A Survey on SQL Injection Attack: Detection and Challenges

Zain Marashdeh
Faculty of Computer Studies
Arab Open University
Riyadh, Saudi Arabia
zainmarashdehm@gmail.com

Khaled Suwais
Faculty of Computer Studies
Arab Open University
Riyadh, Saudi Arabia
khaled.suwais@arabou.edu.sa

Mohammad Alia
Faculty of Sciences and Information Technology
Al-Zaytoonah University of Jordan
Amman, Jordan
dr.m.alia@zuj.edu.jo

2021 International Conference on Information Technology (ICIT) | 978-1-6654-2870-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICIT52682.2021.9491117

***Abstract*—Web applications are seen to play a vital role in the individual lives and even assist in the economic development of the country. However, many security vulnerabilities have been developed which can attack the web applications. One of the most common security threats which affect the web applications is the Structured Query Language Injection (SQL Injection or SQLi) attacks. The Open Web Application Security Project (OWASP) ranked them at the top out of the 10 most effective vulnerable attacks. This attack has been used for a long time for attacking different websites and webpages, due to which the victims had to suffer severe financial loss. In this study, the researchers investigated the different methods which could be used for detecting the SQLi attacks and their limitations. Additionally, they have also highlighted the direction of research being conducted in this area.**

***Index Terms*—SQL injection, static analysis, dynamic analysis, hybrid analysis, genetic algorithm, machine learning**

I. Introduction

The web usage and different internet-based applications have been extensively used because of the easy availability and accessibility of information. However, due to the increasing use of the internet, there is a rising concern about the security lapses that have occurred. Any person who uses the internet wants his data to be secured against any unauthorised usage. The web security issues allow hackers to extract user information, which can negatively affect the user's credentials.

SQL injection was seen to be a major security issue, worldwide. Due to the emergence of e-banking and cloud computing applications, this attack easily penetrates the various web applications [1]. The attacker makes use of specially designed inputs, regarded as database queries, for acquiring unauthorised access. Thereafter, he can execute the SQL database using an application. The web application developers strive to prevent these SQL injections by using one of the common source code analysis types; Static, Dynamic or Hybrid analysis [2]. However, Static analysis have the advantage to cover all source code, and the ability to detect all vulnerabilities among the source code. However, the false positive rate in the static analysis is high, which it makes the results not in that precise of Dynamic analysis [3]. Dynamic analysis could be run through the implementation of the program, where it detects the real vulnerability in source code without false positive results. However, Dynamic analysis can't detect all vulnerability in the source code due to the fact that it doesn't have the access to cover the whole source code. Hybrid analysis proposed as a combination between both; static and dynamic analysis. However, Hybrid analysis inherits the limitation from both approaches. In a similar study, Damodaran *et al.* [4] compared the different malware detection approaches based on their static, dynamic and hybrid analyses. Their results indicated that a hybrid approach was not as effective as a completely dynamic or static detection approach.

Therefore, the researchers started to look further new effective methods such as Genetic Algorithm (GA) and Machine Learning (ML) [5], [6], [7], [8], [9] in a way to reduce the static analysis limitations to detect SQLi and increase the accuracy to detect SQLi attack in web application. In this study, the researchers have summarised the various approaches which have been used for detecting the SQL injection attack by highlighting the advantages and limitations of each method to detect SQLi attack. Section II describes the problem and the types of SQLi attacks. Section III and IV describe the method and studies used to detect SQLi attack by highlighting their advantages and limitations. Section V presented the discussion from the evaluation of the previous approaches. Whereas Section VI presented the conclusion and some expectations from future work.

II. SQL Injection Attack

SQL Injection refers to a type of attack which attempts to acquire unauthorised access to the database after injecting code and investigating the SQL query [10]. The attackers use the SQL injections for investigating the database which was connected to a web application or a website. The data included in these databases must be thoroughly protected against the various SQL injections. If an unauthorised user can access the database, they can conduct unauthorised activities like retrieving information data, deleting tables or other harmful activities.

Fig. 1 presents one such example of SQL injection attack, where the user input received and stored in variables (*'fname'* and *'access'*), then the statements executed by a database server at Line 5. For such example of SQLi, they used a trick that involved a single quote and thereafter set the access field to (*'accessnumber' OR 0=0*). Due to the '*OR 0=0'* statement,

```
1  #Define POST variable
2  fname = request.POST['firstname'];
3  access = request.POST['accessnumber'];
4  #SQL query vulnerable to SQLi
5  db = "SELECT id FROM users WHERE firstname='"+ fname +
         "' AND accessnumber='" + access "'";
6  #Execute the SL statement
7  Execute(db);
```

Fig. 1. SQL Injection Attack Example.

the '*WHERE*' clause can return the first id from a user table irrespective of the first name and access number. Generally, the first user id within the database is departmental. The SQL Injection Attacks are classified into 3 different categories as Union based SQLi, error based SQLi or blind SQLi

A. Union Based SQL Injection

In an SQLi attack, the UNION operator combined 2 different SQL statements or queries. The Union-Based SQLI takes into consideration the advantages of both the designs and develops a database consisting of the desirable and intended results. This was possible after a different query was injected rather than a plain text, wherein a UNION keyword was inserted before a query.

B. Error Based SQL Injection

In the case of an Error-based SQLi, the operator adds an invalid input value in the query, which can trigger errors in the database. In this process, the database is forced to carry out some activities which lead to errors. The operator tends to look for some errors which were generated in the database and then uses them for collecting further information, thus exploiting the SQL queries, and manipulating the complete database.

C. Blind SQL Injection

A blind SQLi attack is a technique wherein the attacker poses some queries to a database and derives the answers. Thereafter, he plans his next course of action depending on the answers which were generated by the database. This was seen to be a difficult SQLi attack since the attacker does not have any prior knowledge regarding the database or the answers which would be generated. The attackers use this type of attack if the database generates some generic errors like the 'Syntax Error'. This blind SQLi attack is categorised into different types like the Boolean-Based or Time-Based SQLi attacks.

Thus, it was noted that the different SQLi attacks posed a serious threat to web applications. Many techniques were used for detecting these vulnerabilities. In this study, the researchers presented some of the techniques that could be used for detecting the SQLi vulnerabilities in web applications.

III. Methods and Techniques

A. Static Analysis

Static analysis is a process which investigates the source code of the web application for determining the vulnerabilities. As this technique analyses the codes, it can detect all the probably infected oaths present in the web application [11], [12]. Hence, this process can find the vulnerabilities affecting the program paths.

However, as the accurate identification of the vulnerabilities in the web applications with the help of a static analysis process is similar to the issue of halting, this technique shows a limitation of generating a higher false-positive rate [2]. False-positive results are those paths which are detected as a vulnerable path; however, they are not vulnerable. This technique shows another limitation, where it can be used for a specific language or framework. For example, a static analysis process which has been developed for the PHP framework is not applicable for the Ruby on Rails framework without carrying out a significant restructuring. Hence, this process was specifically developed depending on the framework or language features.

B. Dynamic Analysis

In direct contrast to the static analysis, the dynamic analysis tools do not investigate the source codes of different web applications. Instead, this technique interacts with the web applications similarly like that used by the user while interacting with the web browser.

Unlike the static analysis, the dynamic analysis process shows a lower false-positive rate [13]. However, the major limitation of this technique is that it cannot detect all the vulnerabilities existing in the web application. This was attributed to the fact that this tool only detects the vulnerabilities present in the program paths which it implements, while the static analysis tool observes all the program paths present in the web application.

Hence, one cannot fully understand the behaviour of the application if a dynamic analysis tool is used. The dynamic analysis tool can lead to a lower detection of the vulnerabilities in the application [3].

C. Hybrid Analysis

The term 'hybrid analysis' indicates a tool which can combine the static and the dynamic analytical tools. In step 1, it is used in combination with the static analysis tool for detecting all the probable vulnerabilities in the application. Thereafter, a confirmation step is carried out. In Step 2, the tool takes advantage of the vulnerabilities in the system. Thus, it reports the vulnerability only if Step 2 is completed.

The hybrid analysis process shows many advantages compared to the static analysis process. This tool can detect all vulnerabilities exiting in the different program paths. Furthermore, it shows a lower false-positive rate, as it combines the dynamic analysis tool. Dynamic analysis tools are known to verify the presence of the vulnerabilities, which decreases the false positive rate. However, the hybrid analysis tool also embraces the limitations displayed by the static analysis process. As a result, it can only be applied to one web language or framework. Hence, these hybrid tools are not as popular as dynamic or static tools alone [14].

D. Genetic Algorithm

A genetic algorithm refers to a search heuristic tool which can stimulate a natural selection process. The genetic algorithms are developed based on an evolutionary concept of natural selection or genetics. Hence, they indicate an intelligent manipulation of random search processes which are used for addressing the optimisation problems [15]. The following steps are involved in the genetic algorithm process:

1) Initial population: The encoding or representation of chromosomes in the genetic algorithm is carried out with the help of a binary format. The population used in the genetic algorithm is regarded as a set of likely solutions which can be used for resolving the issue.

2) Fitness function: This is described as the determination of the effectiveness of the chromosomes in addressing the problems. A higher fitness value indicates that the chromosome is very close in resolving the issue.

3) Selection: In this stage, the fittest chromosome is selected which can reproduce based on the specific selection process. The chromosome is selected based on its fitness value and is carried over in the subsequent generation.

4) Crossover and mutation: The offsprings, which are generated in the genetic algorithm are based on 2 genetic operators, i.e., crossover and mutation rate. Here, the crossover takes place after combining 2 chromosomes for producing a new solution which shows better traits. However, based on a particular mutation probability, the mutations occur when the chromosome values are altered. The primary population can be generated by determining the high-quality GA values of all individuals, whereas every individual in the population offers a solution for resolving the issue [16].

E. Machine Learning

Machine learning algorithms can be categorised into Supervised Learning and Unsupervised Learning algorithms [17]. The Supervised learning algorithms work as follows: They make use of a dataset, known as the training dataset, wherein every individual component is labelled. This model determines the relationship between the label and data and uses this information for classifying novel data which has not been noted before. This novel dataset is known as a test dataset [18]. The Supervised learning algorithms are further categorised into the Regression and Classification algorithms. Some of the examples of such algorithms are Bayesian Network, Decision Tree Induction, K-nearest neighbours, Neural Network and Support Vector Machine.

On the other hand, unsupervised learning is based on information theory and Bayesian principles [19]. In this type of algorithm, the machine attempts to detect the hidden structure in the unlabelled data. It makes no use of the output data or the prior labelled data. The Unsupervised algorithms are categorised into various categories like k-Means Clustering and Hierarchical Clustering Techniques [20]. Fig. 2 presents the various ML algorithms and categories.

As mentioned above, the supervised learning is referred to the algorithm consisting of input and output variables.

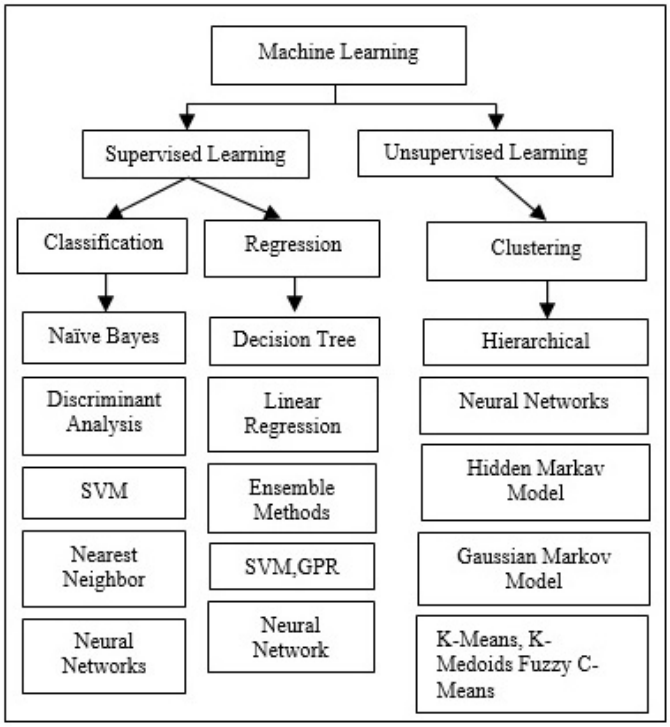

Fig. 2. Machine Learning Categories and Algorithms.

Thereafter, they use an appropriate algorithm for learning the mapping functions from input to output. The Unsupervised learning algorithm uses only the input data and not the conforming output variables. In the next section, the researchers have highlighted the various approaches which were used for detecting the SQLi vulnerability based on the discussed techniques and methods.

IV. DETECTION OF SQL INJECTION

Vulnerability detection is a process wherein the weakness present in the source code of the web application is detected. In this process, the vulnerabilities can be detected either before the use of the application or before an attacker detects the vulnerability after analysing the source codes. Table I depicts the literature studies conducted to detect SQLi vulnerability in web applications.

Fu *et al.* [21] developed and proposed the major concept related to SAFELI, i.e., a Static Analysis Framework for identifying the SIA vulnerabilities during compilation. The SAFELI statically inspects the MSIL bytecode of the ASP.NET web application as it uses the symbolic execution. The researchers used a hybrid constant solver at every hotspot which submitted an SQL query, for determining the corresponding user input which can cause a breach in information security.

Nagy and Cleve [22] further developed the static analysis tool by symbolically implementing the program. The researchers carried out a combined analysis of the database schema and complete database for identifying the code smells existing in the SQL queries. After implementing their prototype, they were able to target the JDBC as the common

TABLE I
STUDIES FOCUSED ON DETECTION SQLI VULNERABILITY

Author	Methods and Techniques					Advantages	Limitations
	SA	*DA*	*HA*	*GA*	*ML*		
Fu, *et al.* [21]	✓					Reduce the false positive results using Symbolic execution and hybrid constraints solver.	Manual which is not hard to be implemented on large web applications.
Nagy and Cleve [22]	✓					Help developers who struggle with complex SQL statements deeply buried in their source code.	Can't work with large queries, especially if it consists many join tables.
Khalid and Yousif [23]		✓				Propose an extraction algorithm to validate all user inputs from POST and Get methods.	Some user inputs exist in the source code, but it would not be implemented under any user inputs (infeasible paths).
Ali, *et al.* [24]		✓				Allows developers that are illiterate about hacking techniques in conducting penetration testing.	Can handle only the SQLi those passed through the URL (using GET method).
Shin, *et al.* [25]			✓			No false positive indicated in their results.	False negative generated due to the predefined attack patterns are not sufficient to detect all the possible attacks.
Prakash and Saravanan [26]			✓			Good detected results indicated with decreasing the time required for scan.	Inherits the limitation of false positive results due to the use of static analysis.
Choraś, *et al.* [5]				✓		Their correlation approach to SQLi allows to improve detection results.	False positive results which reduce their results accuracy.
Aziz, *et al.* [6]				✓		It requires very little interaction with testers, and it provides robustness and flexibility to deal with SQLIA attacks.	Not all inputs were flagged and checked, which make it unable to identify some attack patterns.
Resende and Drummond [7]				✓		An adaptive approach used to select features for profiling and parameters for an anomaly-based intrusion detection method.	Large number of selected features which increase the number of generations and increase the possibility to produce false positive.
Moh, *et al.* [8]					✓	Combined the advantages of pattern matching and supervised machine learning methods, which improved the detection accuracy.	Suitable for some types of SQLi.
Patel and Bhalamurugan [20]					✓	Unsupervised machine learning kmeans algorithm to detect future SQLi attack.	Not find enough attributes and features to cover all types of SQL injection attacks.
Ladole and Phalke [27]					✓	Detection of SQLi vulnerability and classify the users into normal users or attackers.	Not covered all types of SQLi attack.

database access technology, while MySQL was selected as the popular RDBMS. Their prototype tool could effectively serve as the proof concept and highlight the potential of their approach.

Khalid and Yousif [23] developed a novel SQLi detection tool which was based on the different HTTP requests sent by their users or clients. This tool included 2 stages, i.e., it initially collected the data from all web pages in 3 steps, (extraction of the URL addresses, collection of parameter values passed by the GET method, or collection of parameter values by the POST technique). Secondly, it detected the SQLi vulnerabilities after injecting the vulnerable SQL statements collected by the GET and POST methods.

Ali *et al.* [24] proposed a novel web scanning tool (MySQLInjector), which could carry out effective penetration tests on the PHP based websites, for detecting the hidden SQL vulnerabilities existing in the webserver databases. This tool combined the attacking patterns, vectors or models which helped the web developers, who were not very knowledgeable about the hacking methods. Thereafter, the developers could efficiently conduct penetration tests on their web database servers.

Shin *et al.* [25] designed a technique for identifying the input manipulation vulnerabilities by carrying out automated testing that was based on the dynamic and static analysis models. They applied the static analysis for tracing the flow of the user input values and for obtaining a concrete attack input to test the system. This tool can help in identifying the false-positive results generated by the static analysis tool.

In another study, Prakash and Saravanan [26] developed a mechanism which was based on the static and dynamic analysis tools. In this process, they introduced the SQLi Detection (SQLID) as the intermediate virtual database or layer between the database and application. Their results showed that this tool showed a good vulnerability detection ability, with a low false-positive rate. It required a lower scanning time.

Choraś *et al.* [5] proposed a new approach for SQLIA detection, which was based on GA, to detect the anomalous queries.

Thus, this technique exploited the GA, wherein all individuals in a population could investigate the log files generated by a SQL database. Every person aimed at delivering a generic rule (used as a regular expression), which could describe the logline. They noted that this technique could improve the detection ability of the SQLi vulnerability.

Aziz *et al.* [6] developed a search-based software testing tool for detecting the SQLi vulnerabilities in software applications. They used genetic programming for generating test datasets, which were used for testing the applications for SQLi based vulnerabilities. Furthermore, Resende and Drummond [7] used profiling for improving the GA-based system. They developed 2 anomaly-based approaches, coupled with GA, for selecting the optimal features and parameters for profiling. The 1st detection technique was based on profiling the variable distributions, while the 2nd technique generated a profile for centroids and cluster proportions.

Moh *et al.* [8] proposed a different system that was based on the 2-stage intrusion detection system. It made use of log analysis for protecting the web applications from further SQL injection attacks. They used the ML and pattern matching processes. Their experimental results indicated that this 2-stage system could detect a higher number of SQL injections compared to the single-stage system. The researchers concluded that when the Bayes Net model (a supervised ML process) preceded Kibana (a pattern-matching system), then the combined technique showed the best performance.

Patel and Bhalamurugan [20] proposed 2 different processes for detecting the SQL injection attacks. They made use of a pattern matching algorithm to detect the current SQL injection attacks and an unsupervised ML process. Their system determined and predicted the current and future SQL injection attacks. On the other hand, Ladole and Phalke [27] developed a system for detecting the SQLi attacks, by using the Fisher and an SVM Classification Score. They trained the vectors and stored them in an attribute-relation file format.

In another study, Ross *et al.* [28] collected the traffic from 2 different points, i.e., at a web application host and at the Datiphy appliance node, which was pre-sent between the web app host and the related MySQL database server. They further used a machine learning technique for analysing these datasets.

The literature review showed that static analysis could be used for improving the detection of SQLi attacks with a lower false-positive result if all programs were implemented symbolically [21]. The above-mentioned approaches were dependent on the constraint solver which decided if the path pattern was executed. However, the dynamic analysis presents an effective tool for detecting the SQLi, as presented in Table I. It was seen that the studies which used the dynamic analysis could not assess all the cases related to SQLi, which restricted the detection of the SQLi cases using the GET process [6]. On the other hand, the hybrid analytical process offered a solution for deriving the benefits of the static and dynamic processes [25], [26]. Despite these factors, the limitations of the static analysis process (i.e., a higher false-positive rate) were still noted and affected most of the results.

The GA technique was used as a heuristic search approach in the field of software testing for generating test cases using the web source code. Promising results were acquired when the researchers combined the earlier techniques with the GA [5], [6], [7], however, many iterations were needed for generating test cases. A false-positive result was still generated and a lot of time was wasted in carrying out longer iterations. Furthermore, Machine Learning (ML) offered a better solution for the automated testing method [8], [9], [27]–[29]. This technique helped in automating the detection of the SQLi vulnerabilities which increased the detection accuracy. The implemented technique was best suited for some forms of SQLi attacks; however, it could not cover all SQLi attacks. Hence, a lot more research needs to be carried out for resolving the limitations presented in the earlier studies for detecting the SQLi in the web applications.

V. Discussion

In this study, the researchers focused on the detection of the SQLi attacks as they were one of the major vulnerabilities which affected the web applications [30]. It is important to detect new solutions for securing web applications against SQLi attacks. The literature review indicated that the static, dynamic and hybrid analytical approaches could be used for detecting the vulnerabilities in the source codes of the web applications. Each of these techniques presents some limitations as shown in Table I, hence, a solution needs to be generated for resolving these issues. In this study [21], the researchers have proposed a new approach for decreasing the false positive rate noted in the static analysis results. This new technique proved effective in smaller datasets. The researchers could not test this technique in the actual web applications as the web applications are developed using thousands of lines of codes.

The development of this technique helped in solving the issue regarding the false positive rate in the static analysis results. Development of an automated tool offers a solution for the detection of different web application vulnerabilities before running web applications. The solution presented earlier [22] symbolically implemented the program for decreasing the false positive rate noted in the results. However, the constraint solver cannot solve all the different types of constraints, as some paths with a vulnerability could be missed.

On the other hand, the dynamic analysis generated better results while detecting the SQLi vulnerabilities with a low false-positive rate, however, it showed some limitations and could not cover all the different SQLi types of attacks. This was attributed to the fact that the dynamic analysis cannot access 100% of the code, unlike the static analysis. The hybrid approach which combined the static and dynamic analyses still displayed the limitations of the static analysis, i.e., a higher false-positive rate. Hence, one solution involved the improvement of the hybrid analysis techniques, implemented in the program symbolically, for decreasing the false positive rate.

GA was presented as an effective solution for generating test cases amongst the program paths for detecting the SQLi attack [5], [6], [7]. A few studies needed multiple iterations for detecting the vulnerabilities. These studies could be improved by decreasing the no. of attributes that were analysed and improving the fitness function responsible for reaching the solution. However, ML processes could be used for detecting the SQLi attacks owing to their low false positive rate and higher accuracy rate. Several datasets must be used, where the models are tested with multiple ML algorithms. This could prove to be a solution for the approaches which showed a high false-positive result [8], [9], [31], [32].

VI. Conclusion

The SQLi attacks are considered as a dangerous attack and hence were regarded as the most dangerous form of web application vulnerability. Many attackers used these attacks for attacking various websites and applications in the past few decades as they can significantly damage the websites. Some of the earlier studies used different techniques for detecting the SQLi attacks. Here, the researchers have presented a detailed literature review regarding the different SQLi vulnerabilities. They summarised different methods that could be used for detecting these SQL attacks. Additionally, they listed the advantages and limitations of each of the methods used for detecting the SQLi attacks. This study could be helpful to the readers who wanted to acquaint themselves with the topic or researchers who aim to determine all the issues which still affect the detection of SQLi attacks in the web applications.

Acknowledgment

The authors would like to thank Arab Open University, Saudi Arabia for supporting this study.

References

[1] S. Malik *et al.*, "User centric security models for improving the data security from sql injections and cross site scripting attacks," *Journal of the Gujarat Research Society*, vol. 21, no. 4, pp. 95–102, 2019.

[2] A. W. Marashdih, Z. F. Zaaba, K. Suwais, and N. A. Mohd, "Web application security: An investigation on static analysis with other algorithms to detect cross site scripting," *Procedia Computer Science*, vol. 161, pp. 1173–1181, 2019.

[3] V. Prokhorenko, K.-K. R. Choo, and H. Ashman, "Web application protection techniques: A taxonomy," *Journal of Network and Computer Applications*, vol. 60, pp. 95–112, 2016.

[4] A. Damodaran, F. Di Troia, C. A. Visaggio, T. H. Austin, and M. Stamp, "A comparison of static, dynamic, and hybrid analysis for malware detection," *Journal of Computer Virology and Hacking Techniques*, vol. 13, no. 1, pp. 1–12, 2017.

[5] M. Choraś, R. Kozik, D. Puchalski, and W. Hołubowicz, "Correlation approach for sql injection attacks detection," in *International Joint Conference CISIS'12-ICEUTE´ 12-SOCO´ 12 Special Sessions*. Springer, 2013, pp. 177–185.

[6] B. Aziz, M. Bader, and C. Hippolyte, "Search-based sql injection attacks testing using genetic programming," in *European Conference on Genetic Programming*. Springer, 2016, pp. 183–198.

[7] P. A. A. Resende and A. C. Drummond, "Adaptive anomaly-based intrusion detection system using genetic algorithm and profiling," *Security and Privacy*, vol. 1, no. 4, p. e36, 2018.

[8] M. Moh, S. Pininti, S. Doddapaneni, and T.-S. Moh, "Detecting web attacks using multi-stage log analysis," in *2016 IEEE 6th international conference on advanced computing (IACC)*. IEEE, 2016, pp. 733–738.

[9] R. Komiya, I. Paik, and M. Hisada, "Classification of malicious web code by machine learning," in *2011 3rd International Conference on Awareness Science and Technology (iCAST)*. IEEE, 2011, pp. 406–411.

[10] J. Abirami, R. Devakunchari, and C. Valliyammai, "A top web security vulnerability sql injection attack—survey," in *2015 Seventh International Conference on Advanced Computing (ICoAC)*. IEEE, 2015, pp. 1–9.

[11] Z. Zhuo, T. Cai, X. Zhang, and F. Lv, "Long short-term memory on abstract syntax tree for sql injection detection," *IET Software*.

[12] K. Filus, P. Boryszko, J. Domańska, M. Siavvas, and E. Gelenbe, "Efficient feature selection for static analysis vulnerability prediction," *Sensors*, vol. 21, no. 4, p. 1133, 2021.

[13] N. Doukas, P. Stavroulakis, and N. Bardis, "Review of artificial intelligence cyber threat assessment techniques for increased system survivability," in *Malware Analysis Using Artificial Intelligence and Deep Learning*. Springer, 2021, pp. 207–222.

[14] T. Fawcett, "An introduction to roc analysis," *Pattern recognition letters*, vol. 27, no. 8, pp. 861–874, 2006.

[15] F. Duchene, R. Groz, S. Rawat, and J.-L. Richier, "Xss vulnerability detection using model inference assisted evolutionary fuzzing," in *2012 IEEE Fifth International Conference on Software Testing, Verification and Validation*. IEEE, 2012, pp. 815–817.

[16] P. Gupta and S. K. Shinde, "Genetic algorithm technique used to detect intrusion detection," in *International Conference on Advances in Computing and Information Technology*. Springer, 2011, pp. 122–131.

[17] S. Miller and C. Busby-Earle, "The impact of different botnet flow feature subsets on prediction accuracy using supervised and unsupervised learning methods," *International Journal of Internet Technology and Secured Transactions*, vol. 5, no. 2, pp. 474–485, 2016.

[18] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, "Supervised machine learning: A review of classification techniques," *Emerging artificial intelligence applications in computer engineering*, vol. 160, no. 1, pp. 3–24, 2007.

[19] Z. Ghahramani, "Unsupervised learning," in *Summer School on Machine Learning*. Springer, 2003, pp. 72–112.

[20] M. Kaushik and B. Mathur, "Comparative study of k-means and hierarchical clustering techniques," *International journal of software and hardware research in engineering*, vol. 2, no. 6, pp. 93–98, 2014.

[21] X. Fu, X. Lu, B. Peltsverger, S. Chen, K. Qian, and L. Tao, "A static analysis framework for detecting sql injection vulnerabilities," in *31st Annual International Computer Software and Applications Conference (COMPSAC 2007)*, vol. 1. IEEE, 2007, pp. 87–96.

[22] C. Nagy and A. Cleve, "A static code smell detector for sql queries embedded in java code," in *2017 IEEE 17th International Working Conference on Source Code Analysis and Manipulation (SCAM)*. IEEE, 2017, pp. 147–152.

[23] A. Khalid and M. M. Yousif, "Dynamic analysis tool for detecting sql injection," *International Journal of Computer Science and Information Security (IJCSIS)*, vol. 14, no. 2, 2016.

[24] A. B. M. Ali, M. S. Abdullah, J. Alostad *et al.*, "Sql-injection vulnerability scanning tool for automatic creation of sql-injection attacks," *Procedia Computer Science*, vol. 3, pp. 453–458, 2011.

[25] Y. Shin, L. Williams, and T. Xie, "Sqlunitgen: Sql injection testing using static and dynamic analysis," in *The 17th IEEE International Symposium on Software Reliability Engineering (ISSRE 2006)*, 2006.

[26] J. Prakash and G. Saravanan, "Sqlid: Sql injection detection based on static and dynamic analysis," *Transylvanian Review*, no. 1, 2016.

[27] A. Ladole and M. Phalke, "Sql injection attack and user behavior detection by using query tree, fisher score and svm classification," *International Research Journal of Engineering and Technology*, vol. 3, no. 6, pp. 1505–1509, 2016.

[28] K. Ross, M. Moh, T.-S. Moh, and J. Yao, "Poster: Multi-source data analysis for sql injection detection," in *38th IEEE Symposium on Security and Privacy (IEEE S&P), San Jo-se, CA*, 2017.

[29] S. Almanasra, "Parallel algorithm for smoke image detection," *Int. J. Advance Soft Compu. Appl*, vol. 13, no. 1, 2021.

[30] OWASP, "Top-10 threats for web application security –2020," (accessed September 15, 2020), https://owasp.org/www-project-top-ten/.

[31] A. A. Hnaif, E. Kanan, and T. Kanan, "Sentiment analysis for arabic social media news polarity," *INTELLIGENT AUTOMATION AND SOFT COMPUTING*, vol. 28, no. 1, pp. 107–119, 2021.

[32] A. Hnaif, K. M. Jaber, M. A. Alia, and M. Daghbosheh, "Parallel scalable approximate matching algorithm for network intrusion detection systems." *Int. Arab J. Inf. Technol.*, vol. 18, no. 1, pp. 77–84, 2021.