

SQLi Detection with ML: A Data-Source Perspective

Balázs Pejó^a and Nikolett Kapui^b

*ELKH-BME Information Systems Research Group, Laboratory of Cryptography and System Security,
Department of Networked Systems and Services, Faculty of Electrical Engineering and Informatics,
Budapest University of Technology and Economics, Műegyetem rkp. 3., H-1111 Budapest, Hungary*

Keywords: SQLi, Machine Learning, Data Distribution.

Abstract: Almost 50 years after the invention of SQL, injection attacks are still top-tier vulnerabilities of today's ICT systems. In this work, we highlight the shortcomings of the previous Machine Learning based results and fill the identified gaps by providing a comprehensive empirical analysis. We cross-validate the trained models by using data from other distributions which was never studied in relation with SQLi. Finally, we validate our findings on a real-world industrial SQLi dataset.

1 INTRODUCTION

One of the biggest security concerns today is Structured Query Language Injection (SQLi), which is also reflected in the OWASP Top 10 List (OWASP,). Furthermore, not only the occurrence but the complexity and the severity are increasing of the SQLi cases, so faster and easier methods are needed to tackle this problem. Following the recent success of Machine Learning (ML) in many fields, traditional SQLi detection techniques are also being challenged by ML techniques (Jemal et al., 2020).

In this short work we highlight the shortcomings of the previous ML-based results focusing on 1) the evaluation methods, 2) the optimization of the model parameters, 3) the distribution of utilized datasets, and 4) the feature selection. Since none of the previous works explored these aspects in depth, we fill this gap, i.e., we compare different types of ML algorithms with various pre-processing methods. Additionally, we cross-verified the models on datasets corresponding to different distributions than the training samples. We also validate our findings on a private SQLi dataset originating from a major player in the security industry in Europe.

Our findings revealed that the model with the highest accuracy is not necessarily the best choice 1) when a specific (e.g., low) false positive rate is desired and 2) when the model is used on data from other distributions. Our goal is to raise awareness of the is-

sues using pre-trained off-the-shelf ML models and to ease the choice of security engineers in selecting the proper setup for specific use cases.


Disclaimer. The full version of the paper (with extended background, scenario recommendations, etc.) is available at ArXiv (Pejo and Kapui, 2023).


2 PRELIMINARIES

2.1 SQL Injection

SQL is a query language for relational databases to help modify, retrieve, and store data. There are many dialects, such as MySQL, PostgreSQL, and SQLite. SQL Injection is a server-side attack where a web security vulnerability allows attackers to alter the SQL queries made to the central database; therefore, they can retrieve information from or about the database, which often comes with the leakage of sensitive data.

There are three main categories of SQLi: in-band, out-of-band, and blind. The in-band SQLi can be either Error-based or Union-based, where the attacker uses the same channel for attacking and receiving results. In contrast, in out-of-band SQLi, the query response returns on a different channel, usually by utilizing HTTP, DNS, or FTP. Finally, the blind SQLi can be either Content-based or Time-based, where the attacker does not rely on response but instead probes the server and observe how it behaves.

^a  <https://orcid.org/0000-0002-1825-9251>

^b  <https://orcid.org/0009-0007-0620-2382>

2.2 ML Techniques

A key technique to tackle SQLi is to use ML: these techniques learn directly from the data and have the potential to detect hidden patterns which would slip through traditional approaches. Below we give a high-level introduction to the data parsing techniques and ML architectures utilized in this work.

Pre-Processing. The raw benign and malicious SQL payloads cannot be fed directly into ML models. We surveyed the relevant literature and identified three parsing techniques that are the most widely utilized.

- *TF-IDF vectorizer* is based on the Bag of Words model that counts how much occurs from a word in a document. It consists of the Term Frequency (TF) and the Inverse Document Frequency (IDF) parts, which measures the frequency of a word in a specific document and the importance of the words across the entire dataset, respectively.
- *Keyword weights* assign weights to SQL keywords based on their maliciousness.
- *Skip-gram model* is a word embedding model (e.g., Word2Vec) that maps every word into a continuous vector space, making it easier to check which ones are similar.

Models. There are many model architectures choices to feed the processed data into. We surveyed the relevant literature and identified five ML architectures that are the most widely utilized.

- *Logistic Regression* (LR) is a linear model that learns to classify the data by minimizing the corresponding error.
- *Support Vector Machine* (SVM) learns to classify the data by maximizing the distance between the classes.
- *Random Forest* (RF) consists of several Decision Trees that operate as an ensemble: the decision is based on the majority vote of the trees.
- *Gradient Boosting* (GB) is learning by minimizing the loss function, which is achieved by adding more weak learners that are concentrating on the areas where the existing ensemble perform poorly.
- *Neural Network* (NN) mimics the human brain, i.e., it is based on a collection of connected neurons where the output of each is computed by some non-linear function.

3 RELATED WORKS

A summary of previous research efforts concerning SQLi detection without ML are surveyed in (Kindy and Pathan, 2011), while (Pattewar et al., 2019) and (Hu et al., 2020) are surveyed the ML solutions. We

inspected the ML-based SQLi literature while focusing on four aspects: the datasets, the features, the models, and the evaluation. We considered 28 papers, which we obtained by forward and backward snowballing from the surveys and by using targeted queries (e.g., "SQL Injection" + "Machine Learning", etc.) in Google Scholar. Our findings are summarized in Table 2.

Dataset. The datasets' size and diversity are imperative; yet, more than a quarter (29%) of works experiments with small (i.e., below 10k) datasets. Although the rest utilize a more data for training, for many of them (32%), the data comes from a single source. Besides, when the authors consider multiple sources (39%), they merely merge them into a single database. On the other hand, we train our models on many separate datasets from different sources and evaluate them in a cross-verification manner.

Features. Few works (18%) only utilize less than a dozen features, which is insufficient to capture the underlying language's richness. Although other works (43%) exploit more features, only some of them (39%) apply over a thousand features (i.e., by using OneHot-Encoding, Word2Vec, String2Vec, or TF-IDF with large datasets), which is needed to capture the abundance of the payloads appropriately.

Models. Almost a third of the works (32%) mentioned in Table 2 consider only a single ML model without any hyper-parameter tuning. This cherry-picking strategy is superficial and, without proper comparison, could be easily misinterpreted. Although other works (57%) consider comparing more off-the-shelf models or fine-tuning a single one, this still not paints a complete picture of the relationship between these models. Finally, similarly to our work, only a handful of papers (18%) evaluate multiple models and utilize parameter optimization.

Evaluation. Few works (18%) present the accuracy metric only, which is inappropriate in the SQLi use-case: the difference between type I and type II errors is crucial. The majority of the works (61%) indeed consider false positives and false negatives and present them either via the confusion matrix or via the precision, recall, and F1 values. However, this still might be insufficient from the usability point of view: any practitioner of an SQLi detection system would require the possibility to set the trade-off between these values, depending on the underlying scenario's sensitivity. Hence, the ROC curve is of the utmost importance. Besides this work, it is measured only half a dozen times (21%).

Table 1: Notation used in Table 2. Acc. and conf. mx are the abbreviations for accuracy and confusion matrix.

	Data	Feature	Model	Evaluation
·	< 10k	< 12	1 w/o Tuning	acc.
○	> 10k, 1 source	[12, 999]	1 w/ Tuning or > 1 w/o Tuning	acc. & conf. mx
●	> 10k, > 1 source	> 1000	> 1 w/ Tuning	acc. & conf. mx & ROC

Table 2: The symbol ·, ○, and ● means insufficient/mediocre/sufficient, as described in Table 1. R, D, F, M, and E means References, Dataset Size, Number of Features, Model Optimization, and Evaluation Metrics respectively.

Reference	D	F	M	E
(Joshi and Geetha, 2014)	·	·	·	○
(Hasan et al., 2019)	·	·	○	●
(Moosa, 2010)	·	○	○	·
(Chen et al., 2018)	·	○	·	●
(Gandhi et al., 2021)	·	○	○	○
(Pham and Subburaj, 2020)	·	○	○	○
(Mishra, 2019)	·	○	●	·
(Krishnan et al., 2021)	·	○	●	○
(Ingre et al., 2017)	○	○	·	○
(Jothi et al., 2021)	○	○	·	○
(Sheykhkanloo, 2015)	○	○	·	○
(Luo et al., 2019)	○	●	·	○
(Yu et al., 2019)	○	●	·	○
(Alam et al., 2021)	○	●	○	·
(Chen et al., 2021)	○	●	○	○
(Ross, 2018)	○	●	○	○
(Uwagbole et al., 2017b)	○	●	●	●
(Li et al., 2019a)	●	·	○	○
(Tripathy et al., 2020)	●	·	○	●
(Tang et al., 2020)	●	·	●	○
(Hosam et al., 2021)	●	○	○	·
(Liu et al., 2020)	●	○	○	·
(Farooq, 2021)	●	○	○	●
(Xie et al., 2019)	●	●	·	●
(Betarte et al., 2018)	●	●	○	○
(Li et al., 2019b)	●	●	○	○
(Uwagbole et al., 2017a)	●	●	○	○
(Gogoi et al., 2021)	●	●	●	○

4 EXPERIMENTS

Besides providing a comprehensive analysis, our main aim is to compare models on different datasets with various sizes coming from distinct distributions. Thus, obtaining appropriate datasets is crucial. For our experiments, we utilized three public datasets with different sizes (small, medium, large) from two sources. We merged three small datasets (OWASP, BurpSuite, and FuzzDB) from GitHub¹ into one we

¹<https://www.github.com/ChrisAHolland/ML-SQL-Injection-Detector/tree/master/data>

called *United* (containing 1133 benign and 7 malicious samples). We also used two datasets from Kaggle², namely *SQLi1* (containing 950 benign and 3000 malicious samples) and *SQLi2* (containing 11424 benign and 22301 malicious samples). Finally, we employed a private dataset (containing 2337 benign and 257 malicious samples) only for testing (referred to as *Company*) from a SIEM of an international SOC operating company with clients all over Europe. Opposed to the first three datasets, the last one is not public. The data belongs to a single client, and it was acquired in-between 2019/08 and 2021/05.

We considered three scenarios to evaluate. Firstly, to give a comprehensive analysis, we review the well-studies IID case (i.e., when the test and train datasets are from the same distribution). Secondly, to measure the robustness of the models against data distribution change, we provide experiments concerning the non-IID case (which was not studied before), namely when the training and the testing data come from a different distribution. Thirdly, to inspect the applicability of the lab-tested models in the real world, we evaluate the trained models on confidential data of an international SOC operator within Europe. When applicable, we randomly split the datasets into training, validation, and testing using 70-10-20 percentages. All our experiments are performed two-fold to mitigate the randomness of the training process.³

Using the Same Distribution for Training/Testing.

Due to the lack of space, we neither show the accuracy nor the confusion matrices but instead present the more informative F1-scores and the ROC curves. The former is visible in Table 3, while the latter is visualized in the first column of Figure 1 for the considered three datasets (*United*, *SQLi1*, *SQLi2*). In Table 3, we also present the best-performing model types (LR, SVM, RF, GB, NN) with the corresponding optimal pre-processing method (TF-IDF, Keyword, Skip-gram) and hyper-parameters. The models trained on *United* (with Skip-gram) have 125 features, the models trained on *SQLi1* (with TF-IDF) have 9683, and the models trained on *SQLi2* have 1455 and 28679 when pre-processed with Skip-gram and TF-IDF respectively.

²<https://www.kaggle.com/datasets/syedsaqlainhussain/sql-injection-dataset>

³Our implementation can be found at https://github.com/nikikapui/sql_i_detection.

In Table 3, one can see that the best-performing setups (pre-processing, model type, hyper-parameters) across different datasets vary greatly. For instance, Skip-gram pre-processing method outperforms TF-IDF on the United dataset, while the opposite trend corresponds to SQLi1, and neither dominates the other on SQLi2. The optimal learning rate for GB and NN and the optimal weight for LR and SVM depend on the underlying dataset. No one model type dominates, i.e., the model obtaining the highest F1-score is different for all three datasets. Hence, it is uttermost important to see how models optimized for one dataset perform on other datasets with different distributions.

In the first column of Figure 1 from the ROC curves, it is visible that independently of the optimal setup (i.e., model type, pre-processing method, hyper-parameters), RF slightly outperforms the other models in the low false positive rate region as it obtains the highest true positive rate. Conversely, when a high false positive rate is tolerated, the models have only negligible differences. Note that the AUC values are all above 0.99 except for the United dataset due to its small size: there is only a single negative sample in its test set.

In addition to these results, we found that the Keyword weights pre-processing method is inferior to both TF-IDF and Skip-gram, as the corresponding results were always about 10% less, even though besides the model parameters, we also tuned the exact weights for this pre-processing method.

Timing Measurements. Besides its prediction power, another essential aspect is the usability of the models, i.e., how much time it takes to train these

Table 3: For all considered public datasets (DS) and models, we present the F1-scores of the best-performing models with the corresponding pre-processing (PP) methods for the training set, the validation set, and the test set where Sg and TI stands for Skip-gram and TF-IDF. The utilized hyper-parameters are also displayed where S, W, K, F, E, L, D, H, and A are Solver, Weight, Kernel, Feature num., Estimator num., Learning rate, Depth, Hidden layer size, and Activation function, respectively.

DS.	PP.	Model	Parameters	Train	Val.	Test
United	Sg	LR	S:newton,W:0.1	99.7%	99.6%	99.8%
	Sg	SVM	K:linear,W:10	99.9%	100%	100%
	Sg	RF	F:32,E:10	100%	100%	99.9%
	Sg	GB	L:0.01,E:1000,D:2	100%	100%	99.8%
	Sg	NN	L:0.001,H:64,A:sigmoid	99.7%	99.6%	99.8%
	SQLi1	TI	LR	S:newton,W:10	98.7%	96.8%
TI		SVM	K:linear,W:1	98.2%	96.3%	97.9%
TI		RF	F:16,E:100	100%	97.2%	97.8%
TI		GB	L:0.1,E:1000,D:4	100%	98.7%	98.7%
TI		NN	L:0.001,H:64,A:sigmoid	93.9%	90.8%	92.8%
SQLi2		TI	LR	S:newton,W:10	99.5%	99.3%
	Sg	SVM	K:poly,W:10	99.3%	99.5%	99.4%
	Sg	RF	F:1,E:100	100%	99.6%	99.6%
	Sg	GB	L:0.1,E:1000,D:8	100%	99.3%	99.3%
	TI	NN	L:0.1,H:64,A:sigmoid	99.6%	99.5%	99.3%

models, what is their sizes, and how fast they can predict. These details are presented in Table 4 for the best-performing models. The training was done on Ubuntu 20.04.4 LTS Linux with 16 CPUs (3.10GHz) and 98 Gb RAM. One can see that while the models achieve comparable performances, both the time and the size values have a considerable variance. Additionally to the model type and the employed hyper-parameters, these differences are a combined result of the corresponding datasets and pre-processing methods. Yet, two trends are visible: LR is always the smallest model, and GB is always the most costly model to be trained. Along with the ROC curve, such information is essential for SOC operators to optimize the trade-off between the usability and prediction performance of the SQLi-detecting ML model.

Using Different Distribution for Training/Testing.

The previous experiments revealed the sensitivity of the setup: similar high F1-scores could be reached with vastly different settings. Opposed to the common IID practice that uses the same distribution for testing and training (by splitting the same dataset), we are focusing on the non-IID case, i.e., measuring the performance of the models on test sets from other distributions. This experiment measures the model's robustness against data distribution. The F1-scores are shown in Table 5, while the ROC curves for all pairwise scenarios are presented in the second and third column of Figure 1.

As expected, the F1-score of the best-performing models drops when tested on other datasets from other distributions. For instance, training on a small dataset could produce completely unreliable models: the result on the top of Table 5 suggests that the models trained on United are essentially reduced to a random guess when tested on SQLi1 and SQLi2. Similar results can be found on the middle top of Figure 1: when trained on the smallest United dataset, the

Table 4: The pre-processing and training time, the model size, and the prediction speed of the best performing models with Skip-gram or with TF-IDF.

Dataset	Mod. & PreP.	Learn. Time	Mod. Size	Pred. Speed
United	LR (S)	0.0258 s	0.002 Mb	0.002 ms
	SVM (S)	0.0204 s	0.023 Mb	0.002 ms
	RF (S)	0.018 s	0.012 Mb	0.001 ms
	GB (S)	1.4178 s	0.646 Mb	0.005 ms
	NN (S)	0.283 s	0.118 Mb	0.054 ms
SQLi1	LR (T)	1.9164 s	0.219 Mb	0.123 ms
	SVM (T)	44.1636 s	72 Mb	3.077 ms
	RF (T)	2.3232 s	7.1 Mb	0.174 ms
	GB (T)	589.8 s	1.5 Mb	0.150 ms
	NN (T)	1.5184 s	7.2 Mb	0.112 ms
SQLi2	LR (T)	57.7254 s	0.631 Mb	0.320 ms
	SVM (S)	35.2536 s	3.4 Mb	0.089 ms
	RF (S)	3.5634 s	4.2 Mb	0.043 ms
	GB (S)	572.54 s	6.8 Mb	0.035 ms
	NN (T)	21.4831 s	21 Mb	0.165 ms

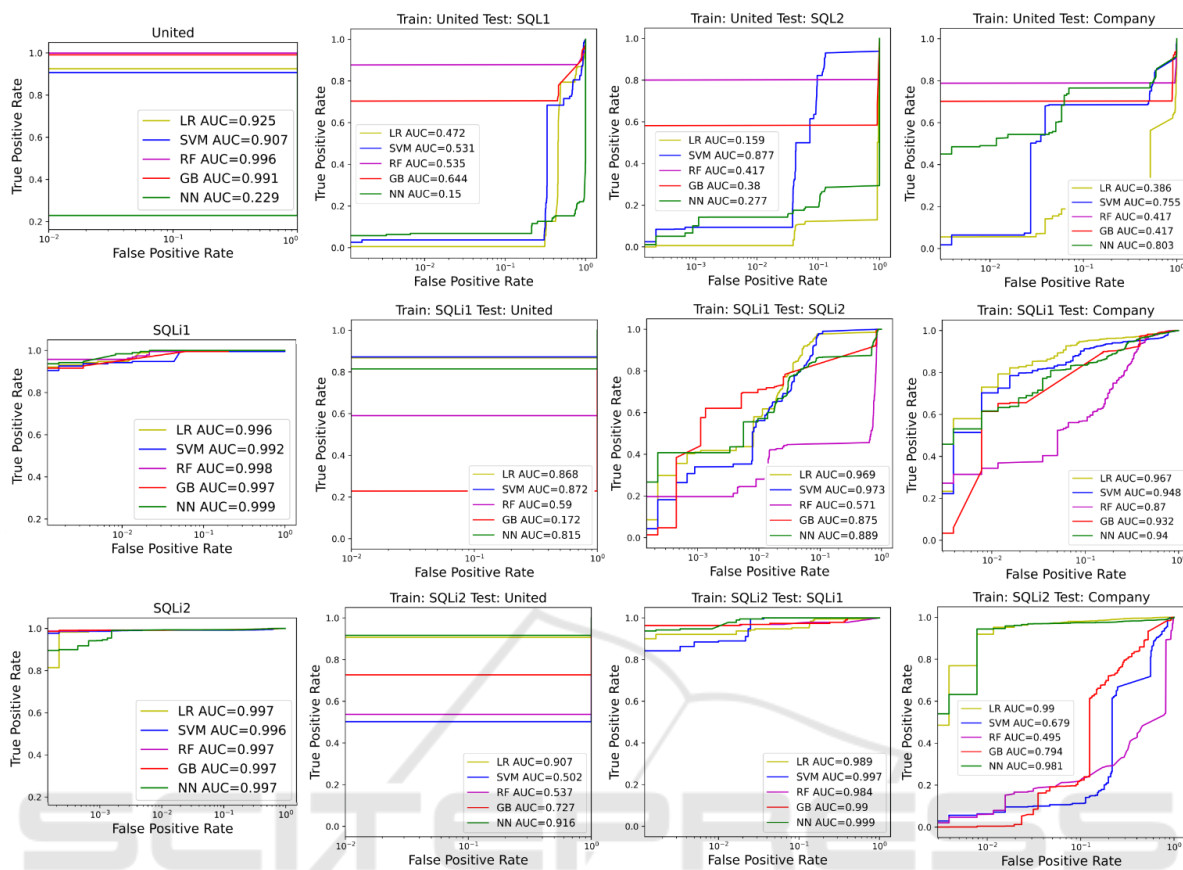


Figure 1: The best performing setting’s ROC curves with the AUC values. 1st column: train and test sets are from the same public distribution. 2nd and 3rd column: train and test sets are from different public distribution. 4th column: train set is the entire public domain and test set is private data.

best models’ AUC is 0.644 and 0.877 when tested on SQLi1 and SQLi2, respectively.

Additionally, the pre-processing method seems crucial too: when trained on SQLi2 and tested on United, models using Skip-gram are idle. In contrast, the ones using TF-IDF have a decent performance. Another interesting finding is that different models could have opposing generalization properties against other distributions. For example, in the middle of Table 5 RF performs exceptionally on United and terribly on SQLi2 when trained on SQLi1. At the same time, the exact opposite trend holds for GB.

Concerning the ROC curves in the second and third column of Figure 1, similarly to the IID case, for this non-IID setup RF is also ideal for low false positive rate region but only when trained on United (left). However, when the models are trained on the large SQLi2 dataset (right), the highest AUC belongs to NN: 0.916 and 0.999 when tested on United and SQLi1, respectively. NN is also a good choice when a low false positive rate is desired.

In addition, when the models are trained on SQLi1

and tested on United (i.e., middle top), SVM is the optimal model choice for sensitive domains where a low false positive rate is required. Yet, when tested on SQLi2 (i.e., middle bottom), multiple models achieve the best trade-off, depending on the desired false positive rate range. What is clear is SVM has the highest AUC values (0.872 and 0.973).

Based on these results, TF-IDF and SVM seem more robust against test data distribution shifts than other methods and models. TF-IDF uses statistical features such as frequencies, which change only slightly when there is a minor change in the underlying distribution. On the other hand, Skip-gram is based on a Neural Network, which takes a predefined input size and could easily overfit, making it rigid to use with different input families. Considering models, SVM is robust, as it maximizes the smallest distance between the benign and malicious samples. Thus it should be tolerant of minor changes in the classes. GB also performed well in this experiment due to its assembly nature. In contrast, NN, the most complex model, might overfit (when trained on United)

Table 5: Dataset-wise the F1-scores with cross-verification of the best performing models with Skip-gram or with TF-IDF.

Tested on	Trained on United				
	LR (S)	SVM (S)	RF (S)	GB (S)	NN (S)
United	99.78%	100%	99.89%	99.78%	99.78%
SQLi1	38.8%	39.0%	38.8%	37.8%	38.8%
SQLi2	50.6%	49.6%	50.6%	50.5%	50.6%

Tested on	Trained on SQLi1				
	LR (T)	SVM (T)	RF (T)	GB (T)	NN (T)
United	92.2%	92.4%	99.8%	76.1%	91.2%
SQLi1	98.43%	97.86%	97.79%	98.68%	92.78%
SQLi2	84.6%	79.7%	52.8%	82.2%	83.2%

Tested on	Trained on SQLi2				
	LR (T)	SVM (S)	RF (S)	GB (S)	NN (T)
United	97.3%	41.8%	50.7%	66.5%	97.5%
SQLi1	95.3%	92.5%	97.8%	98.1%	96.5%
SQLi2	99.36%	99.41%	99.57%	99.30%	99.34%

and lose its generalization capability to tackle samples from other distributions. However, it could also outperform the rest of the models when trained on a large representative dataset, e.g., SQLi2.

Validating the Findings on Private Dataset. Finally, we perform a similar non-IID experiment, but instead of utilizing public datasets for training and testing, we apply the private Company dataset as a test set. The F1-scores of the best performing models are shown in Table 6 while the ROC curves are presented in the last column of Figure 1.

The last column of Table 6 (using the large SQLi2 dataset) elaborates that Skip-gram is indeed not appropriate for models intended to be used on other distributions than the model was trained on. This is seemingly contradicted in the first column; however, that corresponds to the smallest United dataset, which could also produce highly unreliable models, as we showed in Table 5. We hypothesize this excellent result is due to the closeness of United’s and Company’s distribution. Similarly to the previous use cases, the highest F1-score (95.7%) is reached by NN when trained on the biggest dataset using the robust TF-IDF.

Surprisingly, in the last column of Figure 1 (right) shows the simple LR model trained on SQLi2 does outperform NN based on the ROC with a minor AUC difference (0.99 vs. 0.98). Furthermore, LR also performs exceptionally on Company when trained on SQLi1 (middle), making it the best choice even for the low false positive rate domain.

5 CONCLUSION

SQL Injections are top-tier vulnerabilities of today’s ICT systems. As with many other problems, machine

Table 6: F1-scores of the best performing models with Skip-gram or with TF-IDF when tested on the private Company data.

Trained on	United	SQLi1	SQLi2
Model	F1-score on Company		
LR	94.79% (S)	79.08% (T)	92.35% (T)
SVM	90.97% (S)	77.88% (T)	29.98% (S)
RF	92.47% (S)	90.03% (T)	32.68% (S)
GB	91.69% (S)	76.41% (T)	78.47% (S)
NN	94.79% (S)	77.65% (T)	95.69% (T)

learning techniques have also been proven appropriate to tackle this issue. In this work, we highlighted the shortcomings of the previous machine learning solutions, which consider only a few aspects of the underlying problem. Thus, this study is the first to provide a comprehensive (wide and in-depth) empirical analysis of SQL injection detection via machine learning. Furthermore, we cross-validated the trained models by using data from other distributions. This aspect is idle in the literature, even though the sensitivity of models to distribution change is crucial for any real-life deployment. Our work could be beneficial for security engineers and practitioners working with SQL.

ACKNOWLEDGEMENTS

- Support by the the European Union project RRF-2.3.1-21-2022-00004 within the framework of the Artificial Intelligence National Laboratory.
- Project no. 138903 has been implemented with the support provided by the Ministry of Innovation and Technology from the NRDI Fund, financed under the FK_21 funding scheme.

REFERENCES

- Alam, A., Tahreen, M., Alam, M. M., Mohammad, S. A., and Rana, S. (2021). Scamm: detection and prevention of sql injection attacks using a machine learning approach. Brac University.
- Betarte, G., Martínez, G., and Pardo, Á.(2018). Web application attacks detection using machine learning techniques. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE.
- Chen, D., Yan, Q., Wu, C., and Zhao, J. (2021). Sql injection attack detection and prevention techniques using deep learning. In *Journal of Physics: Conference Series*. IOP Publishing.
- Chen, Z., Guo, M., et al. (2018). Research on sql injection detection technology based on svm. In *MATEC web of conferences*. EDP Sciences.

- Farooq, U. (2021). Ensemble machine learning approaches for detection of sql injection attack. *Tehnički glasnik*.
- Gandhi, N., Patel, J., Sisodiya, R., Doshi, N., and Mishra, S. (2021). A cnn-bilstm based approach for detection of sql injection attacks. In *2021 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE)*. IEEE.
- Gogoi, B., Ahmed, T., and Dutta, A. (2021). Defending against sql injection attacks in web applications using machine learning and natural language processing. In *2021 IEEE 18th India Council International Conference (INDICON)*. IEEE.
- Hasan, M., Balbahaith, Z., and Tarique, M. (2019). Detection of sql injection attacks: A machine learning approach. In *2019 International Conference on Electrical and Computing Technologies and Applications (ICECTA)*. IEEE.
- Hosam, E., Hosny, H., Ashraf, W., and Kaseb, A. S. (2021). Sql injection detection using machine learning techniques. In *2021 8th International Conference on Soft Computing & Machine Intelligence (ISCMI)*. IEEE.
- Hu, J., Zhao, W., and Cui, Y. (2020). A survey on sql injection attacks, detection and prevention. In *Proceedings of the 2020 12th International Conference on Machine Learning and Computing*.
- Ingre, B., Yadav, A., and Soni, A. K. (2017). Decision tree based intrusion detection system for nsl-kdd dataset. In *International Conference on Information and Communication Technology for Intelligent Systems*. Springer.
- Jemal, I., Cheikhrouhou, O., Hamam, H., and Mahfoudhi, A. (2020). Sql injection attack detection and prevention techniques using machine learning. *International Journal of Applied Engineering Research*.
- Joshi, A. and Geetha, V. (2014). Sql injection detection using machine learning. In *2014 international conference on control, instrumentation, communication and computational technologies (ICCICCT)*. IEEE.
- Jothi, K., Pandey, N., Beriwal, P., Amarajan, A., et al. (2021). An efficient sql injection detection system using deep learning. In *2021 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE)*. IEEE.
- Kindy, D. A. and Pathan, A.-S. K. (2011). A survey on sql injection: Vulnerabilities, attacks, and prevention techniques. In *2011 IEEE 15th international symposium on consumer electronics (ISCE)*. IEEE.
- Krishnan, S. A., Sabu, A. N., Sajan, P. P., and Sreedeeep, A. (2021). Sql injection detection using machine learning. *REVISTA GEINTEC-GESTAO INOVACAO E TECNOLOGIAS*.
- Li, Q., Li, W., Wang, J., and Cheng, M. (2019a). A sql injection detection method based on adaptive deep forest. In *IEEE Access*. IEEE.
- Li, Q., Wang, F., Wang, J., and Li, W. (2019b). Lstm-based sql injection detection method for intelligent transportation system. In *IEEE Transactions on Vehicular Technology*. IEEE.
- Liu, M., Li, K., and Chen, T. (2020). Deepsql: Deep semantic learning for testing sql injection. In *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis*.
- Luo, A., Huang, W., and Fan, W. (2019). A cnn-based approach to the detection of sql injection attacks. In *IEEE/ACIS 18th International Conference on Computer and Information Science (ICIS)*. IEEE.
- Mishra, S. (2019). Sql injection detection using machine learning. Master's thesis, San José State University.
- Moosa, A. (2010). Artificial neural network based web application firewall for sql injection. *International Journal of Computer and Information Engineering*.
- OWASP. Owasp top 10: 2021. <https://owasp.org/Top10/>. Accessed: 2022-04-10.
- Pattewar, T., Patil, H., Patil, H., Patil, N., Taneja, M., and Wadile, T. (2019). Detection of sql injection using machine learning: a survey. *Int. Res. J. Eng. Technol. (IRJET)*.
- Pejo, B. and Kapui, N. (2023). Sqli detection with ml: A data-source perspective. *arXiv preprint arXiv:2304.12115*.
- Pham, B. A. and Subburaj, V. H. (2020). An experimental setup for detecting sql attacks using machine learning algorithms. In *Journal of The Colloquium for Information Systems Security Education*.
- Ross, K. (2018). Sql injection detection using machine learning techniques and multiple data sources. Master's thesis, San José State University.
- Sheykhkanloo, N. M. (2015). Sql-ids: evaluation of sqli attack detection and classification based on machine learning techniques. In *Proceedings of the 8th International Conference on Security of Information and Networks*.
- Tang, P., Qiu, W., Huang, Z., Lian, H., and Liu, G. (2020). Detection of sql injection based on artificial neural network. *Knowledge-Based Systems*.
- Tripathy, D., Gohil, R., and Halabi, T. (2020). Detecting sql injection attacks in cloud saas using machine learning. In *2020 IEEE 6th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*. IEEE.
- Uwagbole, S. O., Buchanan, W. J., and Fan, L. (2017a). Applied machine learning predictive analytics to sql injection attack detection and prevention. In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE.
- Uwagbole, S. O., Buchanan, W. J., and Fan, L. (2017b). An applied pattern-driven corpus to predictive analytics in mitigating sql injection attack. In *2017 Seventh International Conference on Emerging Security Technologies (EST)*. IEEE.
- Xie, X., Ren, C., Fu, Y., Xu, J., and Guo, J. (2019). Sql injection detection for web applications based on elastic-pooling cnn. In *IEEE Access*. IEEE.
- Yu, L., Luo, S., and Pan, L. (2019). Detecting sql injection attacks based on text analysis. In *3rd International Conference on Computer Engineering, Information Science & Application Technology (ICCIA 2019)*. Atlantis Press.