Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 10, October-2021

Detection and Prevention of web application from SQLI and XSS Attack

Shegaw Demessie Bogale
Oromia State University
Information Technology
Batu , Ethiopia

Hailemichael Kefie Tamiru
Wolkite University
Software Engineering
Wolkite ,Ethiopia

***Abstract*: In the global age of information web applications are the backbone of the technology in which a world becomes one village. In this day it is the era of computers which interconnects various business organizations which uses the internet for their financial transactions, educational endeavors, and countless other activities. Design Science Research Methodology and A rigorous survey have been conducted and consequently, comparative analysis of various detection and prevention techniques is done with respect to various types of attacks.in current research various Hashing algorithm for the detection and prevention of SQLIA are analyzed and few tested. From the survey of various papers, it is found that the SQL Injection and Cross-site Scripting (XSS) attacks are most powerful in today's web application. Hence, the big challenge became to secure such a website against attack via the Internet. The issue of SQLIA still exists and has become a giant online threat to many companies who became a victim of SQLIA vulnerability and cost them a lot of money. The main purpose of this study was designing improved hashing algorithm for Detecting and Preventing SQLIA & XSS approach has been implemented successfully and fully able to fix SQLIA and XSS vulnerabilities. SHA512 hashing algorithm is used to build a good, secure cryptographic hash function by developing a good compression function in which each input bit affects as many output bits as possible. The results obtained by the implementation and evaluation are measured in number of test cases that produce more robust and reliable SQLIA and XSS prevention mechanism.**

Keywords: Web Application Security, SQL Injection, SQLIA, XSS, Vulnerabilities, Hashing Technique

I. INTRODUCTION

The Internet and web applications are the modern day workplace and business ground contributing to driving the world economy. At the same time, hackers and attackers have developed a parallel underground economy of hacking into web applications and stealing a large amount of sensitive business-critical information with malicious intent. Among the various security threats a web application is exposed to, SQL Injection Attack (SQLIA) and cross-site scripting (XSS) has taken the forefront. It has prevailed as a popular attack method. Using a SQL injection attack, an attacker can extract, modify or destroy the back end database of web applications. The simplicity of attacking a web application using SQL injection and abundance of vulnerable applications on the Internet has largely contributed to widespread data breach incidents [1]. As of December 2015, over one billion web sites have been hosted on the Internet, and nearly half the world's population have become users of various Internet services [1, 2]. The popularity of the internet has made the use of web applications ubiquitous and essential to the daily lives of people, businesses and governments. Web servers and web applications are used to handle tasks and data that can be critical and highly valuable, making them a very attractive target for attackers in every transaction. Web application attackers mostly use browsers as attacking tool. Web applications are prone to all major vulnerabilities. Among all, SQL Injections and XSS are the most popular and frequently occurred attacks. The OWASP Top 10 document [1,2,3] is commonly referenced when mentioning the most common web application security flaws. The document describes the 10 most common flaws based on data collected from hundreds of organizations and over 50,000 applications and APIs. According to OWASP and Web Application Security Consortium (WASC) 2010 statistics report SQLI and XSS attacks are the most common web application attacks happened due the existence of Vulnerabilities. Every web app developer should consider web vulnerabilities as much as possible with full effort in order to avoid SQLI and XSS vulnerabilities. In this research, Very mature security values and models will be used as a base for evaluating the class of vulnerability of web apps in organizations.

II. LITERATURE REVIEW

The first ever public disclosure of SQL injection attack vulnerability was documented by Jeff Forristal [4] under the alias rain.forest.puppy (RFP) in a hacker ezine named Phrack Magazine [5] released on December 25, 1998. The relative prevalence of some of these attacks is shown in the figure below. This data was obtained from HACKING & TRICKS, a blog about Hacking &Computer Security by Nirav Desai, in an entry from January 8th, 2013 [10]. Clearly, SQL Injection comprises a significant fraction (approximately 25%) of all web-based attacks.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 10, October-2021

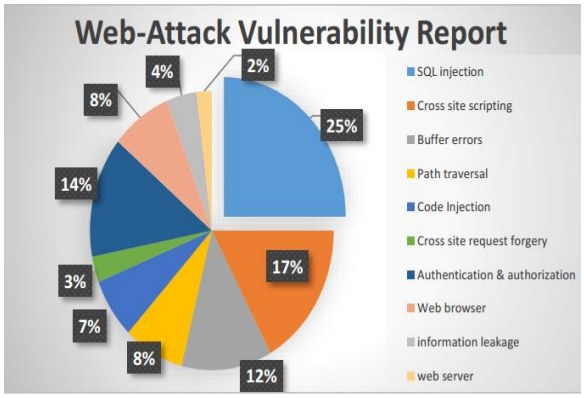

Figure Web attack vulnerability Report [10]

We understand from the following figure researches are conducted in every year and the problem is very severity in web technology. The National Vulnerability Database ('NVD') is a database managed by the National Institute of Standards and Technology ('NIST)'. It is a database containing vulnerabilities that are based on the Common Vulnerabilities and Exposures ('CVE') dictionary[22,23] this shows that That number has fortunately dropped over the last few years. However, vulnerabilities continuously get reported and the problem persists even with newly developed applications.

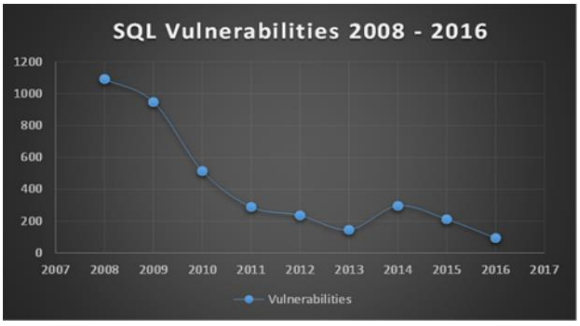

Figure.NIST SQLI Vulnerability Statistics [**23]**

A. Web application and Vulnerabilities

Most of the current web applications use RDBMS (Relational Database Management Systems). Sensitive information like credit card, social security, and financial records are stored in these databases.

Usually, programmers who write these programs are unaware of a technique for writing secure code. They would focus on implementing desired functionalities and would focus less on security aspects. They would focus on implementing desired functionalities and would focus less on security aspects. This results in vulnerabilities in web applications. Vulnerabilities allow an attacker to target this web application and obtain valuable information. Attackers would send SQL (Structured Query language) to interact with RDBMS servers or modify existing SQL to retrieve unauthorized information without any authentication. There is a large amount of literature available covering web application vulnerabilities and associated common attacks [6] [7] [8]. Although the types of vulnerabilities and attacks are known since a long time, there is no single solution to mitigate all of them and the limited security support offered by web application frameworks and the popularity and growing complexity of web applications have made them more prone to attacks than ever [9]. Common web application vulnerabilities can be classified into three types [5] Injection Vulnerabilities, Business Logic Vulnerabilities, and Session Management Vulnerabilities.

B. SQL Injection Attack

SQL injection attacks have many different forms and functions depending on the sources, goals, SQL specific techniques, and platform configurations. The earliest formal classification of SQL injection attacks was presented by Halford et al. [11] which was based on the injection mechanism and attack intent. Following their footprints, Sun et al. [12] proposed a more elaborate model of SQL injection attacks considering assets, threats, and countermeasures in an attempt to establish a semantic relationship between the different classes of attacks. The following sections discuss different types of SQL injection attacks following a classification that is universally accepted by the domain researchers.

Tautological Attacks: In logic, a tautology is a formula that evaluates to true in every possible interpretation. The main goal of a tautological SQL injection attack is to change the conditional in the WHERE clause of the dynamic SQL query so that it always evaluates to true irrespective of the original condition given by the programmer.

UNION based Attacks: In SQL, the UNION keyword is used to combine the result from multiple SELECT statements into a single result set. In UNION-based SQL injection attacks, the attacker injects an additional query using the UNION keyword to bring data from other tables or columns into the result set so that they can be displayed on the web page. For a UNION query to be valid, both queries must have the same number of columns of the same data types. This information is previously collected by the attacker through error messages from illegal or incorrect query attacks.

Blind Injection Attacks: When the database error messages are suppressed, the attacker cannot gather the information needed for crafting the subsequent injection attacks. In other words, the attacker is blind in absence of any feedback from the server through error messages. In such a situation, the attacker injects SQL commands and observes if and when there is a change in response from the web page. By carefully relating the responses to the injection vectors, i.e., when the behavior remains the same and when it changes, the attacker can make inferences about vulnerable parameters and additional information about the database structure.

Therefore, blind injection attacks are also known as inference-based attacks. Since blind injection attacks rely on the correlation between the attack vector and behavior or response of the web page, it is much slower than UNION-based attacks. Nevertheless, it is possible to achieve an extraction rate of about 1 byte per second by issuing multiple requests to the web application in parallel [8].

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 10, October-2021

Boolean-based Blind Attacks: In Boolean-based blind injection, the attacker crafts the attack vectors to ask the server true/false questions. When the answer is true, the web page is displayed normally, but when the answer is false, the display of the page changes significantly. By observing the behavior of the web page the attacker first infers about the vulnerable parameters and then crafts the next set of true/false questions to extract the database structure byte by byte.

Time-based Blind Attacks: Boolean-based blind injection attacks require a large number of requests to be submitted to the web server because the data must be extracted one byte at a time. In some cases, production web servers are configured to limit the maximum number of requests per minute from the same source. Even when such a restriction is not imposed, too many requests from one source within a small interval of time may be noticed by a proactive server administrator during auditing of web server logs, and the administrator may blacklist the IP address of the attacker.

Stored Procedure Attacks: A stored procedure is a set of SQL queries with an assigned name that's stored in compiled form in the database server. Stored procedures separate complex business logic from the application code, generally take parameters, perform SQL queries according to the business logic, and return the results to the web application. Many programmers believe that by moving the database queries into stored procedures, the web application would become resistant to SQL injection attacks, however, this is a misconception. Stored procedures can be equally vulnerable to SQL injection attack as the web application, particularly when the queries are dynamically constructed inside the procedure using string (VARCHAR) type of parameter(s) passed to it. Attackers target stored procedures for privilege escalation, buffer overflows and gaining access to the operating system. The attack methodologies are exactly same, except that the victim is a stored procedure instead of a query on a web page.

C. Cross-Site Scripting

Cross-site scripting is a prominent threat in web-based application, caused through a malicious input to the application. Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user. Flaws that allow these attacks to succeed are quite

Widespread and occur anywhere a web application uses input from a user within the output it generates without validating or encoding it. XSS is a new common vulnerability which can let hackers inject the code into the output application of web page which will be sent to a visitor's web browser and then, the code which was injected will execute automatically or steal the sensitive information from the visits input.

D. Related Work

The study follows the guidelines for conducting Systematic Literature Review (SLR) by Kitchenham and Charters [17] and Kitchenham et al. [18], which are widely used in the Software Engineering domain. The guidelines are also applicable in general to other areas of Computer Science and Engineering. The SQL and XSS injection attack problem has long gained the attention of the research community and significant work on detection and prevention of SQL injection attacks has been done in over a decade. However, very few comprehensive literature surveys have been conducted. We have conducted a systematic literature survey of the qualitative research works on prevention and detection of SQL injection attacks published since 2002 up to late 2017. [13] "Preventing SQL Injection Attacks in Stored Procedures", they also provided a novel approach to shield the stored procedures from attack and detect SQL injection from sit (Ke Wei, M. Muthuprasanna, Suraj Kothari, 2007). This method combines runtime check with static application code analysis so that they can eliminate vulnerability to attack. The key behind this attack is that it alters the structure of the original SQL statement and identifies the SQL injection attack. [14] "Using Parse Tree Validation to Prevent SQL Injection Attacks" ACM, we covered the techniques for SQL injection discovery. This paper also covered very well the SQL parse tree validation. [15] "The Essence of Command Injection Attacks in Web Applications" ACM, they covered the techniques to check and sanitize input query using SQLCHECK, it uses the augmented queries and SQLCHECK grammar to validate query. 16] "Using Automated Fix Generation to Secure SQL Statements" IEEE CNF, they covered brief background, SQL statement, and vulnerability replacement methods [17] "Automated Protection of PHP Applications against SQL-injection Attacks" they covered an original method to protect application automatically from SQL injection attacks. The original approach combines static analysis, dynamic analysis, and automatic code re-engineering to secure existing properties. Certain works provide client side solutions [23, 25] for Cross -Site Scripting attacks where a change in the browser code or a fire wall set up is recommended .Noxes is a client side solution for XSS attacks. It creates a personal firewall on the client side and checks every outgoing request and incoming response. It uses both manual and automatically generated rules to prevent XSS attacks on the client side. Nonce spaces set rules and in each document it randomizes the XML namespace prefix. These prefixes are identifiable by the user and the user will be able to identify the trusted content by the web application and untrusted content provided by the attacker. Proposed client side solution. It was a Mutation event transform system which defined policies to be enforced in the client side browser. It defined security policies based on monitoring client behavior.

III. METHODOLOGY

Design Science Research Methodology (DSRM) approach has been selected. We have preferred this methodology by three objectives it is consistent with prior literature, it provides a nominal process model for doing DS research, and it provides a mental model for presenting and evaluating DS research in security area. Moreover, DSRM propose the creation and evaluation of IT artifacts that may

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 10, October-2021

include constructs, models, methods, and instantiations [20], and therefore, it is reasonable to use DSRM as the proposed artifact is a model which is intended to provide guidance on how to prevent SQLI and XSS injection attack in web applications performed objectively.

A. Inclusion and exclusion criteria

The web application vulnerability prevention and detection identified from the literature review. **Inclusion criteria:** research work within the focus area of preventing and detecting injection attacks in a web application. The research work must address the web vulnerabilities classification and selecting the most serious web injection attacks. We also mention the classification of detection techniques of the web application.

Exclusion criteria: non- English language topics of similar words with security but not in web security concept (for instance hardware concept), and sources in the form of a tutorial, book review and presented slide are excluded.

B. Searching strategy

The searching strategy started by systematically reading and scanning journals, books, conference papers, articles, published about the coding flaws that are the reasons for the attacks on web applications. The digital database containing published papers such as ACM, IEEE, Science Direct, Springer, Elsevier and Google Scholar were used to search desired articles. Having the inclusion and exclusion criteria into account, the first phase starts by applying search string rule to database sources (IEE Explore, Science Direct, spring, Elsevier, Google scholar). Second, title based selection will be performed and checked against inclusion and exclusion criteria thereby maintain studies that are related to this work and include those articles into the third phase.

C. Searching

The searching strategy started by systematically reading and scanning journals, books, conference papers, articles, published about the coding flaws that are the reason for the attacks on web applications. The digital database containing published papers such as ACM, IEEE, Science Direct, Springer, Elsevier, and Google Scholar were used to search desired articles. We accessed those articles by using the Malmö University digital library. Internet resources were also frequently used to search for articles about programming flaws. We found the published articles in between the year 2007-2018.

D. Obtaining and Assessing

The obtaining and assessing information about SQLi coding flaws comes from journal papers, articles, conference and as well as white papers. We obtained that information through Malmö university digital library which allows us to get information freely. We also obtained information by using the Google search engine and assessing the source by using the Google Scholar website. The search term that we used to obtain information related to our topic were Cyber Security, Top vulnerabilities, SQLi, SQLi attack, SQLi coding flaws, Web application attacks, Code Injection cyber-attacks.

E. Critical Evaluation

To investigate our research questions, we first read the articles that were related to our research topic. Then, we assessed articles which provided information about SQLi and XSS such as coding flaws, detection, and prevention. Most of the articles provided more theoretical baseline information and technical overview of SQL and XSS attacks rather than practical - how we can prevent these attacks. Further, it was checked if the paper coming from the journal or conference or from another reliable source. Authenticity and validity of every considered source have been checked.

F. The algorithm used in this study

Secure hash algorithm: Secure Hash Algorithm (SHA) was developed by NIST along with NSA [21] in 1993; SHA was published as a FIPS.The SHA is called secure because it is designed to be computationally infeasible to find two different messages which produce the same message digest. Any change to a message in transit will result in a different message digest, and the signature will fail to verify. Secure Hash Algorithm (SHA) is necessary to ensure the security of the Digital Signature Algorithm (DSA).It takes a message of any length <264 bits as input and produces a 160-bit message digest as output. The message digest is then inputted to the DSA, which computes the signature for the message. Signing the message digest rather than the message often improves the efficiency of the process, because the message digest is usually much smaller than the message.

SHA **variations:** They differ in the word size. SHA-256 uses 32-bit words where SHA-512 uses 64-bit words. There are also truncated versions of each standard, known as SHA-224, SHA-384, SHA-512/224, and SHA-512/256.SHA-512 is roughly 50% faster than SHA-224 and SHA-256 on 64-bit machines, even if its digest is longer. The speed-up is due to the internal computation being performed with 64-bit words, whereas the other two hash functions employ 32-bit words. Because of this reason we have used the SHA-512 hashing algorithm on this study

IV. CONCEPTUAL MODEL

The SHA512 i.e. Secure Hash Algorithm is based on the concept of hash function. The basic idea of a hash function is that it takes a variable length message as input and produces a fixed length message as output which can also be called as hash or message-digest. The technique behind building a good, secure cryptographic hash function is to devise a good compression function in which each input bit affects as many output bits as possible. It is used with the Digital Signature Standard (DSA) for digital signature so it has particular importance [23]. A hash function takes a variable length message as an input and as an output produces a fixed length message which can also be called hash or message digests. SHA512 has a message size of 2^128 bits and a message digest of 512 bits. SHA512 is designed so that it is practically infeasible to find the output of the two input messages to be the same. It is also impossible to get back the input message from the obtained message digest [25].

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 10, October-2021

A. Requirements of Prevention and Detection System

To build an SQLI-A prevention and Detection system, a certain requirement must be fulfilled to make the system efficient. Before describing those requirements, it is necessary to introduce four important terms that clearly define the requirements:

False Positive (FP): represents the pattern which is classified as SQLI by the prevention and detection system but they are normal.

True Positive (TP): represents a pattern which is classified as SQLI by the prevention and detection system and they are truly SQLI.

False Negative (FN): represents the pattern that is classified by the prevention and detection system as being legitimate when in fact they are SQLI

True Negative (TN): represents the number of instances that are classified by the prevention and detection system as being legitimate and that really are truly legitimate.

V. CONCLUSION AND FUTURE WORK

This study implements a technique to prevent and detect SQLIA and XSS. The implementation used hashing technique which is computationally light. The proposed technique effectively prevents and detects SQLIA and XSS attack without much over head on the application. In this work parameterized queries are used to avoid blind SQLIAs. The hashing technique and the parameterized queries have been tested against 65 test cases, here the application behaved as predicted against all these test cases that showed we have meet our objectives and the developed framework is capable of prevent and detect SQLIA and XSS attack. This paper analyzed the problems that current Web Vulnerability Scanners are facing when trying to detect XSS vulnerabilities, as reported in recent research it was found that the vulnerability scanners are a promising mechanism to fight the XSS vulnerabilities in web applications. In the implementation of the Cross site script attack prevention mechanism, every HTTP request and response is fetched through servlet filter and it is analyzed to check for the presence of any malicious injected script. If any additional injected script is presented in the HTTP response, it might be the suspicious script that leads to the XSS attack. We used some performance metrics to measure the performance of the system which include accuracy, false positive rate, and false negative rate.

REFERENCES

[1] OWASP Foundation, "OWASP Top-10 2013," [Online].Available:https://www.owasp.org/images/f/f8/OWASP_Top_10_-_2013.pdf.

[2] OWASP Foundation, "OWASP Top-10 2017 (Release Candidate 1)," [Online]. Available:https://github.com/OWASP/Top10/raw/master/2017/OWASP%20Top%2010%20-%202017%20RC1-English.pdf.

[3] Telstra Cyber Security Report "Managing risk in a digital world" technical report 2017

[4] S. M. Kerner, "How was SQL Injection Discovered?" [Online]. Available: http://www.esecurityplanet.com/network-security/how-was-sqlinjection-discovered.html

[5] R. F. Puppy, "NT Web Technology Vulnerabilities," Phrack Magazine, vol. 8, no. 54. Phrack.org, [Online]. Available: http://phrack.org/issues/54/ 8.html

[6] D. Litchfield, "Remote Web Application Disassembly with ODBC Error Messages," Blackhat Asia, 1,[Online]. Available: www.blackhat.com/ presentations/bh-asia-01/litchfield/litchfield.doc

[7] C. Anley, "Advanced SQL Injection in SQL Server Applications," NGSSoftware Insight Security Research (NISR), Next Generation Security Software Ltd, Online].Available:http://www.cgisecurity.com/lib/advanced_sql_injection.pdf

[8] C. Anley, "(more) Advanced SQL Injection in SQL Server Applications (White Paper)," NGSSoftware Insight Security Research (NISR), Next Generation Security Software Ltd, [Online]. Available: http://www.cgisecurity.com/lib/more_advanced_sql_injection.pdf

[9] TrustWave, "Trustwave 2011 Global Security Report," [Online]. Available: https://www.trustwave.com/Resources/Library/Documents/2011-Trustwave-Global-Security-Report/?dl=1

[10] Nirav Desai "HACKING&TRICK" a Blog about Hacking & Computer Security https://tipstrickshack.blogspot.com/2013/01/list-of-vulnerability-its-tutorial.html

[11] G. Ollmann, "Second-order Code Injection Attacks," NGSSoftware Insight Security Research (NISR), Next Generation Security Software Ltd, [Online]. Available:https://packetstormsecurity.com/files/download/34919/SecondOrderCodeInjection.pdf

[12] Amit Klein "DOM Based Cross Site Scripting or XSS of the Third Kind" (WASC writeup), July 2015.

[13] Friedl's Steve Unixwiz.net Tech Tips. (2007). "SQL Injection Attacks by Example". Retrieved November 1, 2007, from http://www.unixwiz.net/techtips/sql-injection.html

[14] Wei, K., Muthuprasanna, M., & Suraj Kothari. (2006, April 18). "Preventing SQL injection attacks in stored procedures". Software Engineering IEEE Conference. Retrieved November 2, 2007, from http://ieeexplore.ieee.org.

[15] Zhendong Su, Gary Wassermann. University of California, Davis. "The Essence of Command Injection Attacks in Web Applications." Retrieved January 11, 2009, from http://portal.acm.org

[16] Merlo, Ettore, Letarte, Dominic, Antoniol & Giuliano."Automated Protection of PHP Applications against SQL-injection Attacks." Software Maintenance and Reengineering, 11th European Conference IEEE CNF.

[17] B. Kitchenham and S. Charters, "Guidelines for Performing Systematic Literature Reviews in Software Engineering," in EBSE Technical Report, Ver. 2.3. Keele University and the University of Durham, 2007.

[18] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, "Systematic Literature Reviews in Software Engineering–A Systematic Literature Review," Information and Software Technology, vol. 51, no. 1, pp. 7–15. Elsevier, 2009.

[19] K. Peffers, T. Tuunanen, M. A. Rothenberger and S. Chatterjee, "A design science research methodology for information systems research," Journal of management information systems, vol. 24, pp. 45-77, 2007.

[20] A. Hevner and S. Chatterjee, "Design science research in information systems," in Design research in information systems, Springer, 2010, pp. 9-22.

[21] Oates, B.J, (2006). "Reviewing the literature" - Researching Information Systems and Computing. London, England: SAGE Publications Ltd.

[22] Backman, Lars. "Why is security still an issue? A study comparing developers' software security awareness to existing vulnerabilities in software applications." (2018).

[23] Bissyandé, Tegawendé F., et al. "Vulnerabilities of Government Websites in a Developing Country–The Case of Burkina Faso." International Conference on e-Infrastructure and e-Services for Developing Countries. Springer, Cham, 2015.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 10, October-2021

[24] B.SHAH, C., and PANCHAL, D. R. "Secured hash algorithm-1": review paper. In International Journal for advance research in engineering and technology (Oct 2014), pp.1–6.

[25] Temeiza, Q., Temeiza, M., and Itmazi, J. "A novel method for preventing SQL injection using sha-1 algorithm and syntax-awareness." In 2017 Joint International Conference on Information and Communication Technologies for Education and Training and International Conference on Computing in Arabic (ICCA-TICET) (Aug 2017), pp. 1–4