A Novel Intrusion Detection and Prevention Model for SQL Injection Attacks

Malik Rizwan Ali[1†] Muhammad Sheraz Arshad Malik[2††] Noureen Hameed[3†††] Faizan Tahir[4††††]

[2]Department of Information Technology, Government College University Faisalabad, Pakistan
[1, 3, 4]Department of Computer Sciences, Virtual University, Pakistan

Abstract
SQL Injection Attack (SQLIA) is a hot issue now a days in web applications and databases. SQL Injection ignores the authentication checks and affects the confidentiality of the database. SQLIA helps the invader to get the unauthorized access of the whole database and manipulate it. The existing tools and techniques like SQLRand, CANID, AMNESIA and SQL DOM mainly focus on providing access to the database only to the authenticated users. These tools do not provide complete prevention measures against the SQLIA. In this research, a novel Intrusion Detection and Prevention System (IDPMIA) is introduced. The proposed IDPMIA will detect the malicious queries before execution. Whenever an attacker tries to inject a suspicious query, it would immediately be recognized by IDPMIA and preventive measures will be taken. The proposed approach will be justified through case studies where multiple SQL Injection attacks will be simulated and results will be analyzed using the proposed model and existing state of the art techniques from literature.

Key words:
Intrusion Detection & Prevention Model (IDPMIA), SQL, SQLIA

1. Introduction

Online finance marketing, shopping centres, trades share, online purchasing, air ticketing, banking & hoteling services etc. are increasing day by day and for their security and data integrity purpose, new preventive measures are adopted. As the e-commerce and internet banking industries are becoming popular, the threats of cybercrimes are also increasing and new SQL Injection Attacks exploitations of systems are also revealed. Due to some discrepancies in SQL query execution structure, an attacker can attack easily in different ways (Dalai & Jena, 2017). An attacker can add malicious code with the execution of a legitimate query, can drop, alter, update data and retrieve important & secret data even can exploit a website or can crash a system. (Yousaf & Sheraz, 2018) Moreover, a system can be crashed or a website can be exploited due to some serious type of SQL Injection Attacks which are as follows:

2. Problems / Issues

Some basic vulnerable problems are categorized as below.

Sr #	Problem	Query
1	Tautology Query	Select Name, Department, MobNo, Email From Employee Where EmpID = 1 or 1=1
2	Union Query	https://profiles.abc.edu.pk/staff/pharmacy.asp?ProID=1 Union Salary, Residence From StaffPharm;
3	Stacked or Piggybacked Query	https://profiles.xyz.edu.pk/staff/pharmacy.asp?ProID=1 Drop Table StaffPharm
4	Commented Queries	Select * From StaffPharm Where EmpID = '1'; -- and Password = '123';

2.1 Tautology Query

Tautology queries are easily used for SQL Injection Attacks and they retrieve data in the shape of chain / loop from a database server and site server too. These queries are executed until the last tuple record would not be retrieved. A query example of a tautology is given below.
Select Name, Department, MobNo, Email from Employee Where EmpID = 1 or 1=1
Due to its nature 1=1 it becomes always true and executes till the last EmpID record.

2.2 Union Query

Union query is used for data stealing. It combines the two queries result due to Union operator. Union query retrieves some confidential / personal data like as Salary and Address which we hide from the user.
https://profiles.abc.edu.pk/staff/pharmacy.asp?EmpID=1 Union Salary, Residence from StaffPharm;

2.3 Stacked / Piggybacked Query

A stacked query or Piggybacked query is executed an additional code with the execution of a legitimate query. After the completion of a legal query, an attacker include some extra code or an extra query at the end of that legal query. Due to the validation of legal query, both queries are executed. This query is most commonly used for drop or alter able. For example:-

Manuscript received August 5, 2019
Manuscript revised August 20, 2019

https://profiles.abc.edu.pk/staff/pharmacy.asp?EmpID=1
Drop Table StaffPharm
This situation is alarming, that is why, a user cannot delete table. Tables are main parts of a database. Therefore, this malicious query is too much vulnerable, for keeping / managing database record save.

2.4 Commented Queries

The Commented query is also used in SQLIA. With the comment sign "—" an attacker take benefit of comment signs, for example, the following query without password be executed, if the EmpID =1.
Select * From StaffPharm Where EmpID = '1'; -- and Password = '123';
Without the knowledge of password an attacker can access the EmpID = 1 account, which is illegal and challenging for data integrity and security.
It is dire need to save data from hacker's attacks and prevent these attacks before execution. Different types of tools and techniques are introduced for data safety from attackers, like as, Concolic Execution Paths, (Yousaf & Sheraz, 2018), DetAnom, (Hussain & Sallam, 2015) DIDAFIT (Lee & Wong, 2013), Except of 'Select' and 'delete' command use Where, Having, Like or Order By clause (Dalai & Jena, 2017) and Signature base Profile creation (Yousaf & Sheraz, 2018), focused on client side through client's web browser (Shahriar 2013), firewall layer approaches focused on examining the HTTP request, depends on using neural network (Moosa, 2010), hybrid system using Bayesian classifier & pattern matching (Makiou, 2015), focused on web application layer & general classification (Kumar, 2015), (Alwan, 2017), defensive coding approaches, vulnerability testing discovers & fixes possible injection hotspots, automatic generation of test inputs & cases (Shin, 2009), Ruse, 2010), Server side vulnerabilities by designing a black box testing method (Bisht, 2010) web page scanning to discover the vulnerabilities with a defined database error table (Roy, 2011), programming tracing techniques (Wang, 2012), Negative taint model used evasion methods by maintaining a vulnerability lookup table for all possible attacks (Alazab, 2016), Parsing tree validation (Buehrer, 2005), Matching certain Word in the query AC Pattern (Prabakar, 2013), SQL query as a graph of tokens an SVM classifier for recognized possible malicious inputs (Kar, 2016), Lightweight Directory Access Protocol LDAP (Zhang, 2011), dynamically analyzed the developer intended query result size for any input & compared it against the result of the actual query (Jang, 2014), database server internal query trees (Kim, 2014), technique Apriori for checking the queries (Jawanja, 2018), SQLI Detection tools (Elia, 2010), code conversion algorithm (Balasundaram, 2012). All these are useful in different ways. No one is perfect or as ideal solution for all kinds of attacks. Due to this client's satisfaction is still not lingering. But in this research, we are introducing a new technique in which all stored legal paths i.e. queries will store statically and according to each query a negation prediction base query will also be saved. When a user input a query the query will be matched with the simple stored query and data would be retrieved, in other way according to the base on that input query the negation prediction query will also be executed and data would be retrieved. Both query executed results, data retrieval would be compared. If both are equal then query is a valid query and would be executed, otherwise its difference would be measured and on behalf of difference it would be decided that how much query is vulnerable. An Intrusion Detection & Prevention Model for SQL Injection Attacks (IDPMIA) is introduced, which can measure the input queries before their execution. This system can also be implemented dynamically on behalf of all stored legal paths and their negation prediction bases.
There are many techniques and tools for detection of SQL Injection Attacks but no one is having complete reliable security for organizations. For example, SQLRand can detect tautology queries attack, but partially detect the commented queries and mismatch queries attack but it cannot able to catch the attack of stored procedure and alternative methods. Security Policy Descriptor Language (SPDL) can partially detect tautology queries, type mismatch, stacked query, Union Query, Stored Procedure & Inference attacks, but not able to catch alternative methods attacks. SIIMDS is not able to detect the attacks of tautology queries, Union Queries, and Alternative methods. SQLIPA is not able to detect the attack of alternative methods. AMNESIA is a better technology which can detect tautology query, comment line queries attacks etc., but not able to catch stored procedure attacks. The adopted model (IDPMIA) is a very good technology for prevention & detection from hacker's attack of Union Queries, Stacked Queries, Commented queries and Tautology queries etc.

3. Literature Review

(Morsi & Ahmed, 2019) have described in their research paper regarding SQLIA. Attacks can be done in the form of user input, cookie fields containing attach strings, server variables & second order injections. Hacker's main focus is to target the five important critical layers in the end to end web application architecture.
So many past technologies were introduced, but the data integrity & safety problems are remaining unsolved. So the author introduced a new approach, combination of two ancient approaches, consist of static & dynamic algorithm. In this approach, researchers define a combination of AC

pattern for static phase Parser Tree validation technique as dynamic algorithm.

Gurina and Eliseev (2019) stated that infection signs not only include from incoming traffic online, but also, they can exist in a local computer or local network and can travel when data packets are transferring, which are harmful. Mostly signature base anomaly detection patterns are popular, but not a single universal technique or over all complete solution is defined.

Author defined a technique based on the measurement of size of the data, which is exchanging through connection, networking or internet, second is, what is the data exchange time. The data stream response time is also be considered. It is also populate that it does not depend on the order of request, but it depends the information contained in the request. But what would be happened, if a user is having 4G or extra efficient speed stream and second user is too low bandwidth. The record of 100 rows retrieved, but not as required tuples.

Yousaf & Malik (2018) described a technique with the combination of 3 modules, adopted for anomaly detection in SQL and for different security purposes and protection of data from unauthorized users. Due to SQL injection detection technique, queries would be transparent from interloper and code would not be malevolent. Disgusting / intercepted queries cannot be executed and situation of DoS creates in database. But this technique is not useful in real time environmental databases.

An anomaly detection and Reconstruction Schemas (AD&RS) is also emphasized, in which researcher perform work with some Modules i.e. Anomaly detection Module, Query reconstruction module, Query delegation Module etc. Author makes signature base profiles on behalf of all possible outcome ways, in respect of all valid possible queries, their path in term of Profile generation / signature base profile be stored in databases. When a user execute query, that query would be converted on specific signature base type / shape, and then query is analyzed / compared with stored queries. If it is a valid query or authorized path, which would be allowed, then can execute. Otherwise after comparing its difference from authorized query revealed and after the deletion of its additional part, its remaining exact path, which would be a valid query, can be executed and result would be displayed on user's system. This technique emphasized in the shape of tree, query table, code and a final comparison query table. In which for the reconstruction of query the comparison table is a better way for denying a service query or delegate / valid a query for retrieval of data. If a query does not compare then its difference from a near path, considered as additional path, which makes this query anomalous. For comparison linear algorithm is used. Therefore, its additional path / difference from original / valid path would be deleted and the remaining query which remains valid would be executed.

Dalai & Jena (2017) proposed a technique / process, consisting on 6 different points to store the query as strings. Moreover, to avoid the injections attack due to Where clause, they suggested to use the Non-Where clauses, with the help of Order By, Having and Like. They worked for the protection of online injection attacks on Web Applications. They extracted the SQL input queries of users in a different pattern. Researchers emphasize online application problems and also take an insight view of an attack procedure, which attacker normally adopt pattern step by step. i.e take the OS information, SQL Version information, Metadata referred tables from Master DB and Column information etc. After achieving a useful knowledge / information of Database, Attacker easily can inject the additional code with the use of a legal query.

George & March (2016) explained that attacks can rise due to SQL injection; there are some security risks present. An attacker can access database and can perform unauthorized functions. Due to this it is necessary to check the validity of query execution. A server between the main server and user must be present which will check the query that it is legitimate or not. With the help of java template, researchers described it can reconstruct with an intermediary server. An intermediary server may increase the efficiency of server and reduce the denial of service ratio. Researchers described the Where clause query with 'Or' and 'And' functions, and also mentioned the tautologies, Union, statements, logically incorrect and piggybacked queries.

Sadotra (2015) mentioned how SQL Injection Attacks affects web applications and how the attacker can take advantages of database weaknesses. Moreover, researchers discussed the web applications, web tools, weak point, SQL Injection Attacks and web queries. Introduced a more reliable technique based on calculations of HASHING & ENCRYPTION. With the help of this technique user name and password can login more secured. Each username and password would be stored with some calculated hash values of username, calculated of hash password value and calculated Hash Ex-Or value. But it is useful for the authentication of user and based on calculations.

Abu Othman (2014) discussed Structure Query Language Injection Attacks (SQLIA) regarding the Web Applications. Mostly highlighted problems are tautologies, Union query, inference, commented, logical incorrect queries, stored procedure, piggy – backed queries, blind injection, timing attacks and alternate encoding etc. These are very basic SQL injection attacks, used by hackers.

4. Methodology Intrusion Detection & Prevention Model for SQL Injection Attacks (IDPMIA)

Our proposed IDPMIA will diagnose a query and check its parameters, before it's execution, that it is a valid query or not? It will check the input query to find errors (consider as typing error), some additional code or some malicious data. If some extra data or minor typing error is identified in the query, it will be sent for reconstruction to convert it into a valid query. If it is having some malicious code, then query will catch before it's execution, in query comparison part, abruptly query will truncate and an alarm or message would be sent to DBA. However, the most significant part is the negation prediction comparison. Just like, in signature base profile system, we store all the legal paths of an application and whenever a query is received from user side, that input is compared with all legal stored paths, in proposed model, we will store valid queries and negation prediction of that queries and whenever a new query request received from user, that query will be verified with the existing stored queries. If query is valid, the query will be considered a legitimate query otherwise it will be sent for reconstruction as preventive measure. However, if query is not reconstructable then a warning message will be generated for DBA.

This technique consist of SQL Server and dotnet framework and will use for detection of anomalies / illegal queries of user's before their execution, in a database. First of all, it diagnoses a query and its parameters that it is a valid / executable query, then can run it. If it is having an additional code or reveals that it is a typing error then query can send for reconstruction and after reconstruction, if it becomes a valid query then execute it.

But its most powerful part is the negation prediction comparison. It means, as in signature base profile system, we will store all the legal paths of an application and when a query / input received from user side, that input be compared with the stored paths. If it compared as a valid signature then it can be executed otherwise would be terminated / truncated the query. For the severity of data, and the vulnerable / exploitation of SQL Injection Attacks, it is necessary that whereas all legal paths of an application be stored, the negation predication of queries are also be stored. When user input a query, first of all it will compare with a stored legal path, if they are equal then two process / query would be proceeded. a) retrieval of data in regard of query. b) retrieval of data by the negation prediction of that query. Both results would be compared. If the comparison result will same then query will count a legal query. Otherwise its difference of comparison will be evaluated. If that query would be reconstruct able, then rebuild the query and executed. But if it would be found vulnerable, then abruptly system will terminate the query and also will send an alarming message to the DBA / Developer. This technique is helpful, because with the negation prediction comparison an additional data retrieval apparently reveals and the malfunctioning queries base on tautologies, commented and union query diagnoses.

This technique will be implemented for a dynamic environment. All statically generated path and authorized query and their negation prediction would be saved on an additional server. When a user will submit query for execution on a web server. First it will analyze in an IDPMIA, weather it is a legitimate query or not. Which would be based on all static terms generated legal path, negation predictions, queries, restrictions manual and then finally it will decide for the execution of a query.

An IDPMIA is shown in Figure 1.1

Example: 1

The negation prediction comparison can also emphasize with the help of following tables.

Student_Table

StID	Name	Class
1	Irfan	ICS
2	Imran	BCS
3	Sultan	MCS

Select Name From Student_Table

Which selected attribute result is highlighted with green colour.

StID	Name	Class
1	Irfan	ICS
2	Imran	BCS
3	Sultan	MCS

And its negation prediction can be defined as:

Except StID, Class from Student_Table.

Which excepted / exempted attributes are mention in query or highlighted with red colour.

StID	Name	Class
1	Irfan	ICS
2	Imran	BCS
3	Sultan	MCS

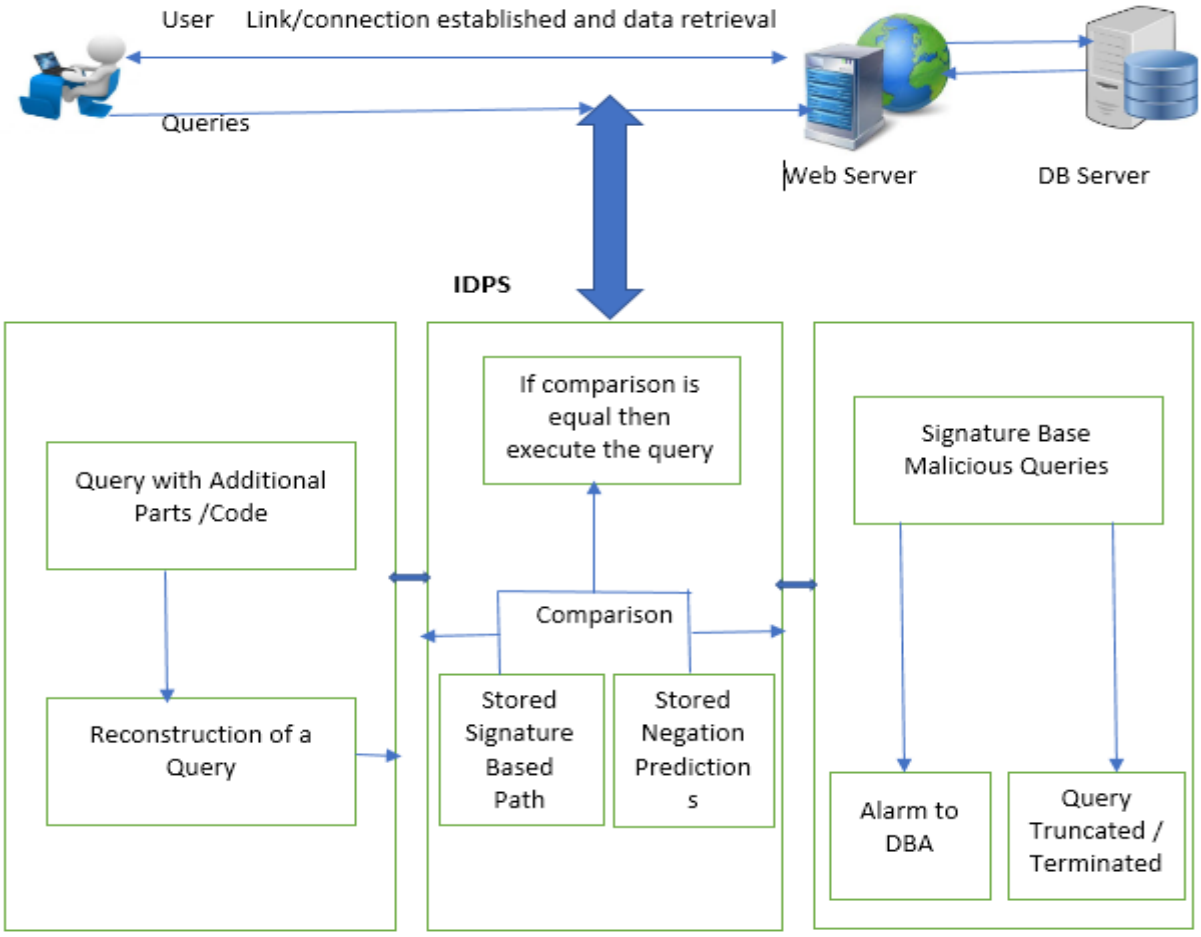
Now it can be compared easily that selected attribute in 1st query is green and exception attributes are highlighted with red colour. In 1st query selected attribute "name" and in 2nd query remaining attribute is "name" column, which are same. Therefore, this query can be executable. So as the additional command, of Stacked / Piggybacked queries for updation or drop a table found, it can easily catch when compared with stored negation predictions.

SQL Server Query for Selected Columns

```
USE [{DB_name}]
GO
/****** Object: StoredProcedure [dbo].[sp_GetSelectedColumnsOfTable] Script Date: 8/27/2019 1:11:23 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
--
============================================
====
-- Author: <MALIK RIZWAN ALI>
-- Create date: <27-08-19,>
-- Description: <SELECTED COLUMNS OF TABLES,>
--
============================================
====
Create PROCEDURE [dbo].[sp_GetSelectedColumnsOfTable]
	-- Add the parameters for the stored procedure here
@Cols AS NVARCHAR(MAX)='Name,City',
@TblName AS NVARCHAR(MAX)='tblEmployee'
AS
BEGIN
 DECLARE @SqlCmd VARCHAR(MAX)=
'SELECT '+@Cols+'
   FROM '+@TblName;
	EXEC(@SqlCmd);
END
```

SQL Server Query for Un Selected Columns

```
USE [{DB_name}]
GO
/****** Object: StoredProcedure [dbo].[sp_GetUnSelectedColumnsOfTable] Script Date: 8/27/2019 1:12:05 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
--
============================================
====
-- Author: <MALIK RIZWAN ALI>
-- Create date: <27-08-19,>
-- Description: <UN-SELECTED COLUMNS OF TABLE,,>
--
============================================
====
Create PROCEDURE [dbo].[sp_GetUnSelectedColumnsOfTable]
	-- Add the parameters for the stored procedure here
@Cols AS NVARCHAR(MAX)='Name,City',
@TblName AS NVARCHAR(MAX)='tblEmployee'
AS
BEGIN

DECLARE @CSV VARCHAR(MAX)

 SELECT
	COLUMN_NAME as ccolumns
	into #txl
FROM
	INFORMATION_SCHEMA.COLUMNS
WHERE
	TABLE_NAME = @TblName

SELECT * into #splitColumns FROM fnSplitString(@Cols , ',')

Select * into #finalColumns from #txl where ccolumns not in (select splitdata from #splitColumns)

SELECT @CSV = COALESCE(@CSV + ', ', '') + ccolumns from #finalColumns

--SELECT @CSV AS Result


 DECLARE @SqlCmd VARCHAR(MAX)=
'SELECT '+@CSV+'
   FROM '+@TblName;


EXEC(@SqlCmd);

Drop Table #txl
Drop Table #finalColumns
Drop table #splitColumns
END
```



IDPS Features:

a. Dynamically and Statically can save queries.
b. All saved queries have their negation base queries.
c. Simple query and its negation base query both will run at once.
d. Both result of data retrieval will compare.
e. If it is equal then query is a valid query
f. If minor change or typing error then query will reconstruct.
g. Redesigned queries / Delete additional Input from query.
h. After reconstruction the query will execute again
i. If found major difference then query is malfunction & vulnerable.
j. Abruptly indicate a message to DBA.
k. Query will truncate abruptly.

Conclusion & Future Work:

This technique is better for the prevention of execution Tautology Queries, Commented queries, Union Queries & Packed Queries. This technique gave an extremely accurate view in the case of Tautology queries. This technique easily handle the union queries result and no additional data can be retrieved. This technique is an efficient to detect the harmful and vulnerable injection attacks. Stored negation prediction queries comparison with the stored legitimate queries, when reveals difference, the input query must be having additional input and can harmful.

The next goal of this work is that how an efficient algorithm be developed that could measure the both side result within a part of nanoseconds. Both side data be measured / compared via its length in shape of tuples or in shape of attributes, or both or can be measured in shape of size or it's weight. Simple saved query results and negation prediction saved query results can be measured by default.

This work does not encompass the combination of signature base profiles systems. More efforts needed to make it efficient in banking systems or online websites for replication and fragmentation.

References

[1] A. Liu, Y. Yuan, D. Wijesekera & A.Stavrou (2009), SQLProb, A Proxy based architecture towards preventing SQL Injection Attacks. (ACM).

[2] Asish Kumar Dalai & Sanjay Kumar Jena, 2017, Neutralizing SQL Intection Attack Using Server-Side Code Modification in Web Applications, Hindawi, Security and Communication Networks.

[3] Asmaa Sallam, Elisa Beritino, Fellow; IEEE, Syed Rafiul Hussain, David Landers, R. Michael Lefler, & Donald Steiner, 2015, DBSAFE- An Anomaly Detection System to Protect Databases From Exfiltration Attempts.

[4] Diallo Abdoulaye Kindly & Al-Sakib Khan Pathan, (2011), A Survey on SQL Injection.

[5] D.Scott & R. Sharp, (2003), Specifying and enforcing application level Web Security Policies. (IEEE).

[6] F. Maggi, M. Mateucci & S. Zanero (2010), Detecting Intrusion through system call sequence and argument analysis (IEEE).

[7] Gurina Anastasia & Eliseev Valdimir, 2019, Anomaly-Based Method for Detecting Multiple Classes of Network Attacks, (MDPI)

[8] Jensen, CS, Prasad, M.R & Moller A (2013), Automated Testing with targeted event sequence generation. (ACM).

[9] Mona Faruk, (2019), A Two-Phase Pattern Matchin-Parse Tree Validation Approach for Efficient SQL Injection Attacks Detection. (ISSN).

[10] Muhammad Sohaib Yousaf, M. Sheraz Arshad Malik, Muhammad Asif, Farhat Naz, Ijaz Ali Shoukat, May, 2018, SQL Anomaly Detection and Reconstruction of Queries Eliminate the Denial of Service, International Journal of Computer Science and Network Security, Vol. 18.

[11] Noor Ashitah Abu Othman, Fakariah Hani Mohd Ali & Mashyum Binti Mohd Noh, (2014), Secured Web Application using Combination of Query Tokenization and Adaptive Method in Preventing SQL Injection Attacks. (IEEE)

[12] Ntagwbira Lambert & Kang Song Lin, (2010) User of Query Tokenization to Detect an SQL Injection Attacks.

[13] Parveen Sadotra, (2015), Hashing Technique – SQL Injection Attack Detection & Prevention. (ISSN).

[14] Parveen Sadotra (CEH), Dr. Anup Girdhar, (2013), Penetration Testing – Cyber Security Assessment (IJTM).

[15]

[16] Shu, X., Yao, D., & Ramakrishnan, N., 2015. Unearthing stealthy program attacks buried in extremely long execution paths. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security.

[17] S.Lee, Low, & P.Wong, (2002) Learning fingerprints for a dtabase intrusion detection System.

[18] Som, S. Sinha, S. & Kataria, R (2016), Study on SQL Injection Attacks – Mode detection and Prevention (Impact Factor)

[19] S.R. Hussain, A.M. Sallam & E. Beritino, (2015), Detanom, Detecting Anomalous Database Transactions by Insiders. (ACM).

[20] S.R. Hussain, A.M Sallam & Bertino, (2016), DetAnom, Detecting Anomalous Database Transactions by Insiders. (ACM).

[21] T.K George & Poulose Jacob, PhD, Cochin University of Science & Technology, March 2016, A Proposed Architecture for Query Anomaly Detection and Prevention against SQL Injection Attacks, International Journal of Computer Application Vol. 137.

[22] V. Prokhorenko, K.R. Choo, & H. Ashman, (2016), Intent Based Extensible Real Time PHP Supervision Framework. (IEEE).

[23] W. Li, B. Panda & Q. Yaseen, (2012), Mitigating insider threat on database integrity. (ISS).

[24] Xu, K., Yao, D. D., Ryder, B. G., & Tian, K., 2015. Probabilistic program modelling for high-precision anomaly classification. In Computer Security Foundations Symposium (CSF), IEEE

Sheraz has obtained his PhD in Information Technology and currently working as Assistant Professor in Department of Information Technology at GCUF, Pakistan. His areas of interests are HCI, Big Data, IoT and Information Visualization.

Rizwan is MS Computer Science Student at Virtual University Pakistan. He is also working in Government of Pakistan. His areas of interests are Databases, Online Information Retrieval Systems and Pervasive Computing technologies.