A hybrid framework for detecting structured query language injection attacks in web-based applications

Md. Hasan Furhad[1], Ripon K. Chakrabortty[2], Michael J. Ryan[3], Jia Uddin[4], Iqbal H. Sarker[5]
[1]Deloitte Cyber Risk Advisory, Canberra, Australia
[2]School of Engineering and Information Technology, University of New South Wales Canberra, Canberra, Australia
[3]Capability Associates, Canberra, Australia
[4]AI and Big Data Department, Endicott College, Woosong University, Daejeon, South Korea
[5]Department of Computer Science and Engineering, Chittagong University of Engineering and Technology, Chittagong, Bangladesh

Article Info

Article history:

Received Jun 23, 2021
Revised May 24, 2022
Accepted Jun 10, 2022

Keywords:

Cyber attacks
SQL injection: blockchain
SQL query matching
Vulnerability detection
Web-based framework

ABSTRACT

Almost every web-based application is managed and operated through a number of websites, each of which is vulnerable to cyber-attacks that are mounted across the same networks used by the applications, with much less risk to the attacker than physical attacks. Such web-based attacks make use of a range of modern techniques-such as structured query language injection (SQLi), cross-site scripting, and data tampering-to achieve their aims. Among them, SQLi is the most popular and vulnerable attack, which can be performed in one of two ways; either by an outsider of an organization (known as the outside attacker) or by an insider with a good knowledge of the system with proper administrative rights (known as the inside attacker). An inside attacker, in contrast to an outsider, can take down the system easily and pose a significant challenge to any organization, and therefore needs to be identified in advance to mitigate the possible consequences. Blockchain-based technique is an efficient approach to detect and mitigate SQLi attacks and is widely used these days. Thus, in this study, a hybrid method is proposed that combines a SQL query matching technique (SQLMT) and a standard blockchain framework to detect SQLi attacks created by insiders. The results obtained by the proposed hybrid method through computational experiments are further validated using standard web validation tools.

Corresponding Author:

Jia Uddin
AI and Big Data Department, Endicott College, Woosong University
Daejeon, South Korea
Email: jia.uddin@wsu.ac.kr

1. INTRODUCTION

Web-based applications play a significant role in global and national economies and are receiving attention in a range of business-related activities [1], [2]. The success and advantages of web-based applications drive business at a greater speed and provide one of the promising research areas in modern times. Despite their significant advantages, web-based applications are vulnerable to cyber-attacks due to the ease of access to users' web-based applications are therefore inherently vulnerable [3].

A basic web application infrastructure consists of a front-end, back-end and communication service between the front-end and the back end. Front-end services are usually open for public access to submit requests to the back end via the communication service, and the back-end is commonly supported through database applications such as MySQL, SQLite, or Microsoft access [4], [5]. The inside attackers with administrative privilege, appropriate access control models and with knowledge of authentication details pose a significant threat to the organizations since they can modify the back-end system through tampering and altering data [6]. According to recent reports [7]–[10], handling inside attackers is very critical to organizations. In most methods, the strategies considered are simplistic and often retrieve values in such a way that the database does not require any validation, which leads to a mismatch, which in turn leads to attackers being able to access vulnerabilities. Further, attackers leave no footprint to allow them to be tracked after the event. In addition, the insider can attack the system either in an active or passive approach. An active approach involves tampering the original data to alter the database query results, and a passive approach includes unethical access and data usage. Thus, protecting the information of the database from inside attackers has become a non-trivial and very important issue that poses a significant challenge and is considered an important problem in this space.

Structured query language injection (SQLi) attack is considered one of the major vulnerabilities for web applications due to its straightforward impact on back-end databases and is reported in a number of applications and recent research findings [11], [12]. An SQLi attack can occur when the user sends malicious commands or simple modified texts to exploit the system and hence causes serious consequences such as data loss, system damage and possible denial of access to the authorized users [13]. This injection attack requires longer processing times for the remedial processes to fix the injected flaws in the code or in the back-end system. SQLi can also be very dangerous if the attacker can gain success by injecting anonymous users into the target system. It is observed that the usage of web-based applications is enormously increasing these days in different organizations such as federal, local, and state government based, commercial and educational. Hence, to ensure security, it is really important to come up with an efficient and simple technique that will be easy to implement. The study carried out in this paper focus on this technique.

In recent times, blockchain-based approaches have been proposed to handle unwanted cyber-attacks like phishing attacks, social engineering attacks, malware attacks, cryptocurrency [14], and web vulnerabilities. Blockchain is a decentralized ledger-based technology that works on a network of distributed nodes and ensures data security without relying on trusted third parties. Technically, a blockchain is a linked list where each node is called a block. Blocks are connected to each other cryptographically, and the first block in the list is known as the genesis block [15]. The next blocks after the genesis block contain the transaction information and other metadata such as a timestamp (which provides information of linkage in terms of the cryptographic hash chain of the network). Some of the key advantages of blockchain technology in cybersecurity are decentralization, confidentiality, integrity, sustainability, and its tracking and tracing capability.

Hence, in this paper, SQLi vulnerabilities are considered the basis of cyber-attack on web-based applications. A solution is proposed to address the problem mentioned above by combining a SQL query matching technique (SQLMT) and a blockchain framework to detect malicious queries generated by an inside attacker. Even though some works reported related to SQL query-based approaches, the proposed solution mentioned here is a new and efficient approach. The key differences between this work with the existing ones are outlined below. In summary, contributions stemming from this research article are: i) a modified SQLMT is proposed for comparing the queries received, ii) a blockchain-based technique is integrated with a database for query filtering, and iii) to justify the appropriateness of this proposed approach, two publicly available web applications, a dataset and some established validation tools are considered for experimental analysis.

The rest of the paper is organized. Section 2 provides the motivation and relevant insight for this research, focusing on the significance of the SQLi attack. This also gives the reader some more information related to the research. Section 3 describes the different components of the proposed method. Section 4 presents the implementation details of the proposed method, and section 5 describes the computational experiments conducted along with a comparative analysis of the computational experiment carried out, and finally, section 6 concludes the paper.

2. LITERATURE REVIEW

Web-based applications connect billions of people around the globe and serve their needs in various business and financial activities. Databases are considered the most commonly used storage mechanisms for these web-based applications. Various constraints such as foreign key constraints, data types of specific attributes, access control mechanisms, and firewalls are imposed on information access for security reasons. However, despite these impediments, the information stored in the databases can still be compromised through web application interfaces. Typically, compromise can take place through any one of the injection vulnerabilities such as cross-site scripting, SQLi, XML injection, XPath Injection, and HTTP response splitting.

SQLi attacks have increased over the years [16] and considerable research has been conducted to detect, identify, and mitigate SQLi attacks. For instance, Warszawski and Bailis [17] presented an SQLi prevention technique using MySQL database parser that detects and terminates malicious queries injected by the attacker. A technique based on the syntactic structure of SQL statements was developed in [18] that used a comparison mechanism for the queries. The comparison was performed between the queries that are provided by the user and the resultant query generated by the proposed system. A query matching approach was developed to detect illegal queries. Meanwhile, Thome *et al.* [19] built a query model and used it to check the queries received from the user. An SQLi prevention system was proposed in which the queries were transformed to strings prior to execution. This way, the authors attempted to consider the malicious queries as data instead of SQL commands to allow them some space to avoid detecting false alarms [20]. A similar approach was presented by Sharma and Jain [21], while they analyzed the syntactic structure of the SQL query [22]. Consequently, they developed a model for SQLi detection based on the integration of the client and server-side of the web platform. They used a filtering technique for the client-side and an entropy based query detection mechanism for the server-side [23].

Most of these existing research approaches have focused on using validation functions and using web application firewalls integrating with predefined statements. Mostly, these methods monitor and block SQL queries generated by outside attackers, but the mitigation strategy is made without having knowledge of the database to process them. Another aspect of vulnerabilities is frequent tampering of database information by attackers through modification and update of fields and tuples. These can be readily performed by a user who has privileged and administrative access to the database. One of the other shortcomings is that the proposed methods discussed above fail to distinguish between the original query and the malicious query. The monitoring and the inspection are done based on simplistic assumptions about the database operations. For example, they assume that the values retrieved from the database do not require any validation. This semantic mismatch leads to vulnerabilities.

The blockchain has stronger cryptographic features than the standard database system due to its strong immutable feature. Once data is written in a block or a transaction is performed, it cannot be easily updated or modified unless an agreement takes place between all the participants. The agreement between the participants is achieved by mathematical algorithms and helps to achieve synergy within the network. If any update takes place, it is logged as an updated transaction that supports traceability [24], which is one of the bases of fair communication between the network nodes. Although blockchain is a decentralized database technology that guarantees data immutability and modification resistance from an insider, there is still a chance that data can be altered by removing the proof of the user having the appropriate rights and privileges.

It is observed that many efforts have been vested in securing web-based applications. However, due to the lack of appropriate authentication techniques, the existing approaches are immune to injection threats, and the malicious codes can be easily bypassed by the system. Hence, in this study, integrating the SQLMT and blockchain framework together is proposed to help the system to inhibit inside attackers attempting SQLi attacks. This study considers a publicly available blockchain framework, multichain stream [25], and MySQL database by integrated them into the proposed method.

3. PROPOSED METHOD

The block diagram of the proposed methodology is shown in Figure 1, and the steps are described in detail in the following sub-sections. However, a brief description is provided here before addressing the detail. Firstly, the user submits a request through an SQL query to the system through the application interface. The query is then fed to the proposed framework, which identifies it as an external query (EQ) and checks its validity through the external query type (EQtype) checking system. The user or the internal attacker generates EQ. Next, EQs are checked with respect to the internal query type (IQtype). IQtype is the internal queries that are stored in the internal query model. At this stage, the structure of the EQs is checked to see if it is syntactically and grammatically corrected with respect to IQtypes, using the SQL query matching technique. Suppose the structure of EQ is correct and matches the relevant query within the model. In that case, it is forwarded to the query integrity checking system, which is a combination of blockchain framework and database. This is performed to ensure that the system is not compromised and that the user's request is verified. The main task is to check whether the EQ is making any unwanted modifications in the database (please refer to sub-section 3.1 for details). The blockchain mainly authenticates the query to make sure that it is not forged, and the user is not making any SQLi attacks. Before the result is returned to the user, the query is tracked as a valid query (i.e., type 'a' as shown in Figure 1 and stored in the log file for security purposes. The log file is a simple spreadsheet that is used manually to write the query types prior to returning the result to the user. This ensures the preservation of the data and performs this checking process to make sure that the inside attacker does not take over the data. If the matching process fails, then an SQLi attack is reported. Finally, the query is identified as a malicious or invalid query (i.e., type ‘b’ as shown in Figure 1, and is stored in the log file, leading to the user request to be rejected.

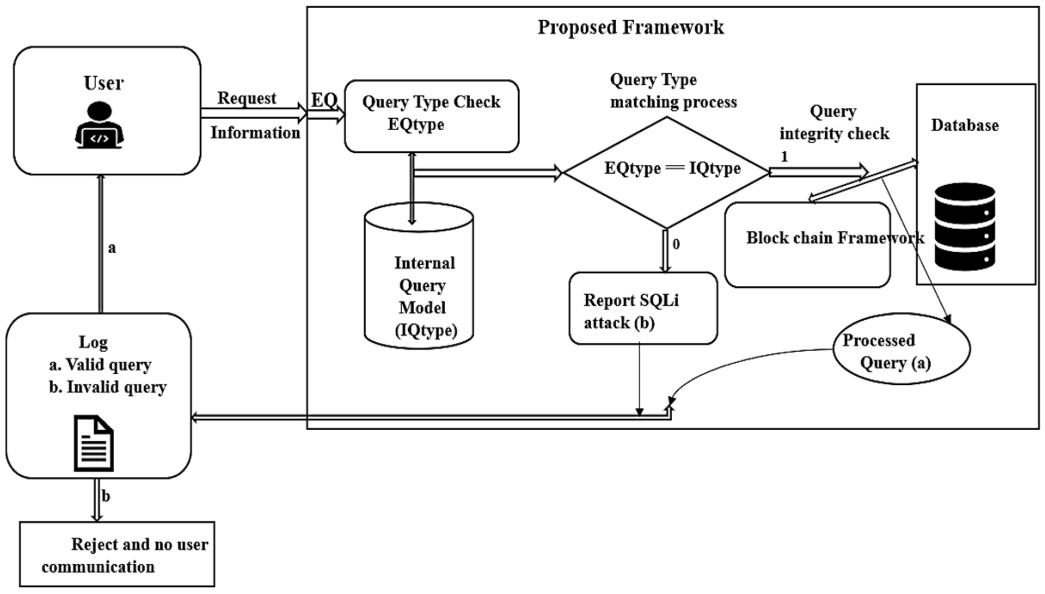

Figure 1. Block diagram of the proposed method

3.1. SQL query matching technique

An SQL parser is used in this study to parse the incoming queries [26]. The parser can parse all kinds of symbols within the query based on lexical, semantic and syntactic analysis principles. Through this analysis, the query type can be determined to be select, insert, update or delete. In addition, the parser helps to identify whether the user injects any invalid parameter prior to submitting the query to the system. The malicious input can severely impact the database and the attacker may be able to take complete control. So, the query type is determined, and the parameters are checked in accordance with the following steps. In the below steps, SQLMT is described considering a select query and where clause. However, this simple technique works with other queries and clauses by making necessary modifications (making different combinations of clauses with the queries).

First, the EQtype for select is checked with the help of the SQL parser. The SQL parser also checks if there are any additional input parameters or not. The rationale is that if the user modifies the query with an additional parameter, then the query structure is modified and changed, which indicates that an SQLi attack is in progress. On the other hand, if there are no such additional parameters detected with the help of SQL parser, then the query structure is left unmodified. Also, the parser calculates the number of tokens within the query. The characters after where clause in the EQtype query is extracted before proceeding to the next step. This is performed using a character extraction technique [27].

The query is then normalized to a simple statement by replacing any present encoded characters. This normalization is done to ensure that the query can be checked with respect to the query model without any complexity. In the query model, which is also used as the basis of query checking, the select query is modified prior to storing it there. For example, if there are any numeric parameters, the numbers are removed, the characters are removed for alphanumeric parameters so that a simple equivalence checking can be performed. This query is known as IQtype.

Now, the two queries, EQtype and IQtype are compared, and if they match, it is considered that there is no attack as shown in Figure 1. To note, in this work, any kind of select query is tagged with a number, say “1”. For example, the IQtype with select queries will be under the group id=1. This id tagging procedure is undertaken for transparency so that the performance of the team members involved in the operations do not remain oblivious to their existence. Similarly, the id for insert is set to 2, update to 3, and delete to 4. If the query fails to match the similarity verification in terms of tokens and the structure, then the query is flagged. This flagged token and any kind of additional words are retained for further analysis. The analysis is performed so that the system can know about these malicious queries and identify the SQLi attacks if it takes place in future.

This method is simple and easy to implement and has some differences from existing approaches. Any algorithm tester can enter input characters whatever they want for testing the system, and the proposed technique does not require any white or blacklisting strategies. This also reduces the pressure on working with the query size. However, there is one limitation in the proposed technique: if any query consists of both parameters from the user (inside attacker), and the system tester, then the parameters specified by the latter person should be at the end. Also, the character extraction technique used here is not observed within the existing techniques. This helps to extract any kind of characters from the select, insert, update and delete queries and creates a straightforward avenue for the comparison process.

3.2. Blockchain based integrity checking procedure

This subsection describes the integrity checking procedure using multichain-based blockchain technology. The public key infrastructure (PKI) that exists within the framework is considered in this study. This is used so that the user's public key (mechanism of the blockchain technology) can be retrieved by considering the unique identifier or primary key (i.e., *employee_name*). In order to integrate the blockchain with the MySQL database, some modifications are undertaken, as shown in Figure 2.

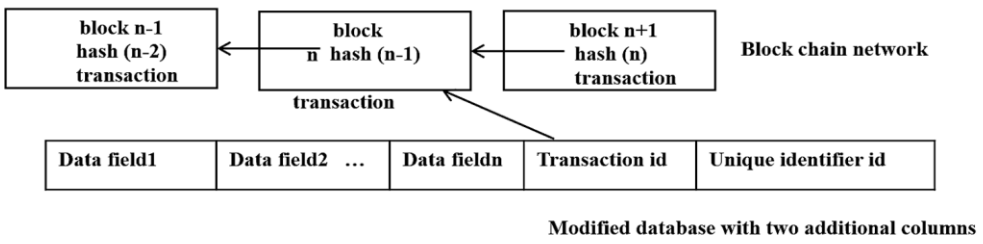

Figure 2. Modified database to integrate with the blockchain (multichain)

The top part of the diagram in Figure 2 represents the nodes, the blocks of the blockchain network, and the table at the bottom represents the modified table of the MySQL database. The data fields on the left-hand side of the table are the fields or columns that store different information within the database. Two additional columns are inserted, one that will store the state of the transaction, and the other one represents the information of the unique identifier. The column transaction id, will store the id of the transaction of the blockchain corresponding to the database row. The unique identifier id column stores the id of the user who issues the EQtype query and consequently names the column as *employee_name*. The communication to handle the transactions is done through a secure communication protocol. The overall integrity checking procedure is performed.

The application (multi-chain blockchain) extracts all the information related to the query (received from the SQLMT process) in the form of a tuple upon receiving a request from the user. For example, in the case of SELECT query, the value for all fields (columns) except the last two columns (transaction ID, and unique identifier ID) is extracted in a tuple. Assuming the tables in the database have a primary key, this key is concatenated along with the table name to generate the hash for the tuple. Next, the tuple is digitally signed using the user's private key and stored in the blockchain log. In addition to this hashed tuple, a digital footprint for each tuple is generated and stored in the blockchain for further verification. The digital footprint is generated by concatenating the tuple ID along with each column value except the last two and adding another 256-bit hash. If there is any NULL value for an attribute, it is skipped, and the next non-NULL value attribute is added. The tuple is then broadcasted to the blockchain network as a transaction activity. The system then waits for the transaction to be retrieved in a block in the first blockchain and receives a number of confirmations as a part of consensus policy from the peers. The consensus policy is reached if the tuple that is traversed (part of the transaction) is verified with the stored corresponding digital footprint. If any modification takes place, it gets logged into the blockchain, and the update is stored. Hence, for every update, the blockchain has the accountability which can trace any sort of changes or modifications. This helps to identify the existence of any internal culprits and whether they have any bad intention to alter or modify. After the transaction has received the confirmations, the system then checks the signature of the user with respect to the existing PKI stored in the unique identifier column of the modified database. This signature matching is performed to make sure that the transaction is verified.

Upon completing the verification, the system then retrieves the corresponding query from the internal log and modifies it by including the transaction id (transaction happened within the blockchain) of the user and sends the information to the database for the query. This integrity checking helps to determine whether any unwanted activities are performed by an insider and if so, to filter them out. Also, it helps to block any unwanted transactions from execution. If there are unwanted activities, the query is passed smoothly through this process to generate the user request and send the expected result. The procedures incur some latency (time consumption) due to tuple integrity checks between the digital footprint and the tuple that has traversed through the network as a transaction. It is done for every tuple, and the latency is considered negligible to overweigh the integrity of the data.

4. RESEARCH IMPLEMENTATION METHOD

The implementation details considered for this work are described in this section. The web application considered in this work is a modified version of an online based human resource application (OHR) [28], [29], which has six vulnerable pages. However, on top of this, some additional vulnerable pages are created to check the authenticity of the proposed approach.

The distributed blockchain, called the multi-chain framework, is considered in this work and stimulates the scenario that each participant in the web applications installs multi-chain in the network and then uses the proposed technique for connecting to the central multi-chain node. Each user is then assigned to a blockchain address for permission assignment. The permissions are distributed according to the main users addresses retrieved from the PKI, allowing them to append transactions and mine blocks. This enables the users to verify the transactions according to the proposed mechanism. The operational procedure of the overall proposed framework, including the SQLMT and blockchain process, is described in the next section. The performance throughput is also analyzed, and the results are shown in the following section.

In addition to the web applications, a publicly available dataset is also considered here. The dataset is called HTTP dataset CSIC 2010 [27]. This dataset comprises many automatically generated web requests. This was developed at Spanish Research National Council. The dataset automatically generates 36,000 normal HTTP requests and usually 25,000 unusual requests or more, and includes SQL injection attacks and other attacks such as file disclosure, parameter tampering, buffer overflow, and XSS. The efficiency of the proposed method is evaluated by considering these eight performance metrics: false acceptance rate (FAR), genuine acceptance rate (GAR), region of convergence (ROC) curve, true positive (TP), false negative (FN), precision, recall and f-measure [30], [31]. These metrics are used to determine how accurately the proposed method can detect the vulnerabilities in web applications.

The approaches used for comparison in this study are: the approach proposed by Kumar and Pateriya called Alg1 [32], the approach proposed by Cho and Pan called Alg2 [33], the approach proposed by Liban and Hilles called Alg3 [34], and the approach proposed by Djuric called as Alg4 [35]. For the sake of simplification and understanding, thenceforth, the proposed algorithm is called as PrAlg. In order to validate the SQLMT process, other SQLi attacks [36], [37] are also considered in this study. The attacks are: commenting the code (SQLi1), type mismatch (SQLi2), stacked query (SQLi3), union query (SQLi4), inference (SQLi5) and alternative query (SQLi6). The SQLi tools used are, Zap, Nikito, SQL inject me, and Wapiti. The metrics, precision, recall and F-measure are used to measure the performance of these tools with respect to the proposed hybrid approach.

5. EXPERIMENTAL RESULTS AND DISCUSSION

This section describes the computational simulations performed to obtain the results for this work. The first application considered is OHR, and some necessary modifications are made to this application for computational set-up. The database tables for this application correspond to the entities: administration, user information, scheduling users, departments, employee information, and employee scheduling. The relationships among them are established through the administration information table, which stores mainly the information about the executive team members. In this case, the cyber team members are assumed to be part of the administration. The table also stores the sensitive information about the employees that need to be protected. As described above, this table has two additional columns, one to hold the transactions and the other for the admin identifier. Restrictions to some other parts of the table are imposed using MySQL views preventing unwanted access. Now, the OHR application receives a query from the user, and then it is forwarded to the proposed system. First, the proposed system uses the SQLMT process for query checking and forwards to the blockchain. Once, the blockchain integrated database receives the query; it generates the hash of the tuple as described in subsection 3.2. Now, the row’s hash is signed, the signature is broadcasted through the network, and the system stores the actual query in its log. Next, this transaction is mined in a block and a certain number of confirmations are received so that the system can output the actual results, and the additional columns containing the transaction id and the admin information are updated. Once the update is performed, the verification is completed to check the authenticity. OHR retrieves the transaction information using the help of the proposed framework and fetches the public key from the PKI using the identifier. The signature of this transaction is then verified with this public key, and if successful, as well as the hash of the data, matches with the one present in the transaction, then the user will be able to see the results. Otherwise, it will be flagged as SQLi attack, and the query will be rejected. To compare OHR vulnerabilities, the performance metrics discussed in the previous section are used. Table 1 shows the ability of SQLi detection results for the various algorithms and the various SQLi's for the OHR application. The number "1" refers to the success of the specific SQLi detection, and "0" means the algorithm is not able to detect that specific SQLi. It is evident from the table that the proposed approach (i.e., PrAlg) can detect all the SQLi's for the test application OHR.

The execution time taken is also investigated for the proposed approach compared with the other algorithms. However, it is observed that, the proposed approach takes slightly longer times than the others due to the operational process of the blockchain, but not significantly so. For instance, while running the SELECT query using the integrity checking procedure, the proposed approach PrAlg consumed 0.575 second per row, and the other approaches Alg1, Alg2, Alg3 and Alg4 consumed 0.231, 0.331, 0.354, 0.421 seconds per row, respectively. This additional time is required due to some modifications required in the blockchain framework for the consensus protocol which needs to make necessary adjustments in the database. This requires some significant effort to ensure smooth transactions by leveraging the information from the queries to identify the SQLi attacks. The proposed approach is also implemented on the CSIC 2010 dataset to investigate its capability. On a good note, this proposed approach has achieved good results for the CSIC 2010 dataset as well by comparing with Alg1, Alg2, Alg3, and Alg4. The FAR and GAR scores achieved for this proposed approach are 10.12% and 78.45%, respectively. The scores for the other algorithms are: Alg1 (FAR: 28.15%, GAR: 46.33%), Alg2 (FAR: 20.15%, GAR: 61.25%), Alg3 (FAR: 26.35%, GAR: 54.12%) and Alg4 (FAR: 30.53%, GAR: 41.12%).

Table 1. SQLi detection ability of the proposed algorithm with respect to other algorithms for OHR

Algorithm	SQLi1	SQLi2	SQLi3	SQLi4	SQLi5	SQLi6
Alg1	1	1	0	0	1	0
Alg2	1	1	0	1	1	1
Alg3	1	1	1	1	1	0
Alg4	1	1	1	0	0	0
PrAlg	1	1	1	1	1	1

5.1. Comparative analysis and improvement

Although there are some similar works presented by the researchers using an SQL query matching technique, significant attention is required to identify the attacks. Thence, in this study, SQL query matching is integrated with multichain blockchain technology to achieve better results. In addition to the computational experiments conducted above, the time taken for each algorithm, including the proposed approach, is also observed. Also, to evaluate the effectiveness of the proposed approach, two algorithms Alg5 [38], and Alg6 [39] have been considered for further evaluation. The performance measurements are shown in Table 2.

Table 2. Performance measurements for the proposed approach with respect to other algorithms and methods

Algorithm/Methods	No of Execution time	Total Execution time	Detection Accuracy (in terms of F-measure)
Alg1	7	445	0.2698
Alg2	7	428	0.5435
Alg3	7	411	0.3145
Alg4	7	395	0.2196
Alg5	7	388	0.4465
Alg6	7	421	0.3875
Zap	7	422	0.5692
Nikito	7	419	0.3563
Wapiti	7	389	0.1851
Sql Inject Me	7	380	0.1812
PrAlg	7	491	0.9483

Table 2 summarizes the performance evaluation of the approaches considered in this study. The second column in this table represents the number of times each approach is executed for the testing purpose. For example, the number 7 in the second column means that the application runs 7 times to collect an average data to observe the trade-off between the execution time and detection accuracy of the approaches. The number of runtimes has been determined hypothetically. The third column represents the execution time in seconds taken by each of the algorithms to complete the task. This is the time elapsed between the user starts sending a request and finishing after receiving a reply from the system. In order to conduct these runs, google chrome is considered as a web browser for testing purposes. The table shows that the proposed approach consumes slightly more time than the others, which is insignificant in value. This extra time is rational since for the blockchain-based hybrid approach takes additional time to maintain atomicity, consistency and isolation properties of the database for any transactional queries of the blockchain. However, considering the web applications in this study, the transaction cost in terms of time is not too higher. The last column represents the capability of each algorithm in terms of detection accuracy, and the data shown here are the corresponding F-measure results, as discussed in the previous section. The results show that the proposed approach has a good detection capability (94.83%).

6. CONCLUSION

This paper proposes a hybrid framework that combines a query matching technique and blockchain technology to demonstrate the effectiveness in detecting a range of SQLi's. This proposed approach will be helpful for any web-based applications, as it is able to detect SQLi attacks. Unlike other approaches, this proposed framework is quite simple to implement, but a small amount of additional computational time is required, as the framework requires a number of confirmations from the peers working on the blockchain mechanism. However, the incurred additional computational cost can be accepted as the query integrity checking process provides a better solution in detecting SQLi's effectively. Besides the overhead issue mentioned earlier, another limitation of this work is that the approach sometimes fails to detect vulnerabilities if a dynamic attack pattern takes place. The performance of this proposed approach is measured using different metrics such as SQLi tools and a publicly available dataset, which shows the proposed approach can successfully detect all of the tested SQLi's. In future, this work will be extended using a more in-depth analysis of blockchain technology and various other SQLi tools and considering complex web frameworks and applications. One aspect that will be emphasized is developing an automated solution for the database administrator focused on the query model to handle dynamic and complex queries. In addition, this work will be extended by testing the proposed approach and other methods on different web browsers. This will ensure that the method works across platforms and can detect SQLi attacks regardless of the type of web platform.

REFERENCES

[1] D. Calvaresi, D. Cesarini, P. Sernani, M. Marinoni, A. F. Dragoni, and A. Sturm, "Exploring the ambient assisted living domain: a systematic review," *Journal of Ambient Intelligence and Humanized Computing*, vol. 8, no. 2, pp. 239–257, Apr. 2017, doi: 10.1007/s12652-016-0374-3.

[2] H. Mora, V. Gilart-Iglesias, R. Pérez-del Hoyo, and M. Andújar-Montoya, "A comprehensive system for monitoring urban accessibility in smart cities," *Sensors*, vol. 17, no. 8, Aug. 2017, doi: 10.3390/s17081834.

[3] M. Zhao, J. Grossklags, and P. Liu, "An empirical study of web vulnerability discovery ecosystems," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015, pp. 1105–1117, doi: 10.1145/2810103.2813704.

[4] W. El-Hajj, G. Ben Brahim, H. Hajj, H. Safa, and R. Adaimy, "Security-by-construction in web applications development via database annotations," *Computers and Security*, vol. 59, pp. 151–165, Jun. 2016, doi: 10.1016/j.cose.2015.12.004.

[5] A. M. Osman, A. Dafa-Allah, and A. A. Mohammed Elhag, "Proposed security model for web based applications and services," in *2017 International Conference on Communication, Control, Computing and Electronics Engineering (ICCCCEE)*, Jan. 2017, pp. 1–6, doi: 10.1109/ICCCCEE.2017.7866696.

[6] A. S. Bretas, N. G. Bretas, and B. E. B. Carvalho, "Further contributions to smart grids cyber-physical security as a malicious data attack: Proof and properties of the parameter error spreading out to the measurements and a relaxed correction model," *International Journal of Electrical Power and Energy Systems*, vol. 104, pp. 43–51, Jan. 2019, doi: 10.1016/j.ijepes.2018.06.039.

[7] H. Poll, *Vormetric insider threat report: Trends and future directions in data security-Healthcare edition*. Harris Poll, 2015.

[8] L. Xiangyu, L. Qiuyang, and S. Chandel, "Social engineering and insider threats," in *2017 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, Oct. 2017, pp. 25–34, doi: 10.1109/CyberC.2017.91.

[9] S. Zeadally, B. Yu, D. H. Jeong, and L. Liang, "Detecting insider threats: solutions and trends," *Information Security Journal: A Global Perspective*, vol. 21, no. 4, pp. 183–192, Jan. 2012, doi: 10.1080/19393555.2011.654318.

[10] M. C. Theis *et al.*, "Common sense guide to mitigating insider threats," Pittsburgh, PA, 2019.

[11] Z. S. Alwan and M. F. Younis, "Detection and prevention of SQL injection attack: a survey," *International Journal of Computer Science and Mobile Computing*, vol. 6, no. 8, pp. 5–17, 2017.

[12] D. Appelt, A. Panichella, and L. Briand, "Automatically repairing web application firewalls based on successful SQL injection attacks," in *2017 IEEE 28th International Symposium on Software Reliability Engineering (ISSRE)*, Oct. 2017, pp. 339–350, doi: 10.1109/ISSRE.2017.28.

[13] G. Deepa and P. S. Thilagam, "Securing web applications from injection and logic vulnerabilities: Approaches and challenges," *Information and Software Technology*, vol. 74, pp. 160–180, Jun. 2016, doi: 10.1016/j.infsof.2016.02.005.

[14] S. Rahman, J. N. Hemel, S. J. A. Anta, H. Al Muhee, and J. Uddin, "Sentiment analysis using R: an approach to correlate cryptocurrency price fluctuations with change in user sentiment using machine learning," in *2018 Joint 7th International Conference on Imaging, Vision and Pternational Conference on Informatics, Electronics and Vision (ICIEV) and 2018 2nd Inattern Recognition (icIVPR)*, Jun. 2018, pp. 492–497, doi: 10.1109/ICIEV.2018.8641075.

[15] V. J. Morkunas, J. Paschen, and E. Boon, "How blockchain technologies impact your business model," *Business Horizons*, vol. 62, no. 3, pp. 295–306, May 2019, doi: 10.1016/j.bushor.2019.01.009.

[16] Y.-S. Jang and J.-Y. Choi, "Detecting SQL injection attacks using query result size," *Computers and Security*, vol. 44, pp. 104–118, Jul. 2014, doi: 10.1016/j.cose.2014.04.007.
[17] T. Warszawski and P. Bailis, "ACIDRain," in *Proceedings of the 2017 ACM International Conference on Management of Data*, May 2017, pp. 5–20, doi: 10.1145/3035918.3064037.
[18] J. Caseirito and I. Medeiros, "Finding web application vulnerabilities with an ensemble fuzzing," in *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks - Supplemental Volume (DSN-S)*, Jun. 2021, pp. 19–20, doi: 10.1109/DSN-S52858.2021.00020.
[19] J. Thome, L. K. Shar, D. Bianculli, and L. Briand, "An integrated approach for effective injection vulnerability analysis of web applications through security slicing and hybrid constraint solving," *IEEE Transactions on Software Engineering*, vol. 46, no. 2, pp. 163–195, Feb. 2020, doi: 10.1109/TSE.2018.2844343.
[20] W. Masri and S. Sleiman, "SQLPIL : SQL injection prevention by input labeling," *Security and Communication Networks*, vol. 8, no. 15, pp. 2545–2560, Oct. 2015, doi: 10.1002/sec.1199.
[21] C. Sharma and S. C. Jain, "Analysis and classification of SQL injection vulnerabilities and attacks on web applications," in *2014 International Conference on Advances in Engineering and Technology Research (ICAETR-2014)*, Aug. 2014, pp. 1–6, doi: 10.1109/ICAETR.2014.7012815.
[22] Z. Su and G. Wassermann, "The essence of command injection attacks in web applications," *ACM SIGPLAN Notices*, vol. 41, no. 1, pp. 372–382, Jan. 2006, doi: 10.1145/1111320.1111070.
[23] D. G. Kumar and M. Chatterjee, "MAC based solution for SQL injection," *Journal of Computer Virology and Hacking Techniques*, vol. 11, no. 1, pp. 1–7, Feb. 2015, doi: 10.1007/s11416-014-0219-6.
[24] B. K. Mohanta, D. Jena, S. S. Panda, and S. Sobhanayak, "Blockchain technology: A survey on applications and security privacy Challenges," *Internet of Things*, vol. 8, Dec. 2019, doi: 10.1016/j.iot.2019.100107.
[25] A. Sawant, N. Prabhu, and S. Nagpure, "Securing IoT using MultiChain," *SSRN Electronic Journal*, 2019, doi: 10.2139/ssrn.3370759.
[26] D. Cao and D. Bai, "Design and implementation for SQL parser based on ANTLR," in *2010 2nd International Conference on Computer Engineering and Technology*, 2010, pp. 276–279, doi: 10.1109/ICCET.2010.5485593.
[27] C.T. Giménez, A.P. Villegas, G.Á. Marañón, "HTTP data set CSIC 2010," *Information Security Institute*. (Spanish Research National Council), 2010, doi: 10.23721/100/1478804.
[28] M. S. Aliero, I. Ghani, K. N. Qureshi, and M. F. Rohani, "An algorithm for detecting SQL injection vulnerability using black-box testing," *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, no. 1, pp. 249–266, Jan. 2020, doi: 10.1007/s12652-019-01235-z.
[29] A. K. Dalai and S. K. Jena, "Neutralizing SQL injection attack using server side code modification in web applications," *Security and Communication Networks*, vol. 2017, pp. 1–12, 2017, doi: 10.1155/2017/3825373.
[30] S. el Idrissi, N. Berbiche, F. Guerouate, and S. Mohamed, "Performance evaluation of web application security scanners for prevention and protection against vulnerabilities," *International Journal of Applied Engineering Research*, vol. 12, no. 21, pp. 11068–11076, 2017.
[31] M. Sokolova, N. Japkowicz, and S. Szpakowicz, "Beyond accuracy, f-score and ROC: a family of discriminant measures for performance evaluation," in *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2006, pp. 1015–1021.
[32] L. Zhang, Q. Gu, S. Peng, X. Chen, H. Zhao, and D. Chen, "D-WAV: A web application vulnerabilities detection tool using characteristics of web forms," in *2010 Fifth International Conference on Software Engineering Advances*, Aug. 2010, pp. 501–507, doi: 10.1109/ICSEA.2010.85.
[33] Y.-C. Cho and J.-Y. Pan, "Design and implementation of website information disclosure assessment system," *Plos One*, vol. 10, no. 3, Mar. 2015, doi: 10.1371/journal.pone.0117180.
[34] A. Liban and S. M. S. Hilles, "Enhancing Mysql Injector vulnerability checker tool (Mysql Injector) using inference binary search algorithm for blind timing-based attack," in *2014 IEEE 5th Control and System Graduate Research Colloquium*, Aug. 2014, pp. 47–52, doi: 10.1109/ICSGRC.2014.6908694.
[35] Z. Djuric, "A black-box testing tool for detecting SQL injection vulnerabilities," in *2013 Second International Conference on Informatics and Applications (ICIA)*, Sep. 2013, pp. 216–221, doi: 10.1109/ICoIA.2013.6650259.
[36] S. Mavromoustakos, A. Patel, K. Chaudhary, P. Chokshi, and S. Patel, "Causes and prevention of SQL injection attacks in web applications," in *Proceedings of the 4th International Conference on Information and Network Security-ICINS '16*, 2016, pp. 55–59, doi: 10.1145/3026724.3026742.
[37] B. Deva Priyaa and M. I. Devi, "Fragmented query parse tree based SQL injection detection system for web applications," in *2016 International Conference on Computing Technologies and Intelligent Data Engineering (ICCTIDE'16)*, Jan. 2016, pp. 1–5, doi: 10.1109/ICCTIDE.2016.7725367.
[38] O. Lounis, S. E. B. Guermeche, L. Saoudi, and S. E. Bouhouita Guermeche, "A new algorithm for detecting SQL injection attack in Web application," in *2014 Science and Information Conference*, Aug. 2014, pp. 589–594, doi: 10.1109/SAI.2014.6918246.
[39] V. Shanmughaneethi, "Detection of SQL injection attack in web applications using web services," *IOSR Journal of Computer Engineering*, vol. 1, no. 5, pp. 13–20, 2012, doi: 10.9790/0661-0151320.

BIOGRAPHIES OF AUTHORS

Md. Hasan Furhad has completed his PhD in Computer Science from UNSW Australia. Prior to PhD, he has completed Master by Research with an outstanding result and good number of research publications from University of Ulsan, South Korea. His research expertise includes cyber security, computational image processing and covers a wide range of computational application domains. Hasan is currently working as a Senior Cybersecurity Advisor in the Risk Advisory team at Deloitte Canberra. Prior to joining industry, he was involved in the academic space for 13 years, and now working towards impactful contribution in the cybersecurity space. He can be contacted at email: hfurhad@deloitte.com.au.

Ripon K. Chakrabortty (Member, IEEE) is the Deputy Director (acting) of the Capability Systems Centre (CSC) and lecturer on System Engineering and Project Management at the School of Engineering and Information Technology, the University of New South Wales (UNSW Australia), Canberra. He is also the program coordinator for Master of Decision Analytics and Master of Engineering Science programs in UNSW Canberra at ADFA. Dr Chakrabortty leads the optimization and machine learning efforts in the Capability System Centre. He is currently the Group Leader of Cross-Disciplinary Optimisation Under Capability Context Research Team. As an educator and active researcher in his field, his focus is to deliver authentic teaching by pursuing real-life and practical knowledge to diversified students. Inarguably, his research involvement in the discipline of decision analytics, applied operations research, systems engineering and project management, Industry 4.0 aids him to expose knowledge gaps in the body of knowledge (e.g., he has got more than 130 internal publications at his credit). He can be contacted at email: r.chakrabortty@unsw.edu.au.

Michael J. Ryan is the Director of the Capability Associates Canberra. He holds bachelor, masters and doctor of philosophy degrees in engineering. In addition, he has completed two years formal engineering management training in the United Kingdom. He is a Fellow of Engineers Australia (FIEAust), a Chartered Professional Engineer (CPEng) in electrical, ITEE and systems engineering colleges, a Senior Member of IEEE (SMIEEE), a Fellow of the International Council on Systems Engineering (INCOSE), and a Fellow of the Institute of Managers and Leaders (FIML). He has over 35 years of experience in communications engineering, systems engineering, project management, and management. He has lectured in a range of subjects including communications and information systems, systems engineering, requirements engineering and project management and he regularly consults in those fields. He is the author/co-author of 12 books, three book chapters, and over a 250 refereed journal and conference papers. He can be contacted at email: mike.ryan@ieee.org.

Jia Uddin is as an Assistant Professor, Department of AI and Big Data, Endicott College, Woosong University, Daejeon, South Korea. He received Ph.D. in Computer Engineering from University of Ulsan, South Korea, M.Sc. in Electrical Engineering (Specialization: Telecommunications), Blekinge Institute of Technology, Sweden, and B.Sc. in Computer and Communication Engineering, International Islamic University Chittagong, Bangladesh. He was a visiting faculty at School of computing, Staffordshire University, United Kingdom. He is an Associate Professor (now on leave), Department of Computer Science and Engineering, Brac University, Dhaka, Bangladesh. His research interests are industrial fault diagnosis, machine learning/deep learning-based prediction and detection using multimedia signals. He can be contacted at email: jia.uddin@wsu.ac.kr. Fax: 0082-70-7545-9767.

Iqbal H. Sarker received his Ph.D. under the department of Computer Science and Software Engineering from Swinburne University of Technology, Melbourne, Australia in 2018. Currently, he is working as a faculty member of the Department of Computer Science and Engineering at Chittagong University of Engineering and Technology. His professional and research interests include Data Science, Machine Learning and AI, Data-Driven Cybersecurity and Threat Intelligence, Context-Aware Smart/Intelligence Computing, Smart Cities, Systems and Security. He has published over 100 research papers including top-ranked Journals and Conferences with reputed scientific publishers like Elsevier, Springer Nature, IEEE, and ACM. Moreover, he is an author of the book titled Context-aware machine learning and mobile data analytics, published by Springer Nature, Switzerland. Recently, he received the recognition of the World’s TOP 2% Researcher/Scientists, listed by Elsevier and Stanford University, USA, 2021. He is one of the founders of the International AIQT Foundation, Switzerland, and a member of ACM and IEEE. He can be contacted at email: iqbal@cuet.ac.bd.