as either legitimate or potentially malicious. To achieve this, we employ a set of commonly used features from the literature to train and evaluate the models. By prioritizing explainability, our aim is to uncover the underlying factors and decision rules that drive the models' predictions. Through this research, we seek to bridge the gap between model accuracy and human understanding, thereby facilitating the practical application of machine learning models in real-world security scenarios. The findings of this study contribute to the development of a case-based explanation system that provides valuable insights for cybersecurity practitioners and researchers.

1. Introduction

The increasing reliance on web applications and the growing sophistication of cyber threats have necessitated the development of effective methods for detecting and mitigating SQL attacks. Machine learning (ML) models have shown promising results in identifying such attacks by leveraging their ability to capture complex patterns and relationships in data [1], however, the inherent black-box nature of these models raises concerns regarding their explainability.

Explainability is a critical aspect in the context of cybersecurity [2]. It refers to the ability to understand and interpret how a model arrives at its decisions or predictions. The lack of explainability in ML models can hinder their adoption in security-sensitive domains, where accountability, transparency, and interpretability are paramount [2]. In the case of SQL attack detection, security analysts and stakeholders need to have confidence in the reliability of the

ICCBR XCBR'23: Workshop on Case-based Reasoning for the Explanation of Intelligent Systems at ICCBR2023, July 17–20, 2023, Aberdeen, Scotland
*Corresponding author.
jareciog@fdi.ucm.es (J. A. Recio-Garcia); mauricio.orozco@itmerida.edu.mx (M. G. Orozco-del-Castillo); le18081125@merida.tecnm.mx (J. A. Soladrero)
0000-0001-8731-6195 (J. A. Recio-Garcia); 0000-0001-5793-6449 (M. G. Orozco-del-Castillo); 0000-0001-5864-275X (J. A. Soladrero)

model's outputs and be able to comprehend the underlying rationale behind the identified threats [3].

Understanding the decision-making process of ML models is crucial for building trust and confidence in their outputs [4]. Tree-based models, such as decision trees (DTs), random forests (RFs) and gradient boosting machines, offer inherent interpretability due to their hierarchical structure and explicit decision rules, however, even these models can become increasingly complex and less interpretable as they grow in depth and breadth [3], as is the case with other ML models which are inherently considered as black-boxes, such as artificial neural networks, support vector machines, etc.

In this paper, we aim to investigate the explainability of several ML models for the detection of SQL injection attacks, particularly, DTs, multi-layer perceptron (MLP), random forests (RFs), support vector machines (SVMs), ADA Boost, naive Bayes (NB) and quadratic discriminant analysis (QDA). Our objective is to shed light on the interpretability of these models when attempting to classify a dataset of SQL commands, both legitimate and considered as possible attacks, by using a set of features commonly found in the literature [1, 3]. The objective is to enable insights into the features and rules driving their predictions. Through the development of a case-based explanation system, we strive to bridge the gap between model accuracy and human comprehension, ultimately facilitating the effective utilization of ML models in real-world security scenarios.

This paper is organized as follows: Section 2 provides an overview of related work on explainable ML models and the detection of SQL attacks. Section 3 describes the dataset and evaluates the performance of several state-of-art ML models for the classification of SQL attacks. We then present in Section 4 our methodology for the explainability of these models, discussing the insights gained from the application of the explainable artificial intelligence (XAI) techniques. Subsequently, in Section 5 we present a case-based explanation approach that exploits the conclusions of the XAI analysis to provide useful explanation examples to the end user. Section 6 analyzes and evaluates its performance. Concluding remarks are discussed in Section 7, including a summary of our findings and implications.

2. Background

Cybersecurity is in charge of protecting everything that can connect to a different digital device, it can be divided into two parts, the systems' security information and the physical security of any devices [5]. In the last few years it has gained particular interest due to, partly, the market of cybersecurity has been valued well over the $156 billion mark, with expectations up to $350 billion by 2026 [2].

One of the most common types of cybersecurity attacks are injection attacks. Injection attacks happen when an attacker makes an input which the system is not ready to receive; this allows the attacker to execute malicious commands which can result in a diversity of negative effects such as data leakage or giving access to users who should not be authorized to have them [1]. In an SQL injection attack the attacker uses a SQL sentence with the objective of extracting or modifying data from the database, this can lead to huge risks to any system since it can give the attackers the ability to even delete the whole database or grant access to anyone they desire [1].

These risks have led to explore AI techniques [2]s, particularly ML models, for the detection of SQL injection attacks.

However, it is considered that the more powerful a ML model is, the less interpretable and explainable it becomes. This issue is particularly important in the field of cybersecurity considering that fundamental decisions cannot be trusted to a system that cannot explain itself. This led to the field of XAI, which aims to shed light on intelligent systems without compromising their performance. Explanability attempts to solve the question as to why a model reaches a given conclusion, since some models are used in a high-risk enviroment, it is important to comprehend and not just blindy trust it because it is possible the model might not have enough information to understand the situation as a whole and give an incorrect prediction [6].

XAI techniques in cybersecurity have focused on the following fields: Intrusion Detection Systems, Malware Detection, Phishing and Spam Detection, and BotNet Detection [2]. Less works have focused on fields such as Fraud Detection, Zero-Day Vulnerabilities, Digital Forensics, and Crypto-Jacking [6, 2]. Injection attacks, despite being considered one of the main threats to cybersecurity [7], which has in turn led to the development of several ML models for their detection [1], has not been considerably explored using XAI techniques [2].

However, other cybersecurity fields have extensibly been favored by XAI techniques. For instance, the explainability of Intrusion Detection Systems has been explored using rule-based models [8] and Shapley Additive Explanations (SHAP) [9, 10, 11, 12]. The SHAP framework assigns values to features based on its marginal contribution to the prediction when considering all possible combinations of features. Other works have relied on Local Interpretable Model-agnostic Explanations (LIME), which aims to explain individual predictions of a model by approximation the model's behavior in a local neighborhood around the instance of interest, which creates a simpler, more interpretable model, such as a linear regression model. In [13] the authors propose the deduction of features and characteristics of a SVM model using LIME, which allows the detection of an adversarial attack which would otherwise fool the model. Malware Detection has also recurred to LIME, for instance, to identify locations considered important by a convolutional neural network in the opcode sequence of an Android application [14] and to calculate scores for words showing the output's significance of another convolutional neural network [15]. In terms of BotNet Detection, [16] describes BotStop, a detection system that examines incoming and outgoing network traffic to prevent infections; BotStop is founded on SHAP use with features extracted from network packets. SHAP is also used in BotNet Detection to determine the relevant traffic features in a framework [17] and, along with LIME, Anchors, Counterfactual Explanation, and Open Source Intelligence (OSINT), to prevent Botnet Domain Generation Algorithm (DGA) for Cyber Threat Intelligence Sharing.

3. Dataset and Model

The dataset of SQL Injection Attacks was obtained from [18]. After eliminating repeated data, we obtained a total of 30,876 instances, from it the non-related SQL instances were removed, leaving only SQL malicious and non-malicious sentences. The final dataset contains 22,931 instances with a balanced distribution of the binary target class (11365: Attack, 11566: No

Model	DT	MLP	RF	SVM	ADA	NB	QDA
Accuracy	.979	.980	.975	.981	.980	.895	.636
F1	.979	.982	.975	.981	.980	.895	.572

Table 1
Accuracy and F1 metrics obtained by different classification models. DT: Decision Trees, MLP: Multilayer Perceptron, RF: Random Forest, KNN: K-Nearest Neighbours, SVM: Support Vector Machine, ADA: ADA Boost, NB: Naive Bayes, QDA: Quadratic Discriminant Analysis

Attack).

From the SQL sentence 82 binary and numerical features were obtained through the analysis of the existing literature [19, 20, 21, 22, 23, 24, 25, 26, 27, 28]. Table 2 describes the features being considered (located at the end of the paper). For example, most features are the use of SQL keywords like *select*, *from*, *where*. These are usual keywords in SQL statements that may indicate a malicious attempt to obtain information from the database. Other keywords like *drop*, *delete*, *alter*, or *update* clearly indicate an injection attack to the database. There are also some features that take into account the count of repetitions of a concrete keyword or even more elaborated expressions commonly used in SQL injection attacks (for example, the use of equality expressions such as "$1 = 1$" to avoid the use of boolean values). After removing features with one single unique value, the resulting dataset contains 74 features, that we denote as $\mathcal{F}$.

Next, different ML classification models were trained and evaluated using 5-fold cross-validation. Similarly to other previous results from the literature, they all achieved very high accuracy, as presented in Table 1. These results led us to initially presume that there was a direct correlation between some of the features and the target label. Thus, the correlation table was computed between all the features (including the target variable). This table is presented in Figure 1 where column/row 0 represents the target variable. As we can observe, there is no correlation between any of the dataset's features. Another possible explanation for this remarkable accuracy is that the combination of values from a bounded subset of features leads to a clear classification. To discard this hypothesis, we analyzed the decision tree model, as is a white-box ML technique. The corresponding tree is presented in Figure 2, which clearly led us to reject this hypothesis.

After discarding any correlation or simple direct proportion between features that led to the classification as an attack, we can conclude that the ML models are very suitable tools for this task, as they achieve remarkable accuracy even though this is not a simple classification problem. In the following section, we introspect these models with XAI techniques to infer which are the most relevant features for detecting SQL injection attacks.

4. XAI Analysis

Explainable Artificial Intelligence (XAI) plays a crucial role in providing transparency to black-box ML models. While often robust and accurate, these models can be challenging to understand due to their complex internal mechanisms. This is the case of the ML models trained in the previous section: they achieve very high accuracy, although there is no information about

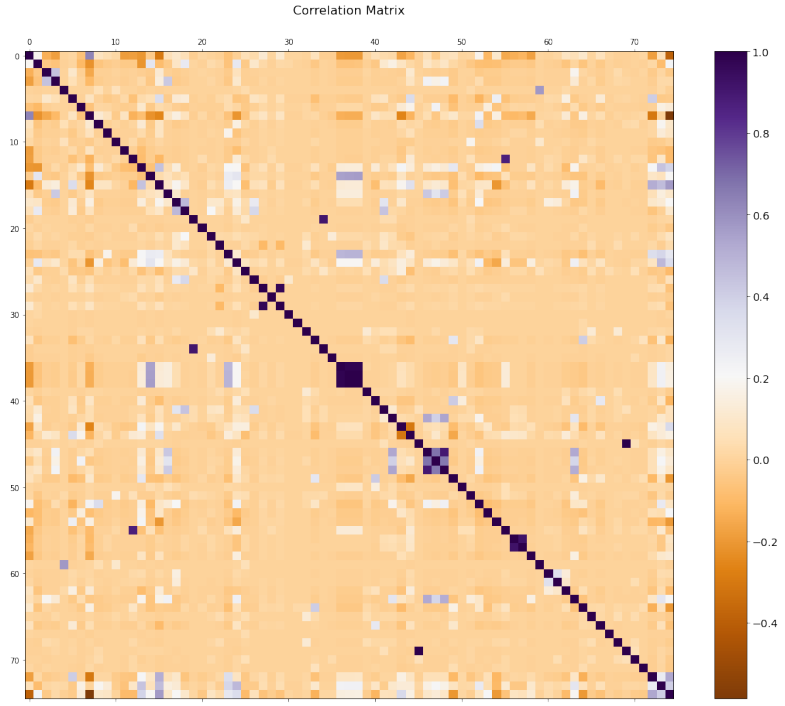

Figure 1: Heatmap representing feature correlation. The first column/row represents the target variable.

which features are the most relevant to identify SQL injection attacks.

XAI techniques aim to bridge this gap by shedding light on how these models make predictions or decisions. By using methods such as feature importance analysis, rule extraction, or local explanations, XAI allows users to gain insights into the factors that influence the model's output. This transparency not only helps in understanding the decision-making process but also assists in identifying biases, errors, or potential vulnerabilities.

To provide transparency to the previous ML models, we have focused on the following established XAI methods: feature importance, permutation importance and Shapley Additive exPlanations (SHAP). As most of the ML techniques evaluated in the previous section achieve a similar performance (with a difference of 0.5%) we have chosen the Random Forest model for its analysis. The decision tree structure of Random Forests provides inherent interpretability, as the paths from the root to the leaves can be traced to understand how the model arrives at its predictions.

The *feature importance* method computes the relative contribution of different input features on the predictions made by a machine learning model. It aims to provide insights into which features are most influential in driving the model's output. The feature importance in Random Forests can be easily obtained by analyzing the average impurity reduction (Mean Decrease in Impurity, MDI) across all trees. The *permutation importance* method evaluates the feature's impact on the model's performance when their values are randomly shuffled. The logic behind this method is that significant features, when shuffled, will cause a more significant drop in the

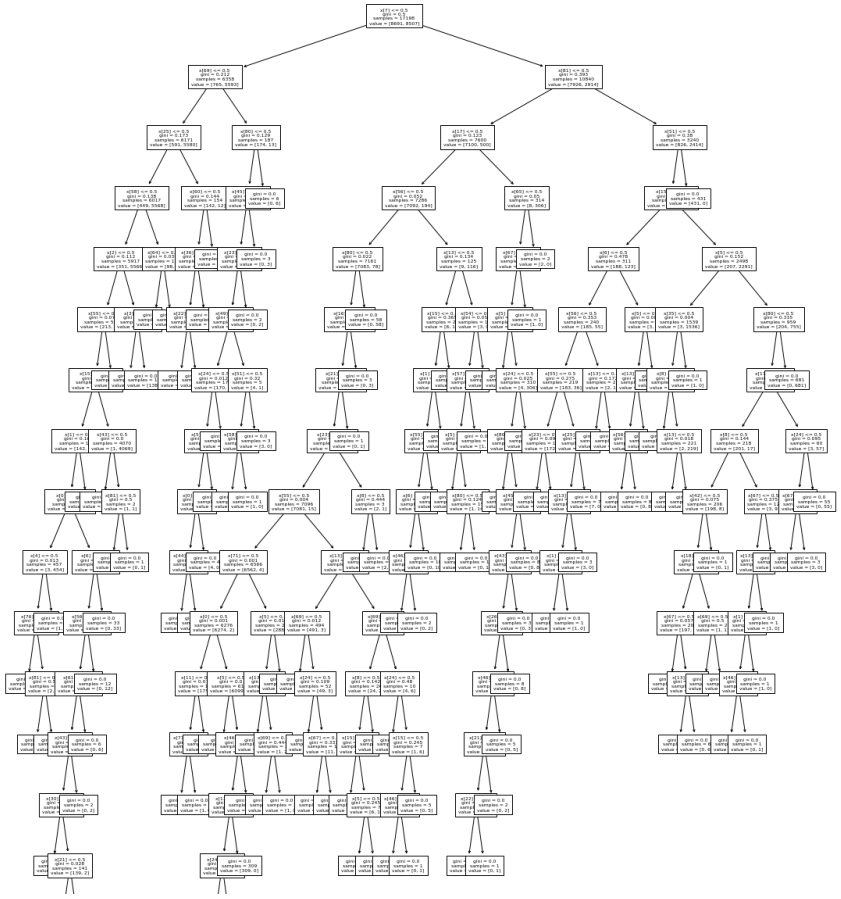

Figure 2: Partial visualization of the decision tree demonstrating the complexity of the model. The two branches on the left side contain up to 15 additional levels.

model's performance, indicating their crucial role in making accurate predictions. Extending this concept, the SHAP (Shapley Additive exPlanations) method is based on cooperative game theory and considers the interactions and dependencies between feature subsets to provide a fair allocation of the model's prediction to individual features.

Applying these explanation methods to the RF model, we obtain the results displayed in Figure 3, where we have selected the 15 most relevant features for each method. As we can observe, *F8* (*'from'* keyword) is the most relevant feature to classify a SQL sentence as an attack, followed by *F16* (=) and *F81* (*hidden equality expression).* From the intersection of the resulting feature lists, we obtain the 11 most significant explanation features: $\mathcal{F}_e = \{F1, F2, F6, F8, F16, F24, F25, F52, F57, F81, F82\}$[1] from the whole original set $\mathcal{F}$ with 74 features used to train the model.

The next step is analyzing the impact (positive or negative) of the values of these most significant features. To do so, we compute the class distribution (*Attack* or *No Attack*) of the instances having a concrete feature value. As all features in $\mathcal{F}_e$ are boolean values representing

[1]see Table 2 for the corresponding description

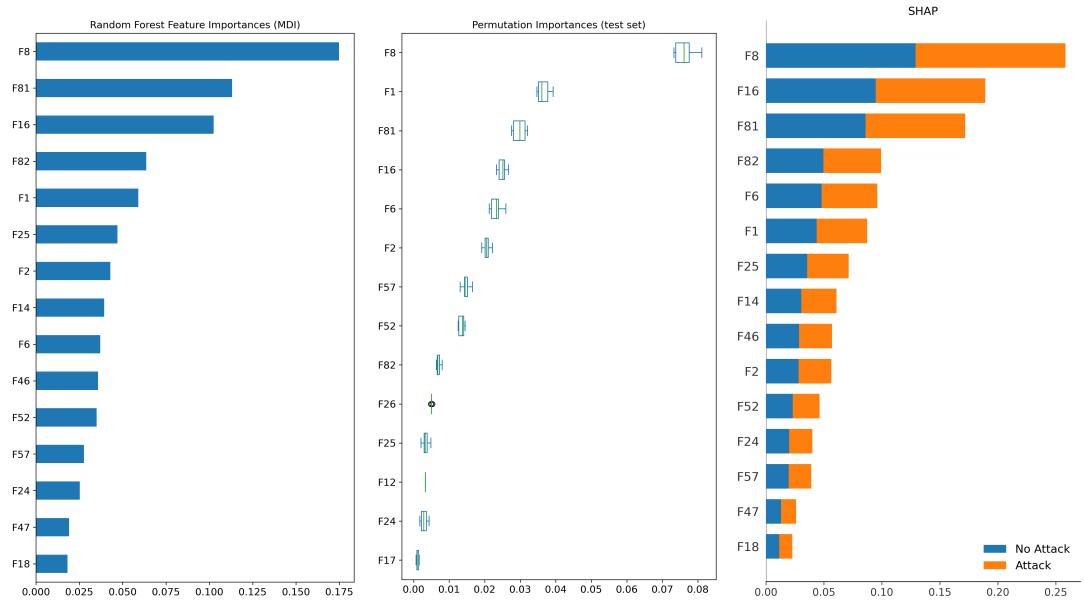

Figure 3: Most relevant features according to feature importance (left), permutation importance (center), and SHAP (right)

that the feature is *found* or *not found* in the SQL sentence, we obtain the two charts presented in Figure 4. These charts represent the percentage of instances (among the total) considered as an attack or not when a feature is found (left chart) or not (right chart). Through the class distributions reported in these charts, we can understand the most significant patterns or rules found by the random forest classifier. For example, the presence of a *hidden equality expression* (F81) such as "1 = 1", that is commonly used to masquerade an attack, has a majority percentage of instances labeled as attacks (Figure 4 left). Another usual keyword in SQL attacks is *sleep* (F57), that, when found, also presents a majority percentage of sentences considered an attack. On the other hand, the absence of common keywords such as *from* (F8) or *select* (F1) is also a clear indication of a potential attack (Figure 4 right).

5. Case-based Explanation

After the analysis of the most relevant features of the dataset for the classification of SQL attacks by an ML model, we can move forward to developing a case-based explanation system. By drawing parallels between the current query and similar explanation cases from the dataset, case-based explanations provide a transparent rationale for the ML classification, enhancing the interpretability of the underlying model. In our concrete scenario, SQL sentences similar to a query are presented to the user. These explanation cases corroborate the prediction made by the ML model regarding the malicious nature of the query.

We propose a case-based explanation process following the MAC/FAC (Many-Are-Called, Few-Are-Chosen) schema [29]. The *filtering step* (MAC) takes advantage of the XAI analysis presented in the previous section to select only explanation cases e_c with the same values as the

Figure 4: Accumulated bar charts representing the percentage of instances (among the whole dataset) labeled as *attack* or *no attack* when a feature (y-axis) is found (left chart) or not (right chart)

query q for the most significant explanation features $\mathcal{F}_e$. This way, we maximize the explicability of the examples presented to the user. This step filters the compatible SQL sentences in the dataset C according to hard restrictions given by the presence or lack in the query of the critical explanation features: *from*, *select*, *sleep*, etc. It also takes into account if the actual label of the explanation case $e_c.l$ and the label predicted by the model $pred(q)$ match. The corresponding MAC function is defined as:

$$MAC(q) = \{e_c \in C : q.l = pred(e_c) \wedge q.f = e_c.f,\ \forall f \in \mathcal{F}_e\} \tag{1}$$

The *selection step* (FAC) obtains the most similar explanation cases to q using a similarity metric that compares the remaining features in the description of a SQL sentence $\mathcal{F}$. This step applies a minimum similarity threshold θ and the inverse of the Euclidean distance as the similarity metric:

$$FAC(q) = \{e_c \in MAC(q) : sim(q, e_c) > \theta\} \tag{2}$$

$$sim(q, e_c) = 1 - \sqrt{\sum (q.f - e_c.f)^2},\ \forall f \in \mathcal{F} \tag{3}$$

In the following section, we present the evaluation of the proposed case-based explanation method following the MAC/FAC approach.

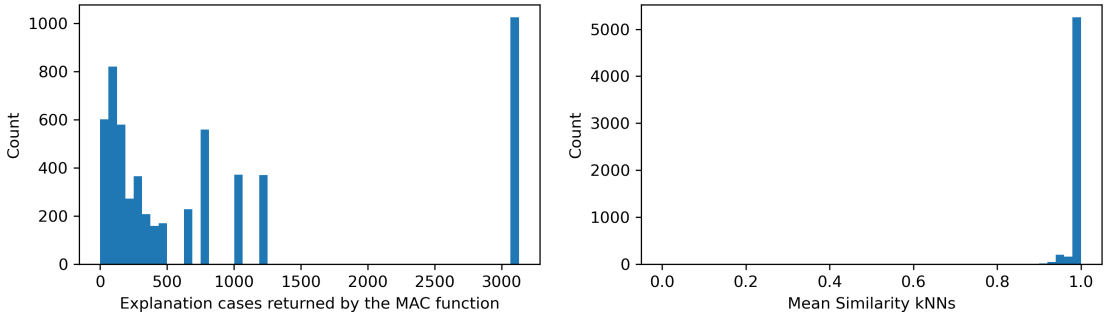

Figure 5: *LEFT*: MAC filtering step. Histogram representing the distribution of instances in the test set (y-axis) according to the corresponding number of explanation cases returned by the MAC function (x-axis). *RIGHT*: FAC selection step. Histogram representing the distribution of the mean similarity scores of the 3-NNs (x-axis) for the instances in the test set. Nearest neighbors are selected after the MAC filter.

6. Evaluation

To evaluate the case-based explanation method we have split the dataset into a 25%-75% test and training sets. The filtering step (MAC) is evaluated by computing how many explanation cases from the training set are obtained, this is, how many match with the query for the most significant explanation features $\mathcal{F}_e$. This process is repeated for each case in the test set. Results are shown by the histogram in Figure 5 (left). As we can observe, the histogram shows that the MAC filter is able to find a large number of explanation cases obeying the restrictions given by the explanation features in $\mathcal{F}_e$. This is mainly due to the very large number of instances in the dataset To evaluate the selection step (FAC) we computed the mean similarity of the k-NNs to the query. Figure 5 (right) shows the corresponding histogram for $k = 3$. As we can observe, our case-based explanation system is able to find explanation cases that are very similar to the query. Again, this is rooted in the large number of options given by the previous MAC step.

7. Conclusions

This paper reports an in-depth analysis of the performance of ML models for detecting SQL injection attacks. The first contribution of the paper is the analysis and collection of the features considered by previous works for training such models. Then a novel dataset represented by these features is developed from a large collection of SQL sentences labeled according to their malicious intents. From this dataset, several ML models are trained and evaluated, demonstrating the high accuracy of these techniques for identifying SQL attacks. However, the black-box nature of these ML models does not allow us to understand the most significant features that denote a SQL attack. Therefore, we apply several XAI techniques to identify the features that characterize malicious SQL sentences. Finally, these relevant features let us build a case-based explanation system based on the MAC/FAC schema. This CBR system enables the understanding of why a SQL sentence is considered an attack by the ML model through the comparison between such sentences and similar explanation cases.

Table 2
Features table

Feature	Description / Keyword
F1	select
F2	union
F3	update
F4	set
F5	alter
F6	where
F7	like
F8	from
F9	table
F10	database
F11	drop
F12	delete
F13	insert
F14	And | Or
F15	null
F16	=
F17	information_schema
F18	user
F19	version
F20	load_file
F21	save
F22	! | # | % | $ | NUL | SOH | STX | ETX | EOT | @
F23	&
F24	|
F25	,
F26	;
F27	\
F28	+ | - | * | /
F29	commit | rollback | grant | revoke | declare | remove
F30	count(;)
F31	/* | */
F32	\x
F33	\u
F34	connection
F35	xor
F36	inner join
F37	file | load_file | load_data_infile | into_outfile | into_dumpfile
F38	OS | Operative System | exec
F39	count(STRING)
F40	count(SUBSTR)
F41	count(SUBSTRING)
F42	count(MID)
F43	count(ASC)
F44	count(<)
F45	count(>)
F46	count(')
F47	count("")
F48	exists
F49	floor
F50	rand
F51	group
F52	order
F53	lenght
F54	ascii
F55	if
F56	count
F57	sleep
F58	between
F59	values
F60	delay
F61	wait
F62	benchmark
F63	indentity_insert
F64	truncate table
F65	username | password
F66	user | pass
F67	")"
F68	limit
F69	concat
F70	ne
F71	find
F72	eq | gt | gte | lt | it | lte | ite | in | nin
F73	mod | regex | text
F74	return
F75	return true | return 1
F76	exists
F77	create
F78	show
F79	collection
F80	while(loop)
F81	hidden equality expression
F82	large expression

Acknowledgments

Supported by the PERXAI project PID2020-114596RB-C21, funded by the Ministry of Science and Innovation of Spain (MCIN/AEI/10.13039/501100011033) and the BOSCH-UCM Honorary Chair on Artificial Intelligence applied to Internet of Things.

References

[1] M. Alghawazi, D. Alghazzawi, S. Alarifi, Detection of SQL Injection Attack Using Machine Learning Techniques: A Systematic Literature Review, Journal of Cybersecurity and Privacy 2 (2022) 764–777. doi:10.3390/jcp2040039.

[2] N. Capuano, G. Fenza, V. Loia, C. Stanzione, Explainable artificial intelligence in cybersecurity: A survey, IEEE Access 10 (2022) 93575–93600.

[3] M. S. Roobini, S. R. Srividhya, Sugnaya, K. Vennela, G. Nikhila, Detection of SQL Injection Attack Using Adaptive Deep Forest, 2022 International Conference on Communication, Computing and Internet of Things, IC3IoT 2022 - Proceedings (2022). doi:10.1109/IC3IOT53935.2022.9767878.

[4] S. Hariharan, A. Velicheti, A. Anagha, C. Thomas, N. Balakrishnan, Explainable artificial intelligence in cybersecurity: A brief review, in: 2021 4th International Conference on Security and Privacy (ISEA-ISAP), IEEE, 2021, pp. 1–12.

[5] P. By, D. Kumar, G. By, M. Mehta, Explainable Artificial Intelligence for Cyber Security, volume 1025 of *Studies in Computational Intelligence*, Springer International Publishing, Cham, 2022. URL: https://link.springer.com/10.1007/978-3-030-96630-0. doi:10.1007/978-3-030-96630-0.

[6] P. By, D. Kumar, G. By, M. Mehta, Explainable Artificial Intelligence for Cyber Security, volume 1025 of *Studies in Computational Intelligence*, Springer International Publishing, Cham, 2022. URL: https://link.springer.com/10.1007/978-3-030-96630-0. doi:10.1007/978-3-030-96630-0.

[7] R. M. Thiyab, M. Ali, F. Basil, et al., The impact of sql injection attacks on the security of databases, in: Proceedings of the 6th International Conference of Computing & Informatics, School of Computing, 2017, pp. 323–331.

[8] T. Dias, N. Oliveira, N. Sousa, I. Praça, O. Sousa, A hybrid approach for an interpretable and explainable intrusion detection system, in: Intelligent Systems Design and Applications: 21st International Conference on Intelligent Systems Design and Applications (ISDA 2021) Held During December 13–15, 2021, Springer, 2022, pp. 1035–1045.

[9] M. Wang, K. Zheng, Y. Yang, X. Wang, An explainable machine learning framework for intrusion detection systems, IEEE Access 8 (2020) 73127–73141.

[10] Y. Wang, P. Wang, Z. Wang, M. Cao, An explainable intrusion detection system, in: 2021 IEEE 23rd Int Conf on High Performance Computing & Communications, IEEE, 2021, pp. 1657–1662.

[11] T.-T.-H. Le, H. Kim, H. Kang, H. Kim, Classification and explanation for intrusion detection system based on ensemble trees and shap method, Sensors 22 (2022) 1154.

[12] S. Wali, I. Khan, Explainable ai and random forest based reliable intrusion detection system (2021).

[13] E. Tcydenova, T. W. Kim, C. Lee, J. H. Park, Detection of adversarial attacks in ai-based intrusion detection systems using explainable ai, Human-centric Comput. Inf. Sci. 11 (2021).

[14] M. Kinkead, S. Millar, N. McLaughlin, P. O'Kane, Towards explainable cnns for android malware detection, Procedia Computer Science 184 (2021) 959–965.

[15] J. Feichtner, S. Gruber, Understanding privacy awareness in android app descriptions

using deep learning, in: Proceedings of the tenth ACM conference on data and application security and privacy, 2020, pp. 203–214.
[16] M. M. Alani, Botstop: Packet-based efficient and explainable iot botnet detection using machine learning, Computer Communications 193 (2022) 53–62.
[17] P. P. Kundu, T. Truong-Huu, L. Chen, L. Zhou, S. G. Teo, Detection and classification of botnet traffic using deep learning with model explanation, IEEE Transactions on Dependable and Secure Computing (2022).
[18] S. S. H. Shah, SQL injection dataset | Kaggle, 2021. URL: https://www.kaggle.com/datasets/syedsaqlainhussain/sql-injection-dataset.
[19] M. Hasan, Z. Balbahaith, M. Tarique, Detection of SQL Injection Attacks: A Machine Learning Approach, 2019 International Conference on Electrical and Computing Technologies and Applications, ICECTA 2019 (2019). doi:10.1109/ICECTA48151.2019.8959617.
[20] H. Gao, J. Zhu, L. Liu, J. Xu, Y. Wu, A. Liu, Detecting SQL injection attacks using grammar pattern recognition and access behavior mining, Proceedings - IEEE International Conference on Energy Internet, ICEI 2019 (2019) 493–498. doi:10.1109/ICEI.2019.00093.
[21] Q. Li, W. Li, J. Wang, M. Cheng, A SQL Injection Detection Method Based on Adaptive Deep Forest, IEEE Access 7 (2019) 145385–145394. doi:10.1109/ACCESS.2019.2944951.
[22] D. Tripathy, R. Gohil, T. Halabi, Detecting SQL Injection Attacks in Cloud SaaS using Machine Learning, Proceedings - 2020 IEEE 6th Intl Conference on Big Data Security on Cloud, BigDataSecurity 2020 (2020) 145–150. doi:10.1109/BigDataSecurity-HPSC-IDS49724.2020.00035.
[23] Q. Li, F. Wang, J. Wang, W. Li, LSTM-Based SQL Injection Detection Method for Intelligent Transportation System, IEEE Transactions on Vehicular Technology 68 (2019) 4182–4191. doi:10.1109/TVT.2019.2893675.
[24] K. Kamtuo, C. Soomlek, Machine Learning for SQL injection prevention on server-side scripting, in: 2016 International Computer Science and Engineering Conference (ICSEC), IEEE, 2016, pp. 1–6. URL: http://ieeexplore.ieee.org/document/7859950/. doi:10.1109/ICSEC.2016.7859950.
[25] D. Das, U. Sharma, D. K. Bhattacharyya, Defeating SQL injection attack in authentication security: an experimental study, International Journal of Information Security 18 (2019) 1–22. doi:10.1007/s10207-017-0393-x.
[26] Ö. Kasim, An ensemble classification-based approach to detect attack level of SQL injections, Journal of Information Security and Applications 59 (2021) 0–3. doi:10.1016/j.jisa.2021.102852.
[27] M. R. Ul Islam, M. S. Islam, Z. Ahmed, A. Iqbal, R. Shahriyar, Automatic detection of NoSQL injection using supervised learning, Proceedings - International Computer Software and Applications Conference 1 (2019) 760–769. doi:10.1109/COMPSAC.2019.00113.
[28] N. M. Sheykhkanloo, Employing Neural Networks for the detection of SQL injection attack, ACM International Conference Proceeding Series 2014-September (2014) 318–323. doi:10.1145/2659651.2659675.
[29] R. L. de Mántaras, D. McSherry, D. G. Bridge, D. B. Leake, B. Smyth, S. Craw, B. Faltings, M. L. Maher, M. T. Cox, K. D. Forbus, M. T. Keane, A. Aamodt, I. D. Watson, Retrieval, reuse, revision and retention in case-based reasoning, Knowl. Eng. Rev. 20 (2005) 215–240. doi:10.1017/S0269888906000646.