# DETECTING AND PREVENTING SQL INJECTION

Mu'awuya Dalhatu*, Aliyu Sani Ahmad, Aliyu Umar, Anas Usman Inuwa and Babando Abdulrashid Aliyu
*Corresponding author: dalhatu@fuwukari.edu.ng

**Abstract**— Most programmers and system administrators depend on invasion detection and prevention tools and strategies to protect their systems against unauthorised access and other online database threats such as a SQL injection. However, the weaknesses, effectiveness and challenges of those tools, strategies or techniques are not really known. This leads to the wrong or poor implementation of some of the tools and strategies that compromises the security of many systems especially in developing countries like Nigeria. This research investigates some of the strategies and tools for detecting vulnerabilities such as a SQL injection and XSS. The research also investigates some of the solutions for preventing SQL injection attacks. The research shows that there is no single tool or strategy that is effective enough to detect or prevent all SQL injection. However, the research recommends the proper combination of two or more tools, strategies or techniques. The research also recommends testing or checking for vulnerabilities at the early stage of systems development.

**Index Terms**— Detecting vulnerability, SQL injection, cross site scripting, static analysis, Dynamic analysis, *and database security.*

## 1 INTRODUCTION

As the dependency on online databases and web applications increases, so also the security threats associate with them. Gollmann [1] identifies threats related to the web applications such as Structure Query Language (SQL) injection, Cross Site Scripting (XSS), Cookie poisoning and stealing, cross site request, JavaScript hijacking, and Domain Name System (DNS) rebinding. Similarly, OWASP [2] classifies the SQL as the most common threat, followed by XSS and password cracking as the third most threat.

## 2 DETECTING VULNERABILITY

Different researchers have proposed several approaches to detect SQL injection and other web applications vulnerabilities such as XSS. The detection of vulnerability is important because it helps developers to implement right defence mechanisms that prevent attacks. The detection can either be manual, automated or a combination of the two. Research on online database threats or security can be broadly divided into identifying vulnerabilities and preventing attacks [3]. It is, therefore, important to introduce the techniques of vulnerability detection before the techniques for preventions.

Gupta, Govil and Singh [4] identified static, dynamic and hybrid analysis approaches for detecting SQL, XSS and other vulnerabilities. Salas and Martins [5] proposed hybrid techniques that use a combination of static and dynamic approach for detecting XSS vulnerabilities. Others include secure programming, modelling in [6], defensive programming, web framework, and machine learning techniques [7].

Some approaches are being used for both detection and prevention, and the prevention techniques can also be based on static, dynamic or hybrid approach. These approaches are briefly explained below.

### 2.1 STATIC ANALYSIS

Filipiak and Sierra [8] stated that static analysis comprises of reviewing an application source codes in order to find vulnerabilities. Static analysis approach is a technique of locating potential vulnerabilities while a program is not executing. The technique is use to trace the cause of security challenges at the early stage of development or at any stage before the system is deployed [4].

Liu, Shi and Li [9] argue that static analysis approach is an effective way of assessing different part of source codes such as functions and variables in order to identify and eliminate weaknesses that will probably compromise a system security. Similarly, Basta and Zgola [10] state that static analysis approach is effective, inexpensive, requires fewer resources and the approach simply requires a tester to have a skill of identifying poorly validating inputs and written functions.

However, Liu, Shi and Li [9] argue that static analysis approach alone is not sufficient because it does not detect configuration and environmental vulnerabilities, it has false positive and false negative, and most of its process depend on human to manually verify the codes while the program is inactive. They further state that false negative exists because the approach is undecidable while false positive exists due to human involvement in verifying and confirming the result of the analysis.

Furthermore, Halfond, Viegas and Orso [11] argue that such approach is only effective in detecting an aspect of codes that are vulnerable to tautology technique of attack because most of the attacks use correct syntax and datatype which make it difficult to be detected with a static approach.

### 2.2 DYNAMIC ANALYSIS

Dynamic analysis approach is used to detect vulnerabilities in a program while being executed. Wang et al., [12] state that dynamic approach relies on testing data that alter the behaviour of the system during execution and it does not modify source code because it relies on the system implementation. According to Pérez, Filipiak and Sierra [8], one advantage of the dynamic approach is that it is less complicated since it does not require an in-depth understanding of source code or programming language because vulnerabilities can be found based on application's behaviour.

The dynamic approach has some limitations like static approach. Gupta, Govil and Singh [4] argue that result obtains from dynamic analysis cannot be generalised for future execution because it only detects part of the codes that execute during testing. Another issue is that executing all the possible data combination for testing is a challenging task.

### 2.3 HYBRID ANALYSIS

Hybrid analysis approach is used in different forms in order to compensate the weaknesses in static and dynamic approach and such approach is capable of producing an accurate result by improving static analysis' false alarm [4].

Liu and Xu [13] proposed a hybrid system based on a static and dynamic monitor. This approach uses syntax and lexical analysis to determine dynamic relative and static base number. Furthermore, Shar, Tan and Briand [14] proposed a hybrid system using data mining that is capable of detecting different SQL injection and XSS vulnerabilities. Their approach categorises unambiguous issues related to security purpose using static analysis while user-defined or built-in functions and strings are categorised using dynamic analysis.

Despite the challenges associated with static analysis approach, it seems to be more useful in dealing with small programs and provides non-professional programmers with simple and inexpensive means of eliminating common vulnerabilities.

## 3 TOOLS FOR DETECTING VULNERABILITY

There are several tools for detecting vulnerabilities such as SQL injection and XSS in online databases and web applications. Such tools are built based on static, dynamic or hybrid approach. The tools are categorised into three: commercial tools, open-source tools and academic or researchers tools [15].

Academic tools are proposed by researchers in order to solve a new problem, improve existing tools or identify a new method of designing other tools; such tools are not available for public use and they are mostly built based on a specific programming language such as Java and PHP [16], [30]. Such tools include enhanced MySQL injector vulnerability checker proposed in [30], a component-based SQL injection vulnerability detector tool proposed in [17], detecting SQL injection and XSS vulnerabilities through mining input sanitisation patterns proposed in [28].

Open-source tools are freely available to public without their methodology, algorithm or design procedures, such tools includes ZAP (Zed Attack Proxy), W3af (Web Application attack and audit framework), and Nikito [15], [17].

Furthermore, commercial tools provide another category of tools for detecting vulnerabilities. Aliero and Ghani [17] argue that commercial tools are entirely different from open-source tools because they are not free, users are not involved in their improvement, thus they can only utilise their functionalities. An example of commercial tools includes HP weinspect, IBM rational and Appscan [15]. Others include Netsparker, Acunetic and Bublast [17].

However, each of the categories has a unique advantage over the other two. For example, an academic category has the advantage of giving students and researchers opportunities to contribute to the existing research. The open-source category has the advantages of providing free and reliable tools to the public and also provides volunteers with opportunities to contribute to the development of the tools. Commercial tools category has the advantages of providing more reliable tools, more features or functionalities and support to their users.

## 4 PREVENTING SQL INJECTION

There has been a long recorded research to detect and prevent SQL injection attacks. This section reviews and compares some of the existing research on preventing SQL injection attacks in order to identify their effectiveness and weaknesses.

Deepa and Thilagam [18], and Bisht, Madhusudan and Venkatakrishnan [3] argue that approaches to prevent SQL injection attacks can be broadly divided into three categories: 1. By using secure programming practice to improve prevention mechanism 2. By using static analysis approach to detect vulnerabilities in the source codes and 3. By implementing prevention mechanisms for preventing attacks. Bellow are some of solutions or tools for preventing SQL injection attacks.

### 4.1 AMNESIA

Halfond and Orso [19] proposed a system that uses hybrid analysis based on a model approach for preventing SQL injection attacks. They developed a tool called AMNESIA that prevents malicious queries from execution. They stated that authorized queries are automatically generated by the static part of the model based on a program analysis while queries are dynamically generated by its dynamic part based on a runtime approach that is compared to the generated authorized queries. Moreover, Kumar and Peteriya [20] stated that queries generated dynamically are not executed if they mismatch the ones generated statically, and the process involved identifying hotspot, building query models and runtime monitoring. Additionally, a possible combination of all queries is generated for each hotspot and different hotspot has different possible combinations [21].

However, the dependency of the model on static analysis affects it effectiveness because static analysis may not be accurate in complex applications [25]. Additionally, partial understanding or predicting of inputs may prevent the execution of legitimate queries [3]. Another drawback with AMNESIA is that it increases CPU overhead due to query generations and comparisons [21].

### 4.2 CANDID

Bisht, Madhusudan and Venkatakrishnan [3] proposed a technique called CANDID for prevention of SQL injection attacks dynamically. The technique works by mining a structure of authorized query and match it against the issued query. The issued query is considered malicious and prevented from execution if it mismatches the structure of authorized queries.
CANDID has an advantage of eliminating manual modification of application to create prepared queries [22]. However, a study by Jang and Choi [22] shows that CANDID is mainly effective in preventing common and simple attacks such as piggy-backed, tautology and logically incorrect queries. Similarly, Lashkaripour and Bafghi [23] compared the effectiveness of different techniques and found out that CANDID was unable to prevent sophisticated attacks and it has a disadvantage of adding overhead due to the building of parse tree dynamically.

### 4.3 SQLrand

Boyd and Keromytis [24] proposed a technique that uses the concepts of randomized instruction set that was introduced in [29] for preventing different types of injection attacks including SQL injection and XSS. They built a tool called SQLrand that uses an intermediary proxy for converting randomized queries to normal queries [24]. According to Kumar and Paterial [20], the approach has the advantage of hiding error messages due to illegal queries. Additionally, Halfond and Orso [19] stated that the technique prevents different code injection attacks.
However, Jang and Choi [22] state that the approach is associated with several challenges: firstly, the security of the system can easily be compromised once the secret key is known. Secondly, computational overhead is high due to proxy server integration. Moreover, Bisht, Madhusudan and Venkatakrishnan [3] state that SQLrand modifies the semantic of a program by increasing the length of original keywords due the addition of secret key. Unfortunately, this will probably prevent authorized queries from execution.

### 4.4 WASP

Halfond, Orso and Manolios [25] proposed a SQL injection prevention technique based on positive tainting and syntax-aware evaluation. They built a tool called WASP (Web Application SQL-injection preventer) that was able to prevent different SQL attacks without generating false positives due to it highly automation. Tajpour and Shooshtari [26] state that the approach allows programmers to control the manipulation of string based on its syntax and source code due to the use of syntax-aware evaluation. According to Kumar and Pateriya [20], the approach is effective and has a simple deployment process and requirement. Additionally, tainting approach is effective in practice due to its low false positives, versatility and it is widely used by different researchers [3].

### 4.5 SQLProb

Liu et al., [27] proposed a proxy-based technique for preventing SQL injection attacks by building a model called SQLProb (SQL Proxy-based Blocker). They stated that the technique extracts user inputs, and analyse them by building a parse tree for validating the inputs. Inputs are not executed or sent to database if they are not syntactically the same with the compared queries [28].
SQLProb has the advantages of not modifying source code, being less complex compared to tainting, independent of programming language and metadata is not required for input validation [27]. Another advantage is in the technique's ability to store query and compute its similarity to an issued one and the dependency of its performance on the stored queries [22]. However, Jang and Choi [22] stated that the major drawback of the technique is it can only be used on MySQL database. The assumption of the technique that the test inputs collected or stored queries are adequate enough to identify all possible legitimate queries in a program might also be considered as a drawback [28]. Nevertheless, the technique has a major advantage of its ability to prevent different types of SQL injection attacks.

## 5 CONCLUSION

Several studies have investigate different methods for detecting SQL injection and XSS vulnerabilities. However, online database and web applications continue to experience such attacks. The literature review showed that detecting and preventing different SQL injection attacks require different methods, tools and strategies. Each method or strategy has a fault which suggests the use of two or more methods for a better result. This research investigates several methods and tools for detecting SQL vulnerabilities. The research also investigates some of the existing tools or solutions for preventing SQL injection attacks.

The research recommends the use of automated tools to prevent the SQL injection and XSS attacks because most attacks are also automated. This research also summarises the static, dynamic and hybrid analysis for detecting vulnerability at the early stage of development. Moreover, the research summarises some of the solutions for preventing SQL injection attacks.

## REFERENCES

[1] D. Gollmann, Securing Web applications. *Information Security Technical Report,* 2008 [online]**, 13**(1), pp. 1-9. [Accessed 8 June 2016]. Available at: <http://www.sciencedirect.com/science/article/pii/S1363412708000046>.

[2] OWASP (2014) Top 10 2013-Top 10 [Accesses 30 April 2016]. Available at: <https://www.owasp.org/index.php/Top_10_2013-Top_10>.

[3] P. Bisht, P. Madhusudan, and V. Venkatakrishnan, CANDID: Dynamic candidate evaluations for automatic prevention of SQL injection attacks. *ACM Transactions on Information and System Security (TISSEC)*, 2010. **13**(2), pp. 14.

[4] M. K. Gupta, M. C. Govil and G. Singh, Static analysis approaches to detect SQL injection and cross site scripting vulnerabilities in web applications: A survey *Recent Advances and Innovations in Engineering (ICRAIE)*, 2014. [online]. pp.1-5. [Accesses 30 April 2016]. Available at: < http://ieeexplore.ieee.org.ezproxy.wlv.ac.uk/Xplore/home.jsp>.

*[5]* M.I.P. Salas, and E. Martins, Security Testing Methodology for Vulnerabilities Detection of XSS in Web Services and WS-Security. *Electronic Notes in Theoretical Computer*, 2014.

[6] I. Hydara, A.B.M. Sultan, H. Zulzalil, and N. Admodisastro, Current state of research on cross-site scripting (*XSS*) – A systematic literature review. *Information and Software Technology* [online]**,** 2015. **58**pp. 170-186 [Accessed 15 June

2016]. Available at: < http://www.sciencedirect.com/science/article/pii/S0950584914001700>.

[7] K. Natarajan, and S. Subramani, Generation of Sql-injection Free Secure Algorithm to Detect and Prevent Sql-Injection Attacks. *Procedia Technology* [online], 2012. **4**pp. 790-796. [Accessed 4 June 2016] Available at: <http://www.sciencedirect.com/science/article/pii/S2212017312004082>.

[8] Pérez, P.M., Filipiak, J. and Sierra, J.M. (2011) LAPSE static analysis security software: Vulnerabilities detection in java EE applications. *Future Information Technology*. [Accessed 4 May 2016]. Available at: < https://scholar.google.co.uk/>.

[9] Liu, B., Shi, L., Cai, Z. and Li, M. (2012) Software Vulnerability Discovery Techniques: A Survey *2012 Fourth International Conference on Multimedia Information Networking and Security*. [online]. pp.152-156. [Accessed 26 June 2016]. Available at: <http://ieeexplore.ieee.org.ezproxy.wlv.ac.uk/Xplore/home.jsp>

[10] Basta, A., Zgola, M. and Bullaboy, D. (2012) *Database security*. Australia: Course Technology.

[11] Halfond, W. G., Viegas, J. and Orso, A. (2006) A classification of SQL-injection attacks and countermeasures *Proceedings of the IEEE International Symposium on Secure Software Engineering* [online]. IEEE, pp.13-15. [Accessed 4 June 2016]. Available at < https://scholar.google.co.uk/>

[12] Wang, Y., Wang, Zhao, D., W. and Liu, Y. (2015) Detecting SQL Vulnerability Attack Based on the Dynamic and Static Analysis Technology *Computer Software and Applications Conference (COMPSAC), 2015 IEEE 39th Annual*. [online]. pp.604-607.

[13] Liu, Z. and Xu, L. (2013) A Detective Tool against SQL Injection Attacks Based on Static Analysis and Dynamic Monitor *Web Information System and Application Conference (WISA), 2013 10th*. [online]. pp.195-198. [Accessed 29 May 2016]. Available at: <http://ieeexplore.ieee.org.ezproxy.wlv.ac.uk/Xplore/home.jsp>

[14] Shar, L. K., Tan, H. B. K. and Briand, L. C. (2013) Mining SQL injection and cross site scripting vulnerabilities using hybrid program analysis *2013 35th International Conference on Software Engineering (ICSE)*. [online]. pp.642-651.

[15] Djuric, Z. (2013) A black-box testing tool for detecting SQL injection vulnerabilities *Informatics and Applications (ICIA),2013 Second International Conference on.*. pp.216-221.

[16] Aliero, M., S., Ghani, I., Khan, M. M. and Nkima, L. H (2015) Analytical evaluation of sql injection protection tools. *Int'l Conf. on Science, Engineering and the Social Science* [online]. University Technology Malaysia, 11 – 13 May. . [Accessed 17 July 2016]. Available at: < https://scholar.google.co.uk/>.

[17] Aliero, M. S. and Ghani, I. (2015) A component based SQL injection vulnerability detection tool *2015 9th Malaysian Software Engineering Conference (MySEC)*. [online]. pp.224-229.

[18] Deepa, G. and Thilagam, P.S. (2016) Securing web applications from injection and logic vulnerabilities: Approaches and challenges. *Information and Software Technology* [online], **74**pp. 160-180. [Accessed 4 June 2016]. Available at: <http://www.sciencedirect.com/science/article/pii/S0950584916300234>.

[19] Halfond,W. G. and Orso,A. (2005) AMNESIA: analysis and monitoring for NEutralizing SQL-injection attacks *Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering*. ACM, pp.174-183.

[20] Kumar, P. and Pateriya, R. K. (2012) A survey on SQL injection attacks, detection and prevention techniques *Computing Communication & Networking Technologies (ICCCNT), 2012 Third International Conference on*. [online]. pp.1-5. [Accessed 4 Aug 2016]. Available at <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6396096>.

[21] Balasundaram, I. and Ramaraj, E. (2012) An Efficient Technique for Detection and Prevention of SQL Injection Attack using ASCII Based String Matching. *Procedia Engineering* [online], **30**pp. 183-190. [Accessed 4 June 2016]. [Accessed 10 June 2016] Available at :<http://www.sciencedirect.com/science/article/pii/S1877705812008600>.

[22] Jang, Y. and Choi, J. (2014) Detecting SQL injection attacks using query result size. *Computers & Security* [online], **44**pp. 104-118. [Accessed 20 June 2016]. Available at: <http://www.sciencedirect.com/science/article/pii/S0167404814000595>.

[23] Lashkaripour, Z. and Bafghi, A.G. (2013) A Simple and Fast Technique for Detection and Prevention of SQL Injection Attacks (SQLIAs). *International Journal of Security and Its Applications* [online], **7**(5), pp. 53-66.

[24] Boyd, S. W. and Keromytis,A. D. (2004) SQLrand: Preventing SQL injection attacks *International Conference on Applied Cryptography and Network Security*. [online]. Springer, pp.292-302.

[25] Halfond, W., Orso, A. and Manolios, P. (2008) WASP: Protecting Web applications using positive tainting and syntax-aware evaluation. *IEEE Transactions on Software Engineering*, **34**(1), pp. 65-81.

[26] Tajpour,A. and Shooshtari,M. J. (2010) Evaluation of sql injection detection and prevention techniques *Computational Intelligence, Communication Systems and Networks (CICSyN), 2010 Second International Conference on*. [online]. IEEE, pp.216-221.

[27] Liu,A., Yuan,Y., Wijesekera,D. and Stavrou,A. (2009) SQLProb: a proxy-based architecture towards preventing SQL injection attacks *Proceedings of the 2009 ACM symposium on Applied Computing*. ACM, pp.2054-2061.

[28] Shar, L.K. and Tan, H.B.K. (2013) Predicting SQL injection and cross site scripting vulnerabilities through mining input sanitization patterns. *Information and Software Technology* [online], **55**(10), pp. 1767-1780 Available at: <http://www.sciencedirect.com/science/article/pii/S0950584913000852>.

[29] Kc,G. S., Keromytis,A. D. and Prevelakis,V. (2003) Countering code-injection attacks with instruction-set randomization *Proceedings of the 10th ACM conference on Computer and communications security*. ACM, pp.272-280.

[30] Liban, A. and Hilles, S. M. S. (2014) Enhancing Mysql Injector vulnerability checker tool (Mysql Injector) using inference binary search algorithm for blind timing-based attack *Control and System Graduate Research Colloquium (ICSGRC), 2014 IEEE 5th*. [Online]. pp. 47-52.

[31] Dalhatu, M., Ahmad A. S. and Hambali M. A. (2019) Manual Testing of SQL Injection Vulnerabilities in an Online Student Database System. *Afr. J. Comp. & ICT* [online], **12**(1), pp. 51 – 66. Available at: <https://afrjcict.net>.

***Mu'awuya Dalhatu*** *is currently lecturing at the Department of Computer Science, Federal University Wukari, Taraba State, Nigeria. E-mail: dalhatu@fuwukari.edu.ng*

***Aliyu Sani Ahmad*** *is currently lecturing at the Department of Computer Science, Federal University Wukari, Wukari, Taraba State, Nigeria. E-mail: alally_ahmad@yahoo.com*

***Aliyu Umar*** *is currently lecturing at the Department of Mathematical Sciences, Taraba State University Jalingo, Taraba State, Nigeria. E-mail: aliyuumar6318@gmail.com*

**Anas Usman Inuwa** *is currently lecturing at the Department of Mathematical Sciences, Taraba State University Jalingo, Taraba State, Nigeria. E-mail:* compumatica11@gmail.com