

Taxonomy and Characterization of Structured Query Language Injection Attacks for Predictive Analytics

Idowu, Sunday A.¹; Awodele, Oludele²; Kuyoro, Shade O.³;
Akinsola, Jide E. T.⁴

Department of Computer Science, Babcock University, Ilisan-Remo, Ogun State, Nigeria^{1,2,3,4}

saidowu07@gmail.com¹; dealealways@yahoo.com²; afolashadeng@gmail.com³; akinsolajet@gmail.com⁴

ABSTRACT

Structured Query Language Injection Attacks (SQLIA) from attackers' exploit have progressively increased the danger of cyber attacks to steal confidential information from the database of vulnerable web application front-ends with potentially damaging security consequences. Several detection methods have been identified for effective mitigation of SQLIA. These detection methods are classification by nature of defense, classification by detection principle, classification by analysis method, classification by detection time, classification by detection location, classification by response, and classification by implementation. The relationship between different attack signatures and patterns were studied to create a taxonomy of SQLIA. This study characterized the SQLIA into eighteen classes for predictive analytics to detect and prevent unauthorized access to the database for stealing confidential information. The analysis of detection and prevention of SQLIA through the taxonomy and characterization into types and classes with order of importance helps in the implementation of various attack mechanism. Thus, reducing the possibilities of new injection points into the database by the attackers.

Keywords: Database, Predictive Analytics, Injection Attack, SQLIA, Web application

1. INTRODUCTION

Structured Query Language Injection Attack (SQLIA) is referred to as forceful code manipulation insertion attack targeted against database through vulnerable web

applications, which is a source of attack into the database. Web applications contain data that is of utmost significance and importance such as bank account information, credit cards numbers, personal email, medical history, personally identifiable information and classified information, that are stored in the back-end database. These data are highly confidential. The compromise of these data accessed through web applications is a source of injection attack into the database.

The resultant effects of the attacks are visible with injection attacks such as SQLIA and Cross Site Scripting (XSS) which exposes confidential information. Data breaches through SQL injection and cyber attacks in 2018 reveal a figure of 17,273,571 only on records leakage with Cyber attacks on the rise [1] [2]. Likewise, code injection attacks such as SQLIA account for 40.8% and cross site scripting attacks account for 11.3% of total attacks in 2018. According to Open Web Application Security Project (OWASP), Injection vulnerabilities for example SQL injection and XSS, rank among the top two out of the top 10 from the analysis conducted by OWASP [3].

The compromise of Confidentiality, Integrity and Availability (CIA) shows that database is attacked. Therefore, once an unauthorized third party have a way of entering unlawfully into the body of related information called database, that signifies failure of the component of deuce-ace of CIA called Confidentiality. When stored information has been altered, that signifies compromise of the component of the deuce-ace of CIA called Integrity. When there is total data vilification, it signifies compromise

of the component of the common chord of CIA security framework called Availability [4].

The Enterprise Web Applications (EWA) are made up of multi-tier design such as Presentation, Application and Data tiers. The presentation tier is the HTTP web interface, the application tier implements the software functionality, and the data tier keeps data structured and answers to requests from the application tier [5]. The most critical of the three tiers is the data tier which is concerned with the backend database.

The back-end information storage is crucial for storing Big Data's huge volume. Internet challenges have increased due to adoption of Internet of Things (IoT), cloud-hosted services and online applications. These applications are intricately web-based. Cloud-based applications offer a lot of security threats [6] [7]. To mitigate attack to this backend databases require predictive analytics.

Predictive analytics utilizes a range of approaches including data mining, machine learning, modelling and simulation as well as artificial intelligence to evaluate existing data for future predictions. Thus, injection attacks prediction requires effective taxonomy which consequently helps in characterization of SQLIA. Hence, attack intent and injection mechanisms as the two most important characteristics of SQLIA for efficient detection and prevention of various forms of injection attacks can be Addressed.

2. SQL INJECTION ATTACK TECHNIQUES

The SQLIA methods are focused on the attack mechanism, which hackers can attempt to carry out the hacking. These seven majorly common SQLIA techniques (mechanisms) are: 1. Tautology 2. System Stored Procedure 3. Inference 4. Illegal / Logically Incorrect Query 5. Alternate Encodings 6. Union Query 7. Piggy-backed Query. These mechanisms require effective taxonomical formulation in order to handle the various forms of SQLIA efficiently. Therefore, classification based on these common seven mechanisms only can create loopholes for the intruders to gain access to database schema, evading detection and hence circumvent confidentiality, integrity and availability of the sensitive information in the database. Thus, the need for new taxonomy as shown in Figure 1.

2.1 Tautology

This is concerned with one or more conditional statements used to inject code so as to always validate the true

statements. This method occurs when the input data to the database is not checked. An instance of such a vibrant SQL statement is the code given thus; query= "SELECT details FROM customer WHERE name=' name' AND pwd=' pwd;'. attackers may use tautologies to make use of this software balance by providing an entry parameter number(x' OR 1='1') with the significance. An intruder could enter customer data without a relevant consideration because the situation of the WHERE clause becomes the same (which makes the system validates the outcome to be true and terminates the remaining query using (--);. (WHERE='x' Or 1='1'--);. Example of Tautology query attack: SELECT * FROM employee WHERE name = ' ' or 1=1 -- ' AND password = '12345';

2.2 Piggy-Backed

The hackers will insert additional queries to be performed by the database in this scenario to extract, input or alter information, service performance denial or carry out commands from distance [8]. Attackers do not attempt to change the initial request in that situation. They actually attempt to attach an additional and different entry to the initial request using personal SQL-based phrases such as OR, AND, INSERT, UPDATE, DROP or DELETE to permit various SQL queries to the database [9]. Example of Piggy-backed query attack: SELECT * FROM employee WHERE name = 'guest' and password = '1234'; DROP TABLE employee; -- ;

2.3 Alternate Encoding

Hackers mainly aim to avoid identification when using this technique. In fact, this sort of attack is used to encode the attack strings to avoid the filtering from the programmer (e.g. by using hexadecimal, ASCII and Unicode character set). In reality, additional encodings are generally applied in relation to other attack methods and target dissimilar application levels [10]. The usage of quote (') in the SQL statement declaration that can be used in the creation of different form of malicious database query request is prohibited for most of SQL injection mechanism that uses filters.. In place of a single quote which can easily be detected as bad character, for instance, the intruder uses char (44). This attacks combines char () function and ASCII hexadecimal encryption. Real characters(s) are returned when char () function is used to convert to hexadecimal character(s) encoding equivalent. Example of alternate encoding query attack: SELECT accounts FROM login WHERE username=" AND password=0; exec (char (0x736875746466j776e))

2.4 Illegal / Logical Incorrect Query

In that assault, attacker attempts injecting declarations which cause the application servers to return a syntax error page to identify injectable parameters, it applies fingerprinting and extract data from the web application's backend databases [9]. In reality, error page gives hackers information about few details of tables' name in the databases, such as instances, or discloses vulnerable / injectable parameters for an intruder and such details will be used in carrying out the next attack phase [11]. Example of Illegal / Logical Incorrect query attack: `SELECT * FROM employee WHERE name = ' ' UNION SELECT SUM(username) from users -- ' and password= ' ' ;`

2.5 Union Query

In this attack technique, the malicious query is added to the initial request via the UNION keyword to obtain information concerning additional database tables. An intruder can pull out column data or type of data details from this sort of attack [12]. By rule, most of the SQL conforming databases, including SQL Server stores metadata with sysobject numbers, syscolumns, sysindexes, and so on, in a set of system tables. This allows a hacker to use the information about the database table to identify schema information for a database in order to help hackers to launch assaults to the database further. Example of Union query attack: `SELECT emp_id FROM employee WHERE name = ' ' UNION SELECT cardNo FROM creditCard WHERE accNo = 10032 -- AND password= ' ' ;`

2.6 Stored Procedures

This method uses vicious SQL codes to execute integrated built-in functions, which further escalate privilege, ensures service denial or to execute remote controls. Indeed, most database providers develop database solutions with standard stored procedures and features to enhance the database functionality and brings interactivity with the operating system. Therefore SQLIAs may be created to perform stored procedures on this particular database once an attacker has known the backend database [12] [13]. Example of stored procedure query attack: `CREATE PROCEDURE DBO @userName varchar2, @pass varchar2, AS EXEC ("SELECT * FROM user WHERE id= ' "+@userName+" and password= ' "+@pass+""); GO`

2.7 Inference

An intruder draws logical conclusion from a response to a right / wrong enquiry about database server answer. Two Blind injection and time injection input methods are used to

launch this attack [14]. In-Blind injection, hackers obtain database information by submitting a server's true / false questions and the answers from this page gives leading information that will be exploited further. If the response is accurate, the request is correct and if the response is wrong, then an error will be triggered. An intruder can therefore obtain implicit response from the database [15]. Part of Inference attack can be classified into Blind SQL injection and Timing Attack. Example of inference (blind) SQL injection attack: `SELECT * FROM emp_name, emp_address, gender, from employee where 1=0; drop employee`

3. TAXONOMY OF SQLIA FOR PREDICTIVE ANALYTICS

The taxonomy of SQL injection attacks are broadly classified in form of Classical, Inference which is otherwise called Blind, Order-wise, and Compounded. Classical SQLIA involves attacks against database (that is, DBMS specific) and Illegal / Logical / Incorrect Queries. Figure 1 shows the taxonomy for the various SQLIA which has been characterized into eighteen (18) malicious attacks with the nineteen characterization as benign as shown in Table 1. Inference SQLIA has been classified into standard attack which involves the usage of TRUE/ FALSE and Timing which is concerned with queries that uses IF/THEN. Order-wise SQLIA classification is divided into First Order, second order and literals. Compounded SQLIA involves mechanism that utilizes fast fluxing. The classification was used to understudy the various attack approaches in order to have an effective SQL injection attacks mitigation system for all the identified taxonomy. Attacks against database is the most prominent because confidential and financial information are stored in the database.

4. CATEGORIZATION OF SQLIA DETECTION METHODS

Structured Query Language (SQL) security threats or injection attacks can be mitigated using seven detection classification methods such as classification by nature of defense, classification by detection principle, classification by analysis method, classification by detection time, classification by detection location, classification by response, and classification by implementation.

Classification of SQLIA detection by analysis method and detection principle method are the two most common SQL Injection detection methods. Detection principle method has been widely utilized for predictive analytics which is one of

the mechanisms to mitigate security breaches such as SQLIA. Category for classification by detection principle include grammar based violation, signature based detection, tainted data flow and anomaly based detection. Anomaly based detection category is sub-divided into learning based and program based while tainted data flow detection category is sub-divided into positive tainting and negative tainting [16]. Furthermore, signature based detection category is sub-divided into input signature and output signature. The learning based sub-category of the anomaly-based detection category is implemented using machine learning approach. The SQL attacks detection methods are categorized as shown in Figure 2.

Category for classification by analysis method include secure programming, static analysis, dynamic analysis, hybrid analysis, black box testing and white box testing. Secure programming is designed to prevent hacking, threats and security vulnerabilities to applications while defensive coding handles unexpected conditions [17]. Static analysis is used to check the dynamically created SQL query for static accuracy [18] [19]. Dynamic analysis deals with program testing and evaluation in relation to real-time data execution. Dynamic analyzes is the mechanism by which a system is checked and analyzed in real time. Query formed is used as the input in dynamic analysis phase at the runtime and Dynamic Query Model is formed for each query. In Dynamic Query Model sequence of queries are tokenized to form SQL tokens for example characters, identifiers, special strings, keywords and numbers [20]. Hybrid analysis combines the mechanisms of both static analysis and dynamic analysis. According to [21], black-Box testing refers to a software testing method in which safety checks, defenses and application design are tested from the outside, with little or no prior information about the internal functioning of the application being tested. The black-box experiments basically take a similar approach to that of a true attacker. White box testing is a software check tool which can be used to verify the specification, verify the security features and reveal exploitable vulnerabilities whether the security implementation is functional [22].

The intention of SQLIA is to identify injectable websites, determining database schema, evading detection and circumvent confidentiality, integrity and availability.

5. CHARACTERIZATION OF SQLIA FOR PREDICTIVE ANALYTICS

Characterization of SQLIA into eighteen is essential to cater for all the attack mechanisms that can be instituted against web applications with the primary focus on stealing confidential information from the database.

The following are the characterized eighteen SQLIA types for effective SQLIA detection and prevention. Time-based error, Database Fingerprinting, Stored procedure, Buffer Overflow, Second Order, Deep Blind, Out of band, Alternate Encoding, Conditional Error, Union, Double blind, Conditional response, Illegal / Invalid / Logical Incorrect, Piggy Back, Error based (blind), Database Mapping, Literal, and Tautology with the non-malicious class referred to as Benign.

The bane of the problems for mitigating SQLIA has been availability of pattern driven dataset. The existing datasets are obsolete and the few ones available cannot be implemented in a cloud-based environment with big data framework for predictive analytics.

The HTTP dataset CSIC 2010 was motivated by the need to upgrade the obsolete DARPA KDD Cup in 1999. Due to the limitation of the few existing data sets relating to SQLIA field of research, in which the existing datasets were also made for competition as in ECML / PKDD, 2007 and HTTP dataset CSIC 2010. There is a need for learning data to train the resulting web application security classifier in real life scenarios. Reliable and pattern driven dataset has constituted a major challenge to SQLIA research and this has been addressed in this research through the following approaches:

- i. Development of expected legal and illegal input query signature string data as shown in Table 1.
- ii. Development of established SQLIA signatures which the background knowledge of SQLIA types provides in the taxonomy Figure 1 as shown in Table 1 and Table 2.
- iii. Development of pattern of SQL tokens (which comprises of query keywords, constants symbols and delimiter identifiers) as shown in Tables 1 and 2.

Order of importance is very essential in order to determine the class of SQLIA for effective detection and prevention for predictive analytics.

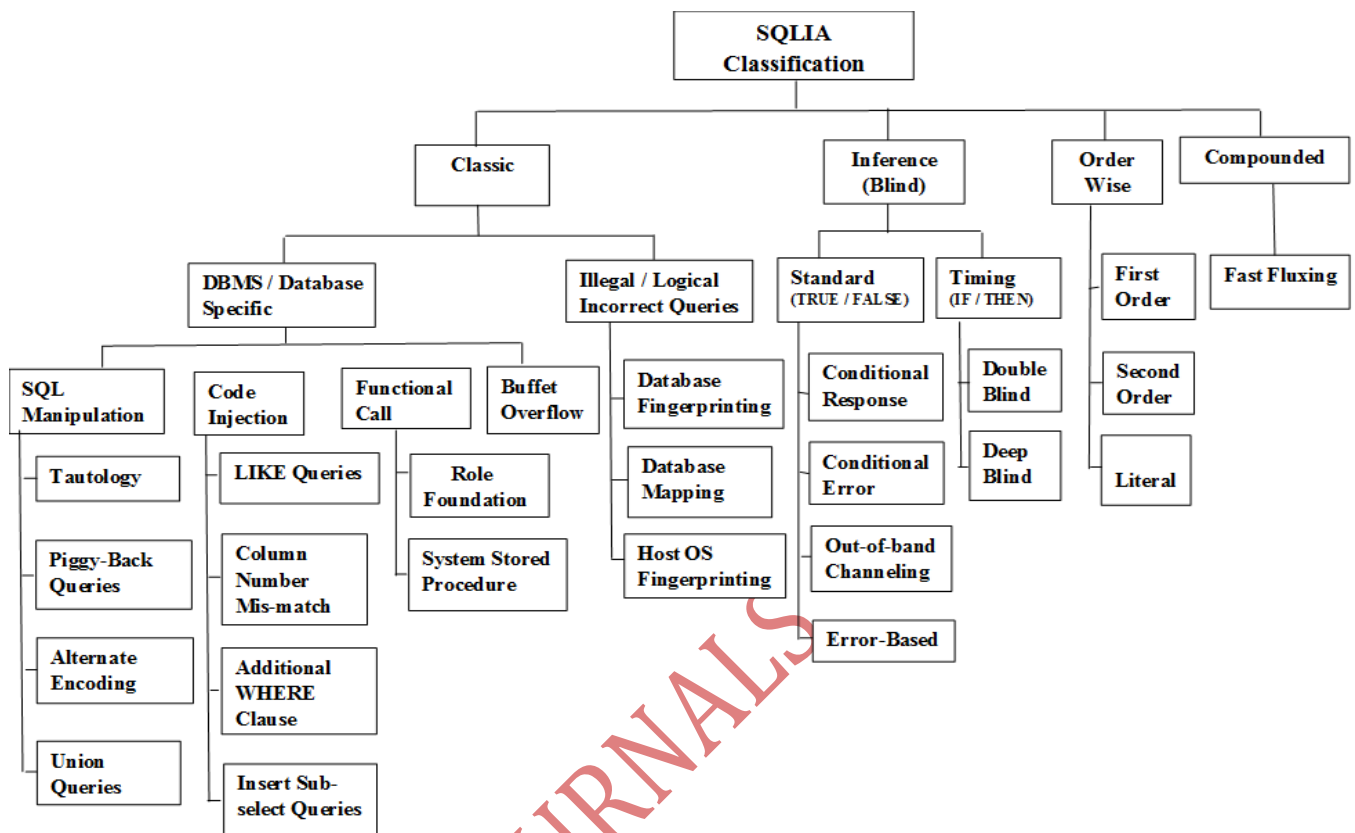


Fig 1: Taxonomy of SQLIA for Predictive Analytics

Table 1. SQLIA Type, Signature, Class and Order of Importance

Order of Importance	SQLIA Type	Signature	Class / Label
1	Time-based error	waitfor delay '00:00:10'--, sleep	6
2	Database Fingerprinting	all_tab_columns, DROP, @@version, version(), admin, administrator, SCHEMA(), ROW_COUNT().; @@GLOBAL.VERSION, CREATE, all_tables, TABLE members, tablename, insert, table_schema, Information_schema.tables	8
3	Stored procedure	Exec, xp_cmdshell, Attrib c:\test\ spuriousfile.vbs " +r Stored procedure keywords (SHUTDOWN, exec, xp_cmdshell(), sp_execwevtask())	5
4	Buffer Overflow	Printf, %s, %n, %d, %s%s%s	18
5	Second Order	"--, 'x'='x'--"	7
6	Deep Blind	TRUE, FALSE	16
7	Out of band	INJECTION, ASCII, OPENROWSET, LOADFILE, DECLARE	12

Order of Importance	SQLIA Type	Signature	Class / Label
8	Alternate Encoding	Hex values, Numeric values, exec (), Char (), ASCII (), BIN (), HEX (), UNHEX (), BASE64 (), DEC (), ROT13 (), MD5(), SHA1(), VARCHAR, SUBSTRING, CAST	15
9	Conditional Error	NULL,	11
10	union	UNION, UNION SELECT	3
11	Double blind	sa, 1/0, dbo, IF, <u>ELSE</u>	17
12	Conditional response	END, AND, OR	10
13	illegal / Invalid / Logical incorrect	CONVERT, GROUP BY, ORDERBY, ORDER BY, COMPUTE, invalid conversions, CONVERT (TYPE)), incorrect logics,	2
14	Piggy Back	;; ;-- ; --	4
15	Error based (blind)	TMP_SYS_TMP, HAVING	14
16	Database Mapping	Information_schema.columns	9
17	Literal	“ ”, In SQL statements, a string literal is a group of characters surrounded by quotes. All through strings NOTE: this means that none of the attack signatures in order of importance 1 to 16 does not exist in the query string. Once there is double quote in any string without the any of the symbols above from 1 to 16 it is a literal attack	13
18	Tautology	like, ‘, ` , 1=1,1<1, 1>1, ‘a’=‘a’, &&_1>1, ‘, = NOTE: 1=1 can be any number or digit, ‘a’=‘a’ can be any alphabet or character, 1<1 can be any number or digit, 1>1 1 can be any number or digit. So using RegEX will help in addressing these and ensuring that all scenarios are all captured	1
0	Benign	This does not contain any token for SQL injection attacks. For example: http://www.van-sant.si/components/com_akeeba/controllers/drive/auth/ or http://bmwusfactory.com/	0

Table 2. Query Signature and Attack Pattern Mapping for Dataset Generation

SQLIA Class	Attack Type	Query Signature	Attack Pattern
1	Tautology	SELECT * FROM tblUser where id=	_ 1=1#
2	illegal / Invalid / Logical incorrect	SELECT * FROM products WHERE category =	1; ' GROUP BY tablle (SELECT * FROM TMP_SYS_TMP) AS Int)
3	union	SELECT loginName, password FROM tblUser WHERE loginName=	Gifts' ' UNION SELECT username, password
4	piggy back	SELECT fieldlist FROM table WHERE field =	x' && ; -- 1=(SELECT * FROM tablename); --';
5	Store procedure	SELECT * FROM users WHERE email =	'1; DECLARE @result int; EXEC @result = xp_cmdshell 'dir *.exe';
6	time-based (blind)	SELECT * FROM tblUser where id=	1'; IF (1=2) WAITFOR DELAY '0:0:10'--
7	Second order	SELECT * FROM Product WHERE ID=	1; 'x'=x'--" OR Name : ' + (SELECT TOP 1 password FROM users) + ' Email : xx@xx.com
8	database fingerprinting	SELECT * FROM information_schema.columns WHERE table_name =	; OR SELECT * FROM information_schema.tables
9	database mapping	SELECT * FROM Product WHERE ID=	SELECT table_name, column_name FROM information_schema.columns WHERE table_name = 'tablename'
10	conditional response (blind)	SELECT email, passwd, login_id, full_name FROM table WHERE email =	'x' AND members.email IS NULL; --';
11	Conditional Error	SELECT * FROM user where id=	11223344) UNION SELECT NULL,NULL,NULL,NULL WHERE 1=2 -- OR ') OR 1=172
12	Out of band (blind)	SELECT name, description FROM products WHERE category =	1 LOAD_FILE("\\\\portal.burpcollaborator.net\ \a')
13	Literal	SELECT * FROM Product WHERE ID=	"1>2"; "joe"
14	error based (blind)	SELECT * FROM products WHERE category =	1; ' (SELECT * FROM TMP_SYS_TMP) AS Int
15	alternate encoding	SELECT * FROM tblUser where id=	1' SELECT CONCAT(CHAR(75),CHAR(76),CHAR(77))
16	deep blind	SELECT * FROM tblUser where id=	1; IF (1=1) SELECT 'true' ELSE SELECT 'false'

SQLIA Class	Attack Type	Query Signature	Attack Pattern
17	double blind	SELECT * FROM tblUser where id=	1; SELECT CASE WHEN (1=1) THEN 'A' ELSE 'B'END;
18	buffer overflow	SELECT * FROM tblUser where id=	'; --' printf (%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s)s (%s%s)s); OR ') OR 1=117

ijournals

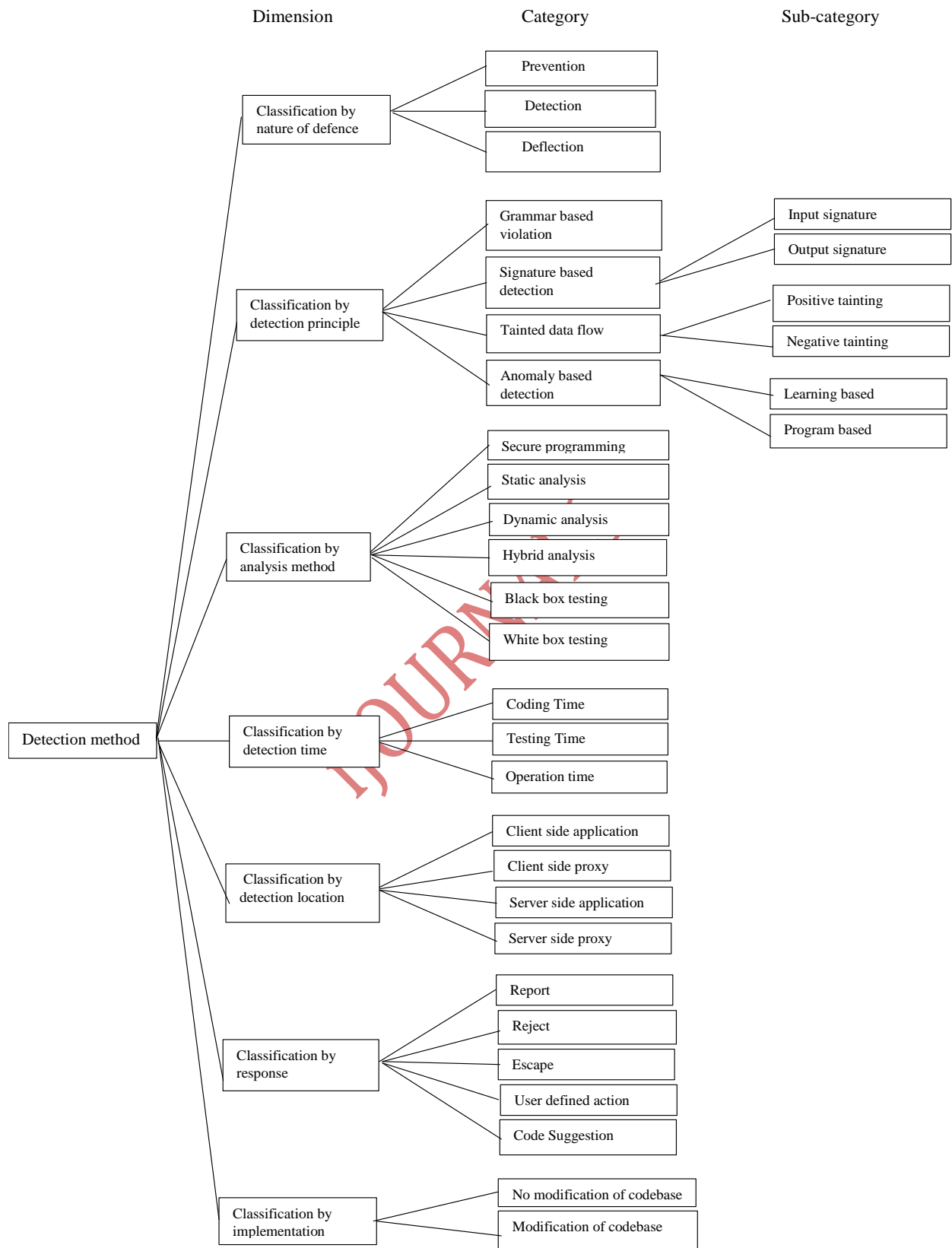


Fig 2: SQL Injection Detection Methods [16]

6. CONCLUSION

The main focus of SQL injection attack is to inject some malicious script through the web applications in order to gain an unauthorized access to the database to steal confidential information. Web applications are essentially vulnerable to such type of attacks because the user provides input through the URL, login and search from HTTP request. The request is passed to the database through the web application where the information is stored and this information gets stolen by the attacker because of the lack of validation at the input side. Several types of attack approaches are being employed by the attackers which have been characterized into eighteen in this study as Time-based error, Database Fingerprinting, Stored procedure, Buffer Overflow, Second Order, Deep Blind, Out of band, Alternate Encoding, Conditional Error, Union, Double blind, Conditional response, Illegal / Invalid / Logical Incorrect, Piggy Back, Error based (blind), Database Mapping, Literal, and Tautology for effective SQLIA detection and prevention.

The attack pattern and query signature can be combined to study the detection and prevention approaches for mitigating the characterized eighteen classes of SQLIA.

7. REFERENCES

- [1]. IT Governance, (2018) Infographic: List of data breaches in 2018. Available at: <https://www.itgovernance.co.uk/blog/infographic-list-of-data-breaches-in-2018>
- [2]. IT Governance, (2019).The 5 most common cyber attacks in 2019. Available at: <https://www.itgovernance.co.uk/blog/different-types-of-cyber-attacks>
- [3]. OWASP (2018). Open Web Application Security Project (OWASP), OWASP Top Ten Project. [http://www.owasp.org/index.php/Category:OWASP Top Ten Project](http://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project).
- [4]. Sheykhkanloo, N. N. (2017). A Learning-based Neural Network Model for the Detection and Classification of SQL Injection Attacks. International Journal of Cyber Warfare and Terrorism. Available at: <https://dl.acm.org/doi/10.4018/IJCWT.2017040102>
- [5]. Bogdan Carstoiu, Dorin Carstoiu, "Zatara, the Plug-in-able Eventually Consistent Distributed Database", AISS, Vol. 2, No. 3, pp. 56 ~ 67, 2010
- [6]. Uwagbole, S. O., Buchanan, W. J. & Fan, L. (2017). Applied Machine Learning Predictive Analytics to SQL Injection Attack Detection and Prevention. IFIP/IEEE IM 2017 Workshop: 3rd International Workshop on Security for Emerging Distributed Network Technologies. ISSN: 978-3-901882-89-0, 1087 - 1090 Available at: <http://dl.ifip.org/db/conf/im/im2017-ws3-dissect/177.pdf>
- [7]. Uwagbole, S. O. (2018). A Pattern-Driven Corpus to Predictive Analytics in Mitigating SQL Injection Attack. A PhD Thesis Submitted In Partial Fulfilment Of The Requirements Of Edinburgh Napier University, For The Award Of Doctor Of Philosophy In The Faculty Of Engineering, Computing & Creative Industries. Available at: <https://pdfs.semanticscholar.org/9ecc/19bdb16db16e195f2b7c4959a51fa781eb65.pdf>
- [8]. Halfond, W.G.J., J. Viegas, & A. Orso, A. (2006). A classification of SQL injection attacks and countermeasures. In Proceedings of the IEEE International Symposium on
- [9]. Khari, M. & Kumar, N. (2013). SQLIA Detection And Prevention Approaches A Survey. International Journal of Computer Science & Information Technology, 3(5)
- [10]. Kindy, D.A. & Pathan, A. S. K. (2013). A Detailed Survey on various aspects of SQL Injection in Web Applications: Vulnerabilities, Innovative Attacks and Remedies. International Journal of Communication Networks & Information Security, 5(2).
- [11]. Vinod, K. K & Jatin, D. D. (2013). Advanced Detecting and Defensive Coding Techniques to prevent SQLIAs in Web Applications A Survey. International Journal of Science and Modern Engineering (IJISME), 1(6), 26 - 31 Available at: <https://docplayer.net/2925830-Advanced-detecting-and-defensive-coding-techniques-to-prevent-SQLIAs-in-web-applications-a-survey.html>
- [12]. Roy, S., Singh, A.K. & Sairam, A. S. (2011). Detecting and Defeating SQL Injection Attacks. International Journal of Information and Electronics Engineering, 1(1).
- [13]. Srivastava, S., (2012). A Survey On: Attacks due to SQL injection and their prevention method for web application.

- [14]. Borade, M.R. & Deshpande, N.A. (2013). Extensive Review of SQLIA's Detection and Prevention Techniques. International Journal of Emerging Technology and Advanced Engineering, Vol. 3 No.10
- [15]. Khochare, N., Chalurkar, S., Kakade, S. & Meshram, B. B. (2011). Survey on SQL Injection attacks and their Countermeasures. International Journal of Computational Engineering & Management (IJCEM). 14, 111 – 114 Available at: https://www.ijcem.org/papers102011/ijcem_102011_19.pdf
- [16]. Shakya, A. & Aryal, D. (2011). A Taxonomy of SQL Injection Defense Techniques. School of Computing Blekinge Institute of Technology, Sweden. Master's Thesis Computer Science Thesis no: MCS-2011-46. Available at: <https://pdfs.semanticscholar.org/c073/d90d5f922321e80deb5993382848b1851584.pdf>
- [17]. Sharma, A. (2019). All About OWASP #1 - SQL Injection Attack. Available at: <https://www.c-sharpcorner.com/article/sound-programming-methodology-s-for-secure-programming/>
- [18]. Gould, C., Su, Z. and Devanbu, P. (2004). JDBC Checker: A Static Analysis Tool for SQL/JDBC Applications. In Proceedings of the 26th International Conference on Software Engineering (ICSE 04) – Formal Demos, 697–698
- [19]. Gould, C., Su, Z. and Devanbu, P. (2004). Static Checking of Dynamically Generated Queries in Database Applications. In Proceedings of the 26th International Conference on Software Engineering (ICSE 04), 645–654
- [20]. Nithya, V., Pandian, L. S. and Regan, R. (2013). The SQL Injection Attack Detection and Prevention by Classification and Analysis. Asian Journal of Information Technology, 12: 131-139. DOI: 10.36478/ajit.2013.131.139 URL: Available at: <http://medwelljournals.com/abstract/?doi=ajit.2013.131.139>
- [21]. Acunetix (2017). What is Black-box security testing. Available at: <https://www.acunetix.com/blog/articles/black-box-security-testing/>
- [22]. CISA (2015). White Box Testing. Available at: <https://www.us-cert.gov/bsi/articles/best-practices/white-box-testing/white-box-testing>