Wireless Personal Communications
https://doi.org/10.1007/s11277-020-07243-z

A Novel Method to Detect and Prevent SQLIA Using Ontology to Cloud Web Security

K. Naveen Durai[1] · R. Subha[1] · Anandakumar Haldorai[1]

Abstract
Many modern day web applications deal with huge amount of secured and high impact data. As a result security plays a major role in web application development. The security of any web application focuses on data the application handles. The web application framework should prevent and detect web application vulnerabilities. Data will be stored in a database, so the OWASP categorized vulnerability SQL Injection Attacks (SQLIA) is the most critical vulnerability for a web application. An Ontology based model for preventing and detecting SQLIA using ontology (SQLIO) is proposed which implements Ontology Creation and prediction rule based vulnerabilities model. The proposed methodology provides prevents and detects SQLIA web vulnerability to a greater extent in cloud environment.

Keywords SQL injection attacks (SQLIA) · Hyper text transfer protocol (HTTP) · Open web application security project (OWASP) · SQL injection ontology (SQLIO)

1 Introduction

Data sharing is quickly expanded around the globe using the web application and web administrations, which enhances the productive development of the e-business [1]. This development does not only involve the output of e-business, but also the analysis of digital vulnerabilities due to web utilization. As a result, it affects the security objectives: Availability, Confidentiality and Integrity.

Various actions have been put in place to mitigate the vulnerabilities, which include security technologies likes the IDS scanners and web-based application firewall [2]. Nevertheless, code-level security is vital for creating powerless-free applications [3]. During

✉ K. Naveen Durai
naveendurai.k@sece.ac.in

R. Subha
kris.subha@gmail.com

Anandakumar Haldorai
anandakumar.psgtech@gmail.com

[1] Department of Computer Science and Engineering, Sri Eshwar College of Engineering, Coimbatore, TamilNadu, India

Published online: 28 March 2020

the process of testing and improvements of these technologies, engineers have to familiarize themselves with the dangers of data and the effects of attacks on profiles. As such, the ultimate goal is to distinguish the attacks and release the stretches. Therefore, formulating a secure and more enhanced website application is necessary. Moreover, the dependency on security experts to evaluate the web application and its security might be limited [4].

The security ontology that is available today provides scientific categorization of vulnerabilities, threat and attacks requires construing the data, which is relevant in foreseeing threats without grouping them. Based on an individual objective to measure these problems, the projected approach potentially evaluates the web application vulnerabilities and threats, which might be mishandled by attacks [5]. The projected approach uses the SWRL guidelines and the surmising procedures in foreseeing the vulnerabilities concerning the possible attacks [6]. These projected threats have been grouped in light of their seriousness level and the framework additionally recommends aversion and moderation techniques.

The sections are organized as follow Sect. 2 discussed detailed review procedure of SQLIA. In Sect. 3 the proposed approach elaborated and discussed. The result described in Sect. 4. Finally the Sect. 5 concluded with future recommendations.

2 Literature Review

OWASP The abbreviation is Open Web Application Security Project, which represents the 501c3 non-profit global charity organization [7]. The objective of the organization is to improvise the application software security.

DVWA The abbreviation represents the Damn Vulnerable Web Application (DVWA), which is completely a PHP/MySQL-based practical tool. This tool is used for testing the security web developers' and professionals' tools in the legitimate practical environment utilized for the aid. Moreover, it trains users concerning the Secure Web Development. The Live CD and the Source Codes arefree-of-cost. Topics covered in this particular tutorial are XSS reflection, SQL Injection (Blind), well-stored XSS, Command Execution, Inclusion of files, CSRF, Upload, and Brute Force. When there arises a comparison between the OWASP and DVWA, DVWA is less wide-ranging and covers up only the least count of topics. Regardless of all, they are even restricted in exposing hand full of information [8]. It reflects not only in straight topic narrations but also in solution providing, hints enhancing and delivering guidelines. In accordance, they afford information to the users via the hyperlinks on related websites.

Web Security Dojo This can be described in short as, 'free open-source self-contained training environment' and this test is used for Application Security penetration. The term Dojo is abbreviated from Eq. (1),

$$\text{Tools} + \text{Targets} = \text{Dojo} \tag{1}$$

In this scenario, the VMware image is free-of-cost. The attractive part is, it is downloadable by the users and allows them to install in their own machine with the rapidity they use and they also hand over the complete documentation to the downloaded users. It also comes out with providing some of the default targets like JSON demos, DVWA (Damn Vulnerable Web App) and REST Demos [9]. Other functions provided are Tools (Ex: burp suite), Hackme Casino vulnerable application, WebGoat, and Insecure Web App. Despite

all the above-mentionedadvantages, it has another positive point on efficiency to the environment and its notable drawback is, beginners in security practice cant tend to choose this option as it is not meant for them of course.

Daffodil For learning purposes, this open source web application project is proposed. This has much and more similarities as DVWA and OWASP applications. Its package consists of exercises and solutions that are applicable to some chosen Web application vulnerabilities [10]. The criteria faced are the same i.e. lack of appropriate topic discussions. Then it becomes the responsibility of the users to fetch the respective practical information for the practice they are intended into. If at all this is designed user-friendly then the case would have been vice-versa as it will be very useful to the beginners' indeed.

2.1 SQLIA Detection Methods

Static Approaches In [11] proposed static approach, during compilation, the possibilities for the occurrence of SQL injections are distinguished or neutralized by the Static approaches. The procedure in doing this is, initial step initiates in scanning the heuristics/ leverage information flow analysis/ application then by using this, SQL injection vulnerable code(s) are identified. At the same time, this will prolong in lots of changes in the source code which will then stretch out as a huge burden to the programmers. Existing web applications will also have an uncomfortable feeling when the source codes are modified. Keeping all thismind, the researchers are trying to execute a dynamic analyze within users' input through SQLs and among those the malicious attacks will be blocked during runtime. Such static analysis is narrated below:

SAFELI To detect SQL Injection Vulnerabilities, a Static Analysis Framework is proposed. The identification of SQL Injection attacks is executed during compiling. Two main rewards of this static framework tool include: (i) White box Static Framework and (ii) Hybrid Constraint Solver [12]. Inside the White Box Static Framework, the byte codes are considered to be numerous out of all and strings are meant as the core. While for Hybrid-Constraint Solver, a string analysis tool that is efficient is implemented and the string, Boolean and integer variables are dealt with. These implementations are inherited through ASP.NET Web applications and the main advantage is, this can even detect the vulnerabilities that actually the black-box vulnerability scanner cannot. Considering the string constraints, this mechanism is a fabulous approximation. Then the shortcoming is its facilitation only to the ASP.NET vulnerabilities.

In order to recognize the input manipulation vulnerabilities [12], an automated testing is undergone by the Static-analysis-based tool 'SQLUnitGen'. When there arises a comparison between SQLUnitGen tool versus FindBugs, SQLUnitGen is confirmed as a static analysis tool. This is communicated as an outstanding mechanism as it concluded with constructive results in the phase of false positive as such crisis is lacking in all the conducted experiments.

WebSSARI This is all about checking/analyzing the sensitive functions that are conveyed as spoiled flows against preconditions and of course it also uses a static analysis [13].

Dissimilar to other types, the input that passed out successfully from the whole predefined set of filters are considered as the vital point while this is accomplished.

Context Sensitive String Evaluation (CSSE) Finding the root cause of SQLIA is the basic idea innovated behind this sort of approach. The data origin may be of any one among these either developer-provided or user-provided and that is the so-called root cause. They could definitely conclude that the data presented by a user are un-trusted and the presented by applications are trusted. Context-Sensitive String Evaluation (CSSE) acts as a base for syntactic analysis where the un-trusted metadata usage is determined. Another important cause for injection occurrence is development stage's programming flaws [14]. CSSE flows like a syntactical analysis based one, where numerical constants and string constants are ultimately distinguished. Then non-numeric characters are eliminated from numeric identifiers and in alphanumeric identifiers, the unsafe characters are eliminated. The entire above-mentioned scenario will tentatively happen before queries are delivered to the server's database. As such, the crisis in this approach is:

1. Unsafe characters' initialization is web programmer dependent.
2. Application functionality restriction will vigorously happen while unsafe characters are removed

2.2 Dynamic Approaches

The [15] proposed dynamic approach. There are far differences between Dynamic analysis and static analysis. None of the adjustments are done in the web applications while locating vulnerabilities of SQL injection attacks using dynamic analysis. All types of vulnerabilities including SQLIAs vulnerabilities within the web application are scanned using Open source program Paros. There are two points that make Paros imperfect. They are (i) scanning are done based on predetermined attack codes, (ii) for sparing the success-rate of the attack, HTTP responses are used. Dynamic approach category comprises many proposed techniques. Noticing the un-trusted data and their flows over the programs are tracked and these actions are performed as the security policies in the taint-based technique. It is the un-trusted input that is used in the creation of various types of SQL tokens in the SQL queries and such queries are rejected using context-sensitive analysis which is used in the taint-based technique. As this analysis method does not require any web application adjustments it is considered one of the best advantageous methods. Of course, a few numbers of vulnerabilities can be rectified as they are of predefined attacks spontaneously other vulnerabilities are supposed to be sorted out and fixed manually by the developers. The proposed method can be applied at websites developed using open source framworks. A listed set of dynamic approaches are given below:

Dynamic Candidate Evaluations Approach [16] proposed CANDID. SQL Injection attacks are automatically prevented when this Dynamic Candidate Evaluations method is used. As per the developer's design phase, query structures are extracted from every SQL query location vigorously in this framework. During the statement preparation, the applications are modified manually and this technique ruined out the altering issue. Statistically speaking, this tool might be considered so good in some of the cases while whereas in others actually it is not. Its inefficiency shall be reflected amongst applying the same at

a wrong level, during external functions dealing and most importantly when the scheme capability is limited.

Parse Tree Validation Approach Parse tree framework is adopted by [17] proposed. The original statement and the parse-tree of a certain statement, which appear in the runtime, are contrasted with each other. Until a match has been grasped the statement execution was not stopped. SQLGuard is used to experiment with this method in a Web application (developed by a student). Perhaps it is determined highly about its efficiency, it has its own shortcomings as, Input lists (black or white) and additional overhead computation.

Positive Tainting and Syntax Aware Evaluation [18] proposed, tentatively in this form, to detect SQLIA valid input string are supplied originally to the system. Initialization plays a vital role as this would help in categorizing the input strings during the runtime and from those categorizations the trusted markings and un-trusted markings are circulated. The propagated strings' estimation is undergone through 'syntax aware evaluation' (dynamic analysis based). This evaluation acts as a firewall through which the query trespassing is prohibited into the database server for furthermore processing when un-trusted strings show up their availability within the query. If it is about the trusted strings then the case is, it will be identified and marked by considering the inputs in mind. The strings could be categorized as:

1. Hardcoded strings
2. implicitly Java created Strings
3. Strings initiated via external sources.

Evaluation of the syntax will be performed in the database interaction point as soon as the case illustrated is a syntax-aware evaluation. A web programmer defines the functions and the functions are nothing but the trust policies as defined by the Syntax. A query receives a positive output and can be processed into the database server only after the pattern matching is performed by the functions. The crisis in using this method is:

1. Developers-dependent trusted strings' Initialization
2. Chance for second-order attack arises due to the constant storage of trusted strings.

SQLGuard [19] proposed to mitigate the attacks on the SQL, injected threats, the SQLGuard technique utilizes a parse tree validation and this is also a runtime analysis technique. As discussed in the previous sections the SQL injections are highly dangerous that the programmers expected output query structure shall totally vary. The simplest rescue operation undergone to find out the SQL injection attacks insertion is by simultaneous checking of SQL query structure duly before and after the input values are submitted by the users. It is not so definite about capturing the query structure accurately during compile time using static analysis. All these controversies led us to a better view that the unfair part is the comparison done between the SQL query structure during Compile-time and runtime. During the execution time tree structures capturing and this is what SQLGuard is attempting for. The process proceeds by using a random key in wrapping a user input and then a single token replace the wrapped user input. Then the major part of SQL query

usage is in the generation of the proposed SQL query structure that is captured at some stage in runtime and it does not involve user input with it.

SQLProb [20] proposed SQLProb. This is also one among the existing dynamic analysis approaches. Usage of MySQL Proxy (a customized one) in this approach is for: query collection, user-input extraction, parse-tree generation and user-input validation. The initial stage is for the collection of data where queries are collected using the Query Collected whereby the programmer actually wanted them to be and are equivalently saved in the system. Simultaneously sending the application generated query to both User Input Extractor module and Parse Tree Generator module is carried out in the query evaluation phase. JavaCC is used in the Parse Tree Generator module to generate a tree for the query. All the queries collected from the data collection phase are aligned by means of the Needleman-Wunsch algorithm (an enhanced version) and this is the responsibility of the User Input Extractor module. The resemblance of the query and the all the collected queries are estimated using the algorithm.

Prototype query is none but a query that possesses the highest similarity value amongst the collected queries. This technique turns out to be an absolute expensive technique as it requires as much number of comparisons as the number of queries. As a remedial measure for this size consumption in the query collection, the system will cordially work on the preprocessing tasks where the collected queries are aggregated and clustered. Needleman–Wunsch algorithm is exceedingly used to align the particular query versus its prototype query and then the user input portion is extracted. Thus extracted user input and tree generated via the Parse Tree Generator module are provided into the User Input Validator module. The collection oof leaf nodes shall be included into the parse tree and the user-input, which is the same set of leaf nodes, N is equivalent to the collection of nodes. hen an upward depth-first search will be conducted for user input validation in all the nodes of N to find a path which intersects with the internal node. With the target of attaining all the leaves of the tree, the breadth-first search will be performed in the internal node. To reject the set of leaves with the reason they are malicious, the result of it should be a superset of N.

The motto [21] of improvising the user authentication mechanism this scheme adopts itself into the hash value approach. The hash value is equivalent to username and password. Development of the prototype SQLIPA (SQL Injection Protector for Authentication) is certified for framework testing. When the user account creation has successfully completed the password and username hash values are formed and calculated in the runtime process.

2.3 Hybrid Approaches

The [22] proposed static analysis will be performed during development and such static analysis will be combined with some runtime dynamic monitoring and this shall be termed as a countermeasure. In each and every access point of the database the application involves in the generation of SQL query models and these models are put together during the phase of development considering the static evaluation according to its database utility. The checking procedure will be carried during the runtime process whereby the SQL information is counterchecked with the inner SQL model before queries are transferred to the database. The Recovery logics are applied and may be executed by the system developers during the process of mismatching the queries, which are considered to be exceptions

and the SQL injections. These considerations appear to be another form of comparison between the runtime query parse tree and the parse tree of the initiate statement, which provides safe and sound vulnerable SQL statements. Considering the difference in both the types, in the original statement both are created during runtime whereas in other one input parse tree is created only during runtime as it is under the development phase.

2.4 Collective Method of Static and Dynamic Enquiry

In [23] proposed this approach. In the progressive scan for SQL injection attack, this technique uses the merits of both the dynamic evaluation and static analysis techniques. In elaboration, web pages are analyzed and concurrently SQL queries are generated to examine the results. The tasks performed by SQL Check are (i) SQL injection attacks are defined and (ii) compiler parsing techniques and context-free grammars are used to propose a complete and healthy algorithm. All the potential SQL queries can be formed after the SQL queries found in the web applications are executed through hotspots and this is the AMNESIA proposed method. JSA library is used in the accordance to classify the static and the dynamic SQL queries. Before classification both the SQL queries had been evaluated. Moreover, such hybrid approaches are described below.

2.5 SQLIA Prevention Using Stored Procedures

The [24] proposed, Subroutines in the database are so-called stored procedures. Static analysis and runtime analysis are embedded together to implement the prevention in these stored procedures. Where, (i) static examination—applied for the identification of commands (stored process parser helps in achieving these) and (ii) runtime analysis—for input identification SQL checker is used. Security of potential vulnerabilities will be fortified when there is a combination between runtime checking and static investigation and it is Huang et al.'s proposition. SourceForge.net is utilized to Web application Security by Static Analysis and Runtime Inspection (WebSSARI). Talking about its drawback, they cannot deduct the SQL Injection Vulnerabilities (SQLIVs) and rather it can produce input lists (either White or black).

2.6 Automated Approaches

The [25] proposed automated approaches. Uses of automated approaches are highlighted by Mei Junjin. The two important schemes are Web-attack scanning and the Static examination to find bugs. The possible bugs in SQLIAs are noticed and variable made SQL query are warned about the same in this Static analysis Find Bugs approach. When the process of scanning is involved for Web vulnerability, for crawling software agents is used, Web applications are scanned and under the complete observation about the attack's behavior, the vulnerabilities are detected [26]. Scheme–Query values are separated from the SQL structure using the prepared statement which is supplied in SQL and this is a single line description about Thomas et al.'s Scheme. For instance, the programme will keep the SQL query pretty empty and it is narrated as 'Skeleton' and those unfilled spaces are filled in at runtime. As any changes in the SQL structure becomes impossible, the injection into SQL queries becomes pretty harder when prepare statement is used. An automated prepared statement generation algorithm is suggested by Thomas et al. with his utmost concern

about removing SQL Injection Vulnerabilities. The Table 1 shows the overall comparison methods.

Thus, it is essential to a fundamental strategy for the injection of SQL whereby the model can decide not only whether to detect the SQL injection, but also whether to prevent. In order to build such a model, it is desirable to develop better detection classifiers algorithm which would certainly improve the efficiency of SQL injection detection techniques with less computation involved and improved scalability.

3 System Architecture of SQLIAO

The proposed framework design as appeared in Fig. 1 differentiates into information gathering, ontology creation, rules generation and interface creation. The rile generation and ontology-creation is major element that is used to formulate threat expectation guidelines. The data based is applicable to safeguard information for the web threats, attacks and the vulnerabilities as how the ontology displays them. The critical utility of the vulnerability ontology model is expandability and reusability in reference to the security necessities of the website applications. Each ontology display is considerable sub-isolated into the submodules [27]. The architecture of SQLIA provides vulnerability model, threat model and attack model for ontology creation which improves data pre-processing. Figure 1 proposes the learning model for the system to learn the semantics and grammar of the SQL Injection attacks.

The projected basis can be used to gather the learning frameworks concerning the threats that mishandle the negatives to tamper with the security, and undertake conceivable mischiefs of the security agendas. The queries, in reference to the vulnerabilities, can be applied to differentiate the security defects in the web application, which allow the attackers to attack. The potential outcomes are cause by the effects of the attacks, aversion methods and relief from the projected attacks, which have to be recovered. The proposed model detects web vulnerability for SQL Injection attacks based on web protocols (http/https). The model predicts the SQL Injection attacks on websites on http (insecure) protocol for the websites which are hosted on cloud. Cloud Service Providers like AWS, Azure have their own security system which prevents their network, while our system works on the application layer.

3.1 Operating Methodology

The projected operating method includes four fundamental stages, which have been illustrated below.

Data Collection To begin with the segment of the projected system is the data collection, whereby the capability of the section is to collected the required information into attacks, avoidance, vulnerability and the countermeasures from various sources such as the Open Web Application Security Project (OWASP) to effectively collect the threat information, National Vulnerability Database (NVD) that is a mistake for the standardized vulnerabilities, including the Common Vulnerability Scoring System (CVSS) that represents the scoring aspect that processes the vulnerability scores.

Table 1 Contrast between different SQLIA discovery and deterrence approaches/methods

Technique/ approach	SQL attack detection	SQL attack prevention	Additional infrastructure	Drawbacks
Hybrid approach	Yes	Yes	N/A	Does not operate for the zero-day exploit
SQLIMW	Yes	Yes	Middleware	Operate for the sign-in application
Runtime monitors for tautology	Yes	Notification	Software repository used to safeguard source codes, paths and critical variables	For Java applications, detects tautology attacks only
Attribute removal (SQL query checker)	Yes	Yes	Developer learning	Probabilities of false positives dependent of information in the symbol tables
CANDID	No	Yes	N/A	Performance problems that have to be mitigated
Obfuscation	Yes	Report Generation	N/A	Renders inefficient for the SQL queries that are determined at the runtime in the Java servlet
Query tokenization	Yes	Blocked	N/A	Based on length factor only
Hidden Web crawlers	Yes	Error messages	Additional hardware required to store authorization record	Require more mechanism to deal with the SQL injection problems
SQL DOM	Yes	Yes	Developer learning	More runtime, cannot determine the stored process form of injection
AMNESIA	Yes	Yes	N/A	Operates for the JSP-based application
Secure prepared statements	Yes	Yes	N/A	Renders vulnerable to the iterative statement
SCC model	Yes	Alert trigger	Minimal user interaction required	Operate efficiently for the Union and the obfuscated injection
WAVES	Yes	Report generation	N/A	Cannot detect all vulnerabilities
Instruction-set Randomization (JDBC checker)	Yes	Blocked query, additional source code suggested	Proxy, developer knowledge and key organization	Renders ineffective when the random values are revealed. It does not operate for the illegitimate or illogical obfuscated queries
IDS	Yes	Report generation	Training set	A lot of false negatives and positives as for poor training sets

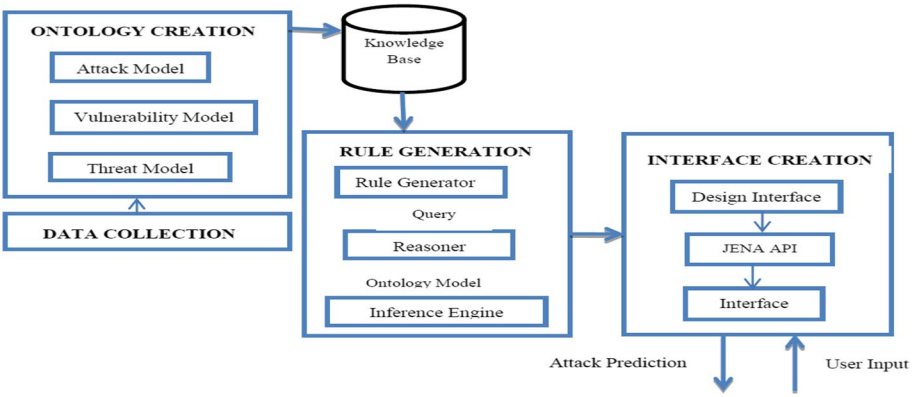

Fig. 1 Proposed system architecture of SQLIAO

Ontology Formation In this section, there exist three ontological frameworks which include the threat model, attack demonstration and weakness demonstration. There is reliance to links between these frameworks used to focus on the vulnerabilities.

Model-Threat The attack ontology demonstration provides the basic used to create the defenceless ontology displays and it mostly fundamental to determine the attack conditions used to anticipate the vulnerability conditions. In this condition ontological framework, the web application-related attacks are illustrated. The category threats are the extreme category of a certain web application attack are include the passive (dynamic) and the active (static) attacks, The framework includes the additional details for Data Disclosure, Eavesdropping, Side-channel Attacks, Malformed Inputs, Code Injections. Figure 2 descries the connection of various ideas and dangers.

Model-Vulnerability Vulnerability indication provides the foundation for creating a threat ontology framework that is vital in the process of focussing on the potential vulnerable conditions. In the vulnerability ontology demonstration, the web applications and their vulnerabilities are shown. The category vulnerabilities are the majors segment of a certain web application attacks. The attacks are the Cross Site Request Forgery (CSRF), Cross Site Scripting (XSS), Content Spoofing, Insufficient Authentication, Insufficient Authorization, SQL Injection and Brute Force and so on.

Figure 3shows the interlinking of the attacks and the ideas. The attacks displaying includes more ideas.

Model-Attack The vulnerability demonstration shows less fundamental security wordings. It includes various ideas such as the web application vulnerabilities and threats that are used to foresee the vulnerabilities in the web applications. The disadvantages of the web applications are considered to escape clauses in the performance of attacks, and resources which are affected by the attacks. The influence of the web application and the applicable counter measures are used to moderate the vulnerabilities and aversion techniques used to evade the vulnerabilities.

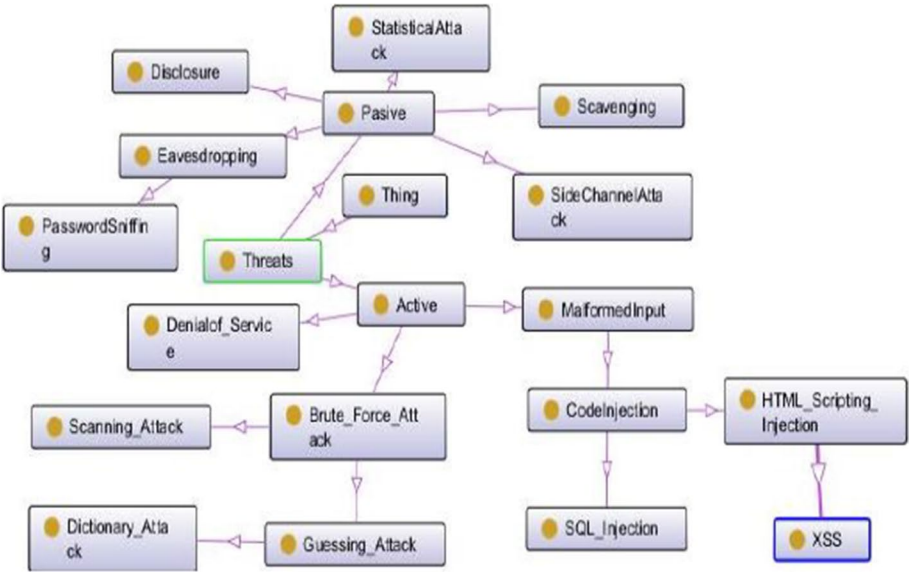

Fig. 2 Threats and their concepts

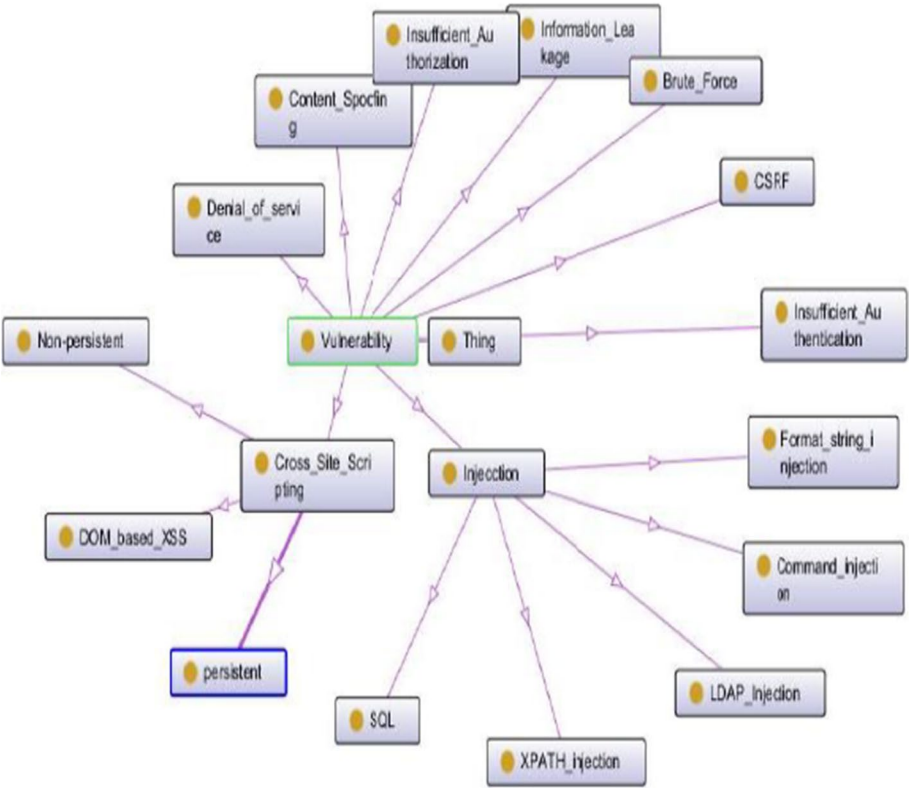

Fig. 3 Vulnerabilities and their concepts

In the Web, the application attack is significantly developing whereby the model is equipped with the capability to be reutilized any moment. The major ideas and categories of the ontology framework in the Attack category with protest properties is affectedTo, avoidedBy and has Threats that are featured by the ideas/categories threats, assets, countermeasures, impacts and the prevention techniques shows in Fig. 4.

The XSS threat is a sub-category threat category, which is initiated by the XSS vulnerability in the Web application. This threat uses the detached and dynamic threat, mishandles by the XSS defencelessness, client browser, and the aspect that brings about the page divert and adjustment, which is relieved through the counter measure using the HTTP treat hail avoided using the counter-active action methods. The model focusses on the SQLI attacks on websites. The proposed model has the ability to track vulnerabilities on any given day or traffic conditions.

3.2 Generation of Instruction

The following sector includes the formation of the expected instructions that typically include the advances illustrated below:

1. Weakness ontology put aside in the RDF tuples and the ontology framework selected in the surmising procedure.
2. The obtained induced data is constructed based on certain run syntax and instruction formats.
3. Providing the forecast instructions based on querying the construed data. The queries in 3-tuple is used to obtain that outcomes based on ontology frameworks and principles.

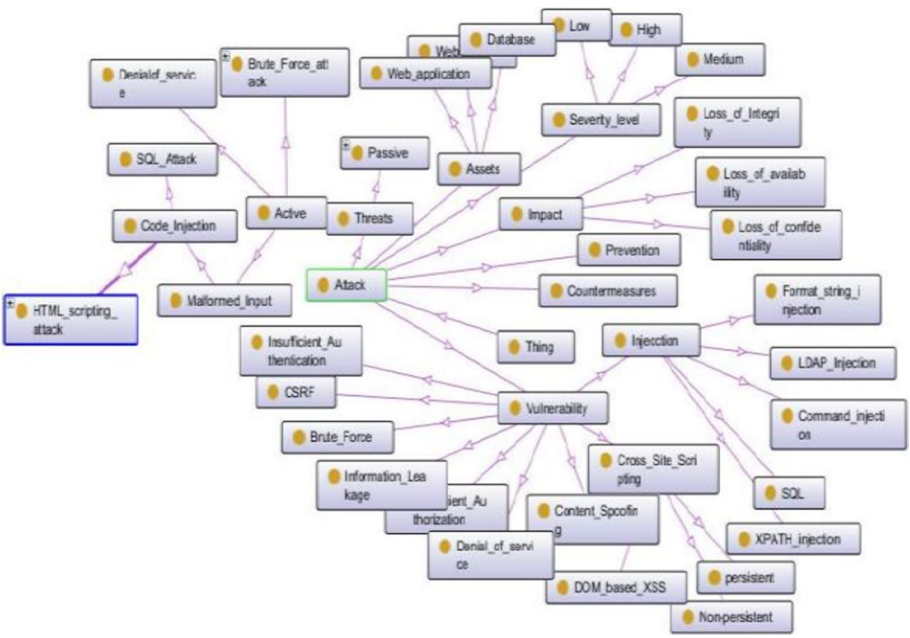

Fig. 4 Attack model and their concepts

Fig. 5 Rule template

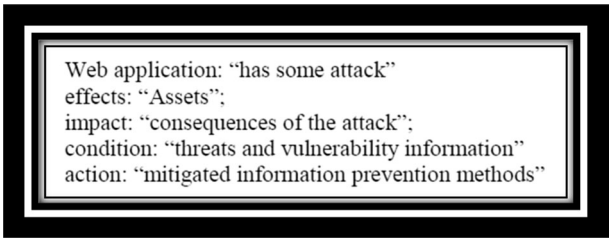

Fig. 6 Instances of rules for attack prediction

The process of derivation is utilized to deliver novel standards like the capability to attain the customers' requirements.

The instruction format can possibly be framed as shown in Fig. 5. It categorized the fundamental data streams based on the information provided by the customers, and the procedure of data used to foresee attacks. It critically demonstrates the attack information that includes the impact variables and the web applications that denote the merits influence by the threats. The impact variable depicts the attack outcomes and the condition variables utilized to focus on the threats in consideration to the contribution of the effected activities that include the countermeasures used in the attacks. This also includes the counter-active measure used in controlling the threats.

The rule layout additionally depicts the seriousness level of the web application attack. Keeping in mind the end goal to anticipate a particular attack circumstance the format is vital with the assistance of induced learning (attack and alleviation mechanisms). Each control occasion indicates the status under which the vulnerabilities mishandled attacks are projected, including the moves executed as provided in Fig. 6. Generally, each aspect control the condition variable, which is composed of the incentives used to focus on certain attacks from the customers' input, control activities and the related alleviations.

Inference Procedure Considering the data base and the present link between the ideas makes it possible to create new arrangements. This summation process is used to formulate novel declarations. The prescient surmising guidelines are provided in Fig. 7.

Decide one expresses that how the web application is powerless and lead 2 speaks to a general scenario of attack expectation, moderation and counteractive action. Run 3 denote the characterization of attack in light of their seriousness stage. The Semantic Rule Language (SWRL) is used to indicate the standards applicable in the process of

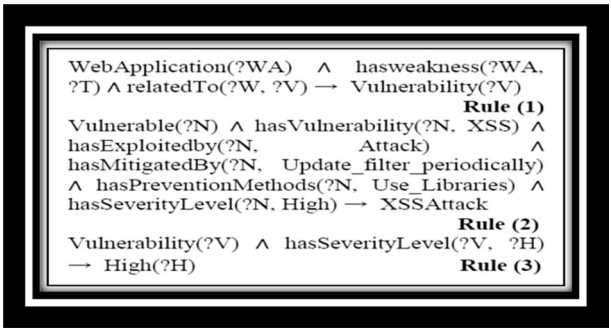

Fig. 7 Prediction rules based on vulnerabilities

establishing the vulnerabilities and the Web applications initiated by the attacks. Based on the surmising process, information is accomplished; in this manner attack is forecast is performed viably.

Interface Formation The final section includes the projected model of interface creation.

Design Edge Considering the overall beans status and java, the interface is purpose for the process of connectivity. Jena API is used to initiate the ontology framework, application interface and standards.

3.3 Functionality

The interface is intended to help easy to understand as portrayed in Fig. 8. It comprises of two significant segments: The first segment includes obtaining the contributions from the customers while the second segment includes the showcasing of data. The predictions for SQLIA scenario is discussed below:

Fig. 8 Application interface

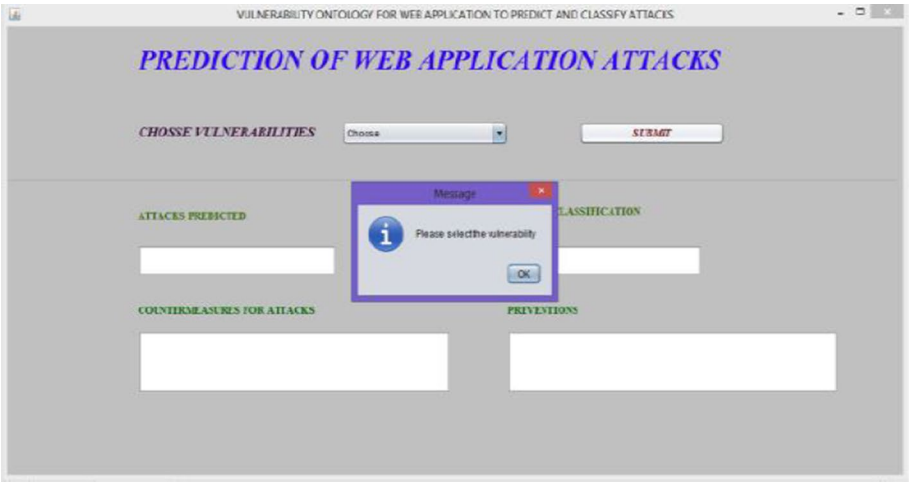

Fig. 9 Application interface with alert

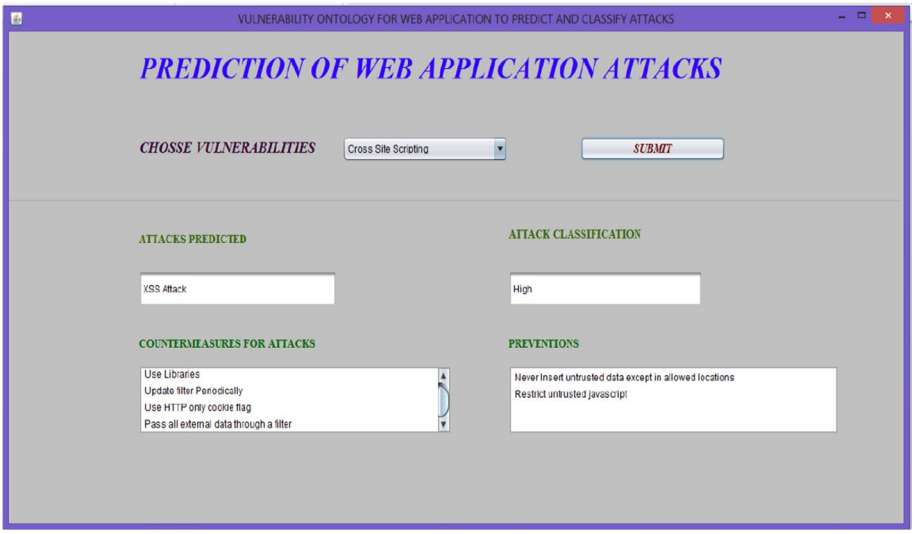

Fig. 10 Predict and classify attacks

The initial segment is utilized to obtain the client input: the weaknesses are selected by the user from the run-down question and box. The ontology demonstration, SPARQL, tenets inquiries are used to foresee the possible attacks. The alarming messages appear in Fig. 9 with some attacks.

The second segment is used to focus on the group threat in reference to the contributions posed by the client as shown in Fig. 10. These countermeasures are projected to deal with the vulnerabilities, which initiate the implementation of the preventive measures used to safeguard the Web application attacks.

4 Experimental Evaluation

Considering the final goal of this evaluation to enhance the assessment of the projected model, the Web application was initiated with the objective of dealing with the attacks and executing the counter-measures over the projected attacks. The Web applications were counterchecked to show some vulnerability. The retrieved information was evaluated based on parameters such as the forecast rate, actuality of the retrieved data, accuracy reviews and F-measures.

These are evaluated through the application of:

$$\text{Accuracy} = \text{Correct}/(\text{Correct} + \text{Wrong})$$
$$\text{Review} = \text{Correct}/(\text{Correct} + \text{Missed})$$

where,

- Correct: The amount of projected records is retrieved from the additions of insignificant data based on the methodology and added human.
- Wrong: The amount of projected records is retrieved from the addition of the unessential data based on the methodology, but not through human.
- Missed: The amount of projected records are retrieved from the addition of a significant amount of unessential information through human, but not based on the framework.

$$F\text{-}Measure = (2 * (Precision * Recall))/Precision + Recall \tag{2}$$

The projected rate of the projected methodology structured for the vulnerability is indicated in Fig. 11 and denotes that every threat is projected to the high. This enables

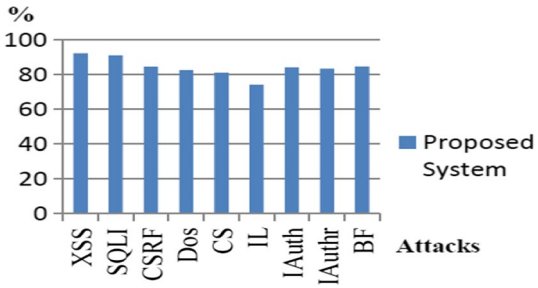

Fig. 11 Prediction rate of the proposed system

Table 2 Comparison of prediction rate

Web application attacks	Proposed system	Existing system
Cross Site Scripting (XSS)	92.3	86.9
SQL Injection (SQLI)	91.05	87.09
Denial of services (Dos)	84.56	63.04
Cross Site Request Forgery (CSRF)	84.56	63.04
Content Spoofing (CS)	82.57	72.43
Information Leakage (IL)	74.09	70.08
Insufficient Authentication (IAuth)	84.23	66.89
Insufficient Authorization (lAuthi)	83.45	64.23
Bmte Force (BF)	84.67	60.56

the aggressors to effectively comprehend the possible outcomes of the threats based on Web application that has been formed.

The forecast rate of the proposed framework comes about are analysed based on the security ontology structure shown in Table 2. From the Table 2 unmistakably the proposed framework anticipate more attacks with the assistance of the deduction procedure than the current framework.

4.1 Discussion

The attack order is additionally contrasted and the current framework. The Fig. 12 demonstrates the outline looking at the level of attack grouping. The capacity of attack arrangement of existing framework is low.

The exploratory outcomes demonstrate that the expectation capacity and attack characterization rate of the proposed framework is fundamentally superiorto the current framework. The framework effectively anticipatesthe Web application threats; typical projection rate, which is of high contrast based on the available framework.

Moreover, the proposed framework attack order rate is likewise high contrasted with the existing framework. This is on the grounds that the framework is equipped for foresee advanced attacks viably by examining the vulnerability that may abuse the attacks and the threat that was utilized.

A deduction procedure is to procure new learning and standards, used to foresee the most extreme attacks.

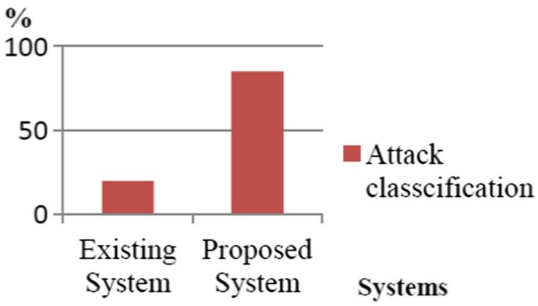

Fig. 12 Comparison of attack classifications

5 Conclusion

The endurance shows that e-business and data sharing is significantly advancing, which means that the security of users is vital. Based on Web application programming, securing clients' data is consistently becoming a significant objective of mitigating adventure threats. The analysers and designers of Web applications do not have the access to the data concerning the vulnerabilities, attacks and threats, which makes them to unpin uncertain applications. Various innovations have been accessed to control this potential issue. However, they are insufficient to provide complete security responses used to establish secure Web applications. The ontology-based frameworks are used to foresee and categorize the Web application vulnerabilities. The projected model effectively dissected the vulnerabilities and threats, which might mishandle the Web application attacks. Ontology showcases the weaknesses, threat and rules that project sophisticated attacks more effectively. The process of deduction motor, considering the Web application vulnerability information, assures the rundown of the potential attacks. These attacks have been designed considering the degree of seriousness of these attacks in accordance to the security intentions. The projected model also offers recommendations to effectively alleviate counteractive actions for the attacks. The retrieved data is significant for the analysers and designers to control the attacks, which means that the secure application process is well-planned. In future, the projected ontology framework can be reutilized to identify the Web application vulnerability in its testing level. The projected model is vital to track the SQLI Attacks on websites.

Compliance with Ethical Standards

Conflict of interest The authors have declared to have no confliect of interest.

Human and Animal Rights This research includes no studies involving animals or humans controlled by the authors.

References

1. Azfar, A., Choo, K. R., & Liu, L. (2019). Forensic taxonomy of android productivity apps. *Multimedia Tools & Applications, 76*, 3313–3341.
2. Nguyen-Tuong, A., Guarnieri, S., Greene, D., Shirley, J., & Evans, D. (2003). Automatically hardening web applications using precise tainting. In *Security & privacy in the age of S. Christensen, A. Moller, & M. I. Schwartzbach. Precise analysis of string expressions. Static Analysis, Proceedings*, Vol. 2694, pp. 1–18.
3. Yeole, S., & Meshram, B. B. (2011). Analysis of different technique for detection of SQL injection. In *Proceedings of the International Conference & Workshop on Emerging Trends in Technology (ICWET '11)* (pp. 963–966). Mumbai: ACM.
4. Sharma, A., & Sheth, J. N. (2004). Web-based marketing: The coming revolution in marketing thought & strategy. *Journal of Business Research, 57*(7), 696–702.
5. Alserhani, F., Akhlaq, M., Awan, I., & Cullen, A. (2011). Event-based alert correlation system to detect SQLI activities. In *2011 IEEE international conference on Advanced Information Networking & Applications (AINA)* (pp. 175–182). IEEE.
6. Avireddy, S., et al. (2012). Random4: An application specific randomized encryption algorithm to prevent SQL injection. In *2012 IEEE 11th international conference on Trust, Security & Privacy in Computing & Communications (TrustCom)*. IEEE.

7. Pankaj, P., Nagle, M., & Pankaj, K. K. (2012). Prevention of buffer prevention of buffer f buffer overflow attack overflow attack overflow attack blocker blocker blocker using IDS. *International Journal of Computer Science & Network (IJCSN)*, *1*(5). www.ijcsn.org. ISSN 2277-5420.
8. Benedikt, M., Freire, J., & Godefroid, P. (2002). VeriWeb: Automatically testing dynamic web sites. In *Proceedings of 11th International World Wide Web Conference (WWW'2002)*. Citeseer.
9. Bertino, E., Kamra, A., & Early, J. (2007). Profiling database application to detect SQL injection attacks. In *IEEE International Performance, Computing, & Communications Conference, 2007. IPCCC 2007* (pp. 449–548). IEEE.
10. Buehrer, G., Weide, B. W., & Sivilotti, P. A. G. (2005). Using parse tree validation to prevent SQL injection attacks. In *Proceedings of the 5th international workshop on Software engineering & middleware*. ACM.
11. Anley, C. (2002). *Advanced SQL injection in SQL server applications*. White Paper Next Generation Security Software Ltd. 2002, 9. *Oracle SQL injection in web applications*. Red-Database-Security GmbH Company, Germany, 2009. https://www.red-database-security.com/whitepaper/oracle_sql_injection_web.html. Accessed 16 March 2010.
12. Gould, C., Su, Z., & Devanbu, P. (2004). JDBC checker: A static analysis tool for SQL/JDBC applications. In *Proceedings of the 26th International Conference on Software Engineering (ICSE 2004) Formal Demos*, pp. 697–698.
13. Blanco, C., Sheras, J., Fernández-Medina, E., Valencia-García, R., & Toval, A. (2011). Basis for an integrated security ontology according to a systematic review of existing proposals. *Computer Standards & Interfaces, 33*(67), 372–388.
14. Pinzón, C. I., De Paz, J. F., Herrero, Á., Corchado, E., Bajo, J., & Corchado, J. M. (2013). idMAS-SQL: Intrusion detection based on MAS to detect & block SQL injection through data mining. *Information Sciences, 231*(10), 15–31.
15. Ezumalai, R., & G. Aghila. (2009). Combinatorial approach for preventing SQL injection attacks. In *IEEE International Advance Computing Conference. IACC 2009*. IEEE.
16. Abdoli, F. & Kahani, M. (2009). Ontology-based distributed intrusion detection system. In *Proceedings of the 14th International CSI Computer Conference*.
17. Valeur, F., Mutz, D., & Vigna, G. (2005). A learning-based approach to the detection of SQL attacks. In *Detection of intrusions and malware, and vulnerability assessment, Proceedings*, Vol. 3548, pp. 123–140.
18. Valeur, F., Mutz, D., Vigna, G. (2013). A learning-based approach to the detection of SQL Attacks. In *Conference on detection of intrusions & malware and vulnerability ass* (Vol. 55, No. 10, , pp. 1767–1780). Elsevier.
19. Shar, L. K., & Tan, H. B. K. (2013). Predicting SQL injection & cross site scripting vulnerabilities through mining input sanitization patterns. *Information & Software Technology, 55*(10), 1767–1780.
20. Cova, M., Felmetsger, V., & Vigna, G. (2007). Vulnerability analysis of web applications. In L. Baresi & E. Dinitto (Eds.), *Testing & analysis of web services*. Berlin: Springer.
21. El-Moussaid, N. E. B., & Toumanari, A. (2014). Web application attacks detection: A survey & classification. *International Journal of Computer Applications, 103*(12), 1–6.
22. Bisht, P., Madhusudan, P., & Venkatakrishnan, V. N. (2010). CANDID: Dynamic candidate evaluations for automatic prevention of SQL injection attacks. *ACM Transaction on Information System Security, 13*, 14.
23. Pietraszek, T., & Berghe, C. V. (2006). Defending against injection attacks through context-sensitive string evaluation. *Recent Advances in Intrusion Detection, 3858*, 124–145.
24. Haldar, V., Chandra, D., & Franz, M. (2005). Dynamic taint propagation for Java. In *Proceedings 21st Annual Computer Security Applications Conference*.
25. García, V. H., Monroy, R., Quintana, M. (2006). Web attack detection using ID3. In *workshop International Federation for Information Processing* Santiago, Chile, pp. 323–332.
26. Chung, Y.-C., Ming-Chuan, Wu, Chen, Y.-C., & Chang, W.-K. (2012). A Hot Query Bank approach to improve detection performance against SQL injection attacks. *Computers & Security, 31*(2), 233–248.

27. Su, Z., & Wassermann, G. (2006). The essence of command injection attacks in web applications. *ACM SIGPLAN Notices., 41*, 372–382.

Dr. K. Naveen Durai Professor (Assistant) and Training Head in Department of Computer Science and Engineering, Sri Eshwar College of Engineering, Coimbatore, Tamilnadu, India. He has received his Ph.D. in Information and Communication Engineering from GCT Coimbatore under, Anna University, Chennai. His research areas include Web security, Java Full Stack, Spring MVC, SpringBoot, Hibernate, SQL injection, Django framework.

Dr. R. Subha had obtained B.E in Computer Science and Engineering from Periyar university in 2002 and M.E. in Software Engineering from Anna University, Chennai in 2006. Also, she had received her Ph.D. from Anna University, Chennai in 2014. She has more than 15 years of experience in teaching and currently she is working as Professor and Head in the Department of Computer Science and Engineering at Sri Eshwar College of Engineering, Coimbatore. Her research interests include Software Engineering, Theory of Computation and Data Mining.

Dr. Anandakumar Haldorai Professor (Associate) and Research Head in Department of Computer Science and Engineering, Sri Eshwar College of Engineering, Coimbatore, Tamilnadu, India. He has received his Master's in Software Engineering from PSG College of Technology, Coimbatore and Ph.D. in Information and Communication Engineering from PSG College of Technology under, Anna University, Chennai. His research areas include Big Data, Cognitive Radio Networks, Mobile Communications and Networking Protocols. He has authored more than 82 research papers in reputed International Journals and IEEE conferences. He has authored 7 books and many book chapters with reputed publishers such as Springer and IGI. He is editor of Inderscience IJISC and served as a reviewer for IEEE, IET, Springer, Inderscience and Elsevier journals. He is also the guest editor of many journals with Elsevier, Springer, Inderscience, etc. He has been the General Chair, Session Chair, and Panelist in several conferences. He is senior member of IEEE, IET, ACM and Fellow member of EAI research group.