ISSN No. 0976-5697

Volume 8, No. 3, March – April 2017

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

SQL Injection Impact on Web Server and Their Risk Mitigation Policy Implementation Techniques: An Ultimate solution to Prevent Computer Network from Illegal Intrusion

Parveen Sadotra (CEH)
Research Scholar,Department of computer, Application,
Career Point University,Kota, Rajasthan, India

Dr. Chandrakant Sharma
Assistant Professor,Department of computer Application,
Career Point University,Kota, Rajasthan, India

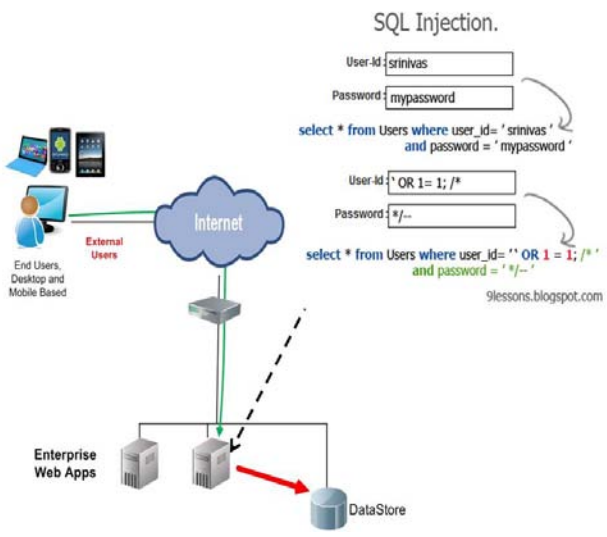
***Abstract*:** SQL Injection attacks pose a very serious security threat to Web applications and web servers. They allow attackers to obtain unrestricted access to the databases underlying the applications and to the potentially sensitive and important information these databases contain. Although researchers and security professionals have proposed various methods to address the SQL injection problem but current approaches either fail to address the full scope of the problem or have limitations that prevent their use and adoption. Many researchers and security professionals are familiar with only a subset of the wide range of techniques available to attackers who are trying to take advantage of SQL injection vulnerabilities. As a result, many solutions proposed in the literature address only some of the issues related to SQL injection. To address this problem, we are presenting an extensive review of the different types of SQL injection attacks known to date. Also for each type of attack, we provide descriptions and examples of how attacks of that type could be performed. We also presented and analyze existing detection and prevention techniques against SQL injection attacks.

***Keywords*:**SQL injection attack, SQL queries, web application, DBMS, taxonomy, web application**.**

I. INTRODUCTION

There are many web attacks hacker follows to attack on your web servers. SQL Injection is mostly used attack mechanisms used by hackers to steal data from web server or manipulate them. It is surely one of the most common known application layer attack techniques used in present time. It is the type of attack which takes advantage of improper coding of your web applications that consequently allows hacker to inject SQL commands into a login form and allows them to gain access to the data held within the database. [1]SQL Injection targets the web servers and web applications that use a back-end database.

To understand better way here is a brief illustration.



II. NEED / IMPORTANCE OF THE STUDY

SQL injection is a very important to study now days, here are some interesting facts for a few different reasons in SQL injection: -

- ❖ It's getting increasingly tougher to write vulnerable codes due to frameworks that automatically parameterize inputs while designing web applications yet we still write bad code.
- ❖ You are not necessarily in the clear just because you have used stored procedures or a shiny ORM (though you are aware that SQL Injection can still get through these, alright...?) You still build vulnerable web applications round these mitigations. [2]
- ❖ It is very easy to detect remotely by automated tools which can be orchestrated to crawl the web searching for vulnerable websites yet you are still putting them out there.

III. IMPACT OF SQL INJECTION ON WEB SERVER (RISK ASSOCIATED WITH SQL INJECTION)

When an IT department noticed an enormous spike in queries to its website and corresponding error messages, they correctly suspected it was the subject of an SQL injection attack. [3] In such case of attack, an attacker sends intentionally malformed requests to a company's website hoping that the server will malfunction and either return non-public data in response to the request or grant the attacker deep, administrative access to the server.

The risk factors associated with SQL injection are as below: -

The platform affected with SQLIA can be as:

- Language: SQL
- Platform: Any (requires interaction with a SQL database)

SQL Injection has become a common issue with database-driven web sites and web applications. The flaw is easily detected, and easily exploited, and as such, any site or software package with even a minimal user base is likely to be subject to an attempted attack of this kind. Essentially the attack is accomplished by placing a meta character into data input to then place SQL commands in the control plane, which did not exist there before. This flaw depends on the fact that SQL makes no real distinction between the control and data planes. The main impact of SQL Injection attacks is as below:

- **Confidentiality**: Since SQL databases generally hold sensitive data, loss of confidentiality is a frequent problem with SQL Injection vulnerabilities.
- **Authentication**: If poor SQL commands are used to check user names and passwords, it may be possible to connect to a system as another user with no previous knowledge of the password. [4]
- **Authorization**: If authorization information is held in a SQL database, it is very much possible to change this information through the successful exploitation of SQL Injection vulnerability.
- **Integrity**: Just as it may be possible to read sensitive information, it is also possible to make changes or even delete this information with a SQL Injection attack.

Classification of SQL Injection

Following table presents various types of SQL injection attack and their vulnerabilities: -

Si No	Vulnerability	A brief explanation
1	Bypassing Web Application Authentication	One of the most common usages adopted by the cyber criminals to bypass authentication pages, used in web applications. In this category of attack, an attacker exploits an input field that is used in a query's 'where' condition part.
2	Getting Knowledge of Database Fingerprinting	This attack is considered as pre-attack preparation by the attackers. This category of attack is performed by entering some inputs by which it generates an illegal or the logically incorrect queries. The error messages reveal the names of the tables and the columns that cause error. The attacker also comes to know about the application database used in the backend server.
3	Injection with UNION query	In such an attack, an attacker extracts data from a table which is different from the one that was intended in the web application by the developer. An attacker exploits a vulnerable parameter to change the data set returned for a given query.
4	Damaging with additional injected query	The attack in this category of attack is generally very harmful. An attacker enters input such that an additional injected query is generated along with the original query.
5	Remote execution of stored procedures	This category of attack is conducted by executing the procedures, stored previously by the web application developer.

IV. RESULTS AND DISCUSSION

In this section, we will provide a detailed analysis of our results and findings. The complete audited results are presented below. To understand better we have organized the results according to following three headings:

- The scope of SQL injection threats
- Why the SQL injection threat remained pervasive so far
- SQL injection threat and behavioral analysis to combat this threat

The scope of SQL injection threatsSQL attacks is pervasive.

As shown in following table 63 percent of respondents said that their organization experienced one or more SQL injection attacks during last 12 month which evaded its firewalls and other parametrical defense system. On an average, it took approximately 3 months to detect the attack also at the same time 40 percent of respondents said that it took about six months or even more to detect the attack. So approximately it took an average of 70 days to detect such attacks.

Si No	1 of more SQL injection Attack in last 1 year	
	Reported Response	Percentage
1	Yes	63
2	No	31
3	Unable to determine	6

Table: SQL Injection reported in last 12 months

Seriousness of SQL Injection Threat

52 % i.e. around half of respondents accepted that SQL injection threat faced by their organization is absolutely

important if we rate threat from number 1 to 10 then seriousness of the treat is as responded is given in following table

Si	**Scale of seriousness of threat**	**Percentage**
1	1 to 2	6
2	3 to 4	12
3	5 to 6	13
4	7 to 8	22
5	9 to 10	48

Table: Seriousness of SQL Injection Threat

Present state of SQL Injection threat
We can see the present state of SQL injection state as below: -

- SQL injection attacks are increasing : 50 %:
- SQL injection attacks are decreasing : 30 %
- SQL injection attacks are staying at the same level:15 %
- Unable to determine :5 %

Does threat of SQL injection remain pervasive
There are many institutes and organizations which are not familiar with the tricks and techniques used by cyber attackers.in the following table, we can see about pervasive of SQL injection threat and familiarity with Web Application Firewall bypass: -

Si	**Scale of seriousness of threat**	**Percentage**
1	Very Familiar	15
2	Familiar	30
3	Not Familiar	45
4	No idea about it	10

Table: Familiarity with the bypass techniques used by cyber criminals

Understanding the root causes of an SQL injection attack more difficult.
Respondents' perceptions about the SQL injection threat are shown in table below.

Si No	**Descriptions**	**Strongly Agree In %age**	**Agree In %age**
1	Understanding the root causes strengthens my organization's readiness to mitigate future attacks	34	37
2	Determining the root causes is more difficult because of personally owned mobile devices in the workplace	28	30
3	Determining the root causes is more difficult because of the sophistication of cyber attackers	22	21
4	Technologies are in place to quickly detect a SQL injection attack	16	20
5	IT security personnel possess the skills, knowledge and expertise to quickly detect a SQL injection attack	12	19

Table: Understanding the root causes of an SQL injection attack more difficult

If we see above table, we can see that Seventy one percent of respondents believe understanding the root causes of SQL injection attacks strengthens my organization's readiness to mitigate future attacks.

However, 56 percent of respondents say determining the root causes of SQL injection is becoming more difficult because of the trend for employees to use their personally owned mobile devices in the workplace. Another challenge, according to 42 percent of respondents, is increasing stealth and/or sophistication of cyber attackers

Expertise & the correct technologies are critical while preventing SQLIA
While respondents see the SQL threat as serious, only 33 percent say their organization's IT security personnel possess the skills, knowledge and expertise to quickly detect a SQL injection attack and 36 percent agree that they have the technologies and tools to quickly detect a SQL injection attack.

Measures to prevent SQL injection attacks are also lacking.
Despite concerns about the threat, 54 percent do not take such precautions as testing and validating third party software to ensure it is not vulnerable to SQL injection attack, as we can see in above table.

Is Third party software being tested and validated to ensure it's not vulnerable

As you can see the table below, 44 percent of respondents say their organization make use of use of security professional as penetration testers to identify vulnerabilities in their information systems but only 37 percent of these organizations include testing for SQL injection vulnerabilities.

Si No	Software tested or not	Percentage
1	Yes, most third-party software	13
2	Yes, some third-party software	31
3	No	51
4	Not sure	5

Table: Is Third party software being tested and validated to ensure it's not vulnerable

Whether Professional testers employed to identify vulnerabilities

As shown in above table below forty five percent of respondents say their organization used professional penetration testers to identify vulnerabilities in their information systems but only 36 percent of these organizations include testing for SQL injection vulnerabilities. Results are as in following table: -

Si No	Professional testers employed or not	Percentage
1	Yes	46
2	No	49
3	Not sure	05

Table: Whether Professional testers employed to identify vulnerabilities

Frequency of Monitoring for active databases

See following table to response about monitoring for active databases is needed

Si No	Frequency of monitoring	Percentage
1	Continuously	21
2	Daily	14
3	Weekly	06
4	Monthly	04
5	Quarterly	03
6	Half Yearly	02
7	Yearly	07
8	Irregular Interval	24
9	Don't Scan	19

Table: Frequency of Monitoring for active databases

Above table shows that one-third of respondents say they either scan continuously or daily for active databases. However, 24 percent scan irregularly and 19 percent do not scan at all.

Behavioral analysis solution to combat the SQL injection threat

Here we define behavioral analysis technology for securing database transactions as a technology that automatically creates a model of proper SQL behavior. Each SQL statement attempts to access and analyze the database is being tested against the behavioral model. Any activity that deviates from the established behavioral model is flagged as a likely security event. Behavioral analysis provides immediate protection against zero-day threats.

Si No	opinion about behavioral analysis approach for detecting SQL	Percentage
1	Very favorable	39
2	favorable	50
3	Not favorable	11

Table: Behavioral analysis solution to combat the SQL injection threat

We can see here that Organizations are adopting behavioral analysis. According to above Table, 89 percent of respondents view behavioral analysis either very favorably or favorably.

How will behavioral analysis-based systems be used

See the Following table for behavioral analysis based systems

Si No	behavioral analysis-based systems	Percentage
1	On client systems	65
2	On the perimeter network	55
3	For database transaction security	50
4	For web application security	35
5	Other	03

Table: How will behavioral analysis-based systems be used

Fifty nine percent of respondents say their organizations have or will within the next two years replace its signature-based IT security systems with behavioral analysis-based systems, as shown in above table. Most will be used on client systems (65 percent of respondents) followed by on the perimeter network 55 percent of respondents.

With the help of above findings, we can see that SQL injection attack is on increase. Though organizations are becoming more aware concerned about this attack and they are trying various measures to avoid these kinds of attack still we can see these attacks are not stopping. Lack of trained manpower is also one of main hindrance in this regard.

V. NEW PROPOSED MITIGATION POLICY

Several papers in literatures have proposed ways to prevent SQLIA in the application or database tier. Comprehensive survey of SQLIA [5] detection and prevention techniques. SQLIA countermeasures techniques are divided into three main approaches: static, dynamic and hybrid approaches. Static approaches are desirable during development and testing phase of applications. Developers should follow some techniques for SQL injection prevention. Static approaches counteract the possibility of SQLIA at compile time. [6] Whereas, dynamic approaches are useful for analysis of dynamic SQL query, generated by web application. This approach performs countering the possibility of SQLIA at runtime. Both approaches may need analysis or modification of application's source code. In hybrid approaches exploit a combination of static and dynamic approaches. These approaches attempt to utilize advantages of both approaches for preventing and detecting SQLIA. [7] In the rest of this paper, some of the new and popular existing static, dynamic and hybrid techniques are presented.

A. Static Approach

Static Approach has following types of techniques: -

1) An algorithm of prepared statement replacement for removing SQLIVs

Thomas *et al.* proposed a prepared statement replacement (PSR) algorithm and corresponding automation for removing SQLIA vulnerabilities from vulnerable SQL statements by replacing them with secure prepared statements. This method analysis source code containing SQLIVs and generates a specific recommended code structure containing prepared statements. An SQLIV exists does not keep statement structure and input separate.

PSR-algorithm collects information from application's source code which possible including SQLIVs. Then generates secure prepared statement code that maintains functional integrity. Another algorithm which called Prepared Statement Replacement Generator (PSR-Generator) is created for automates the generation of the prepared statement-based code in Java, which results from the PSR-Algorithm. PSR-Algorithm is useful for developers which have source code contains SQLIVs and need to be removed. Authors claim that their proposed method is remove SQLIVs with minimal manual intervention. Note that PSR-Algorithm is used to remove only SQLIV and does not have to be integrated into the runtime environment.

2) MUSIC: mutation-based SQL injection vulnerability checking

Shahriar and Zulkernine, proposed a Mutation-based SQL Injection Vulnerabilities Checking (testing) tool (MUSIC) that automatically generates mutants for the applications written in Java Server Pages (JSP) and performs mutation analysis. Mutation is the act of intentionally modifying a program's code, then re-running a suite of valid unit tests against the mutated program. Mutation testing is a method of fault-based software testing, which involves modifying programs' source code or byte code in small ways. Mutation testing is done by selecting a set of mutation operators and then applying them to the source program one at a time for each applicable piece of the source code. [8] The result of applying one mutation operator to the program is called a mutant. These mutants are killed by a test case if it is causes different end output or different intermediate state between the original program and a mutant. Otherwise the mutant is remaining alive. Additional test cases should be generated for killing live mutants. Authors proposed nine mutation operators to inject SQLIV in source code of application which four of them inject faults into generated SQL queries and remaining five of operators inject faults into the API method calls. However, MUSIC is very simple and effective way for testing SQL queries having simple form, but it cannot address the SQLIV of stored procedures.

3) Sania: syntactic and semantic analysis for automated testing against SQL injection

Sania, is a technique for detecting SQLIV in web applications in development and debugging phase which using the following procedures.

1) Sania intercepts the SQLqueries between a web application and a database. Then, collects normal SQL queries between client and web applications and between the web application and database, and analysis the vulnerabilities.
2) It automatically generates elaborate attacks according to the syntax and semantics of the potentially vulnerable spots in the SQL queries.
3) After attacking with the generated code, it collects the SQL queries generated from the attack. [9]
4) Sania compares the parse trees of the intended SQL query and those resulting after an attack to assess the safety of these spots.
5) Finally, it determines whether the attack succeeded or not. By analyzing the syntax in the parse tree of SQL queries, it is possible to generate precise pinpoint attack requests. [10]

B. Dynamic Approach

We discuss three different popular dynamic approaches for countering to SQLIA.

1) AIIDA-SQL

This method suggests a hybrid approach based on Adaptive Intelligent Intrusion Detection (AIIDA-SQL) for detection of SQLIA. AIIDA-SQL combines the advantages of Case-Based Reasoning (CBR) systems, such as learning and adaptation, with the predictive capabilities of a combination of Artificial Neural Network (ANN) and Support Vector Machine (SVM). Through these mechanisms, [11] their take advantages of both strategies in order to trustworthy classifying the SQL queries. In final manner, in order to classify SQL queries as distrustful, utilized a virtualization mechanism, which combines clustering techniques and unsupervised neural models to reduce dimensionality of the data.

2) A query tokenization based method

This is a technique based query tokenization is proposed for detect and prevent SQL injection attacks. This method checks user inputs whether their cause changes query's intended result. At the next step, this method tokenizing original query and malicious injected query, separately. After tokenizing, two arrays are created by all tokens. Finally, the lengths of obtained arrays are compared. If their length be different, an injection attack is detected.

3) A learning based approach

Bertino *et al.* have proposed a framework based on anomaly detection techniques to detect malicious behavior of database application programs. The approach is as follows. At first step, a fingerprint of an application program based on SQL queries is created. Then, take advantages of association rule mining techniques to extract useful rules from these fingerprints. These rules depict normal behavior of the database application. Finally, dynamic queries check against these rules to detect injection attacks that not conform to these rules.

C. Hybrid Approach

1) A method based on removing SQL query attribute values

Lee *et al.* proposed a simple and efficient method for detecting SQLIA. Their method utilizes static and dynamic phases for finding vulnerabilities in web application. This method removes the attribute values of SQL queries at runtime (dynamic method) and compares them with the SQL queries analyzed in advance (static method). [12] It detects attacks by comparing the structure and the grammar of the queries. If a dynamically generated query has a different structure or uses a different grammar from that of a static query, it is detected. Authors shown that their proposed method has time complexity and can implement on any type of DBMS.

2) Obfuscation-based Analysis of SQL Injection Attacks

Halder and Cortesi proposes the obfuscation and de-obfuscation based technique to detect the presence of possible SQLIA in a query before submitting it to a DBMS. In the static phase, the queries in the application are replaced by queries in obfuscated form. Now the Obfuscated code is a source code that has been made difficult for human. [13] In obfuscation approach the possible attack injection are verified at atomic formula level and only those atomic formulas which are tagged as vulnerable, also this approach avoids the root cause of SQL injection attacks in dynamic query generation. Authors show that their proposed algorithm could detect SQLIA with negligible runtime overhead and do not dependent on specific application.

Here are the ways you can help prevent or mitigate SQL injection attacks in your organization and save your websites and web applications.

1. Trust no-one:

This first rule to assume all user-submitted data is evil and validate and sanitize everything.

2. Don't use dynamic SQL when it can be avoided:

Don't use dynamic SQL when it can be avoided used prepared statements, parameterized queries or stored procedures instead whenever it is possible.

3. Automate SQL injection testing

In the early days of SQL injection attacks, manual testing was the only way to determine if systems, databases or web applications were vulnerable to the SQL injection threat. Manual testing sifting through error messages and database structure information is a long and tedious process, [14]and even then, is no guarantee that you will find every vulnerability. So, you must follow automatic SQL testing system.

You can say thanks to new several automated tools available to carry out simulated SQL injection attacks on your own databases to see how susceptible your systems and applications are to threats. Here you can also learn more about how ethical hacking tools can help detect vulnerabilities before they are exploited and how to perform automated tests for all vulnerabilities, including SQL injections, to stop attacks before they start.

4. Updating and patching the systems:

Vulnerabilities in applications and databases that hackers can exploit using SQL injection are regularly discovered so it's very important to apply patches and updates as soon as practical and possible.

5. Implementation of Web Application Firewall:

Never forget firewall. Always Consider a web application firewall (WAF) either software or hardware based which will help to filter out malicious data. Good ones will have a comprehensive set of default rules, [15] and make it easy to add new ones whenever necessary. A WAF can be particularly useful to provide some security protection

against a particular new vulnerability before a patch is available.

6. Reduce your attack surface:

Get rid of any database functionality that you don't need to prevent a hacker taking advantage of it. For example, the xp_cmdshell extended stored procedure in MS SQL spawns a Windows command shell and passes in a string for execution, which could be very useful indeed for a hacker. The Windows process spawned by **xp_cmdshell** has the same security privileges as the SQL Server service account.

7. Implement Host Intrusion Prevention System (HIPS) Rules

A foundational goal in computer and system security is maintaining the health or integrity of individual hosts a HIPS is a valuable component used to defend computer host integrity. In enterprise deployments, Host Intrusion Prevention Systems are centrally managed, and SAs push policies and rules down to the individual hosts. Alerts of malicious or abnormal activity on the hosts are pushed back up to the management system where they can be correlated and acted upon.

8. Control Administrative Privileges

Administrative privileges on a computer system allow access to resources that are unavailable to most users and permit the execution of actions that would otherwise be restricted. When such privileges are administered improperly, granted widely, and not closely audited, attackers are able to exploit them and move effortlessly through a network. Gaining administrative privileges is commonly achieved through a technique known as privilege escalation. Privilege escalation is defined as the act of exploiting a bug, design flaw, or configuration oversight in an operating system or software [16] application to gain access to resources that are unavailable to normal users. Poorly managed administrative privileges make executing this technique much easier.

It is very much important that on an SQL server, appropriate controls be put in place to prevent a hacker from “breaking out” of the database itself. If you are using SQL Server, ensure that you have disabled xp_cmdshell and that the web user’s permissions are as limited as possible. Ensure that the database user account is NOT given system privileges as this is one of the main methods for attackers to break out of the database.

9. Use appropriate privileges:

Don't connect to your database using an account with admin-level privileges unless there is some compelling and concrete reason to do so. Using a limited access account is always safer and can limit to manifold what a hacker is can do.

10. Secrets is secret:

Secret is Secret so keep it secret only. Assume that your application is not secure and act accordingly by encrypting or hashing passwords and other confidential data including connection strings then never people let it know about it.

11. Use a Web Vulnerability Scanner to Find and Fix Vulnerabilities

Network owners and operators should remain constantly vigilant in knowing at all times the state of their network. It is important that System Administrators (SAs) implement a plan to scan their public-facing web-server for common vulnerabilities, [17] using one of any number of very good commercial scanners. As vulnerabilities are found, they should be fixed or patched. This is especially crucial for networks that have older web applications; as sites get older, more vulnerabilities are discovered and exposed.

Recommendations to scan for following vulnerabilities:
- SQL Injection
- Local File Inclusion
- Cross-Site Scripting
- General Coding and Input Errors

12. Don't divulge more information than you need to:

Don't divulge more information than you need to hackers can learn a great deal about database architecture from error messages so ensure that they display minimal information as possible. Use the "Remote Only" custom Errors mode to display verbose error messages on the local machine while ensuring that an external hacker gets nothing more than the fact that his actions resulted in an unhandled error.

13. Harden Web Applications

It’s the most effective way to prevent attacks. The unsecured purchasing site played a crucial role in the adversary’s attack, essentially serving as an unlocked door to the back-end database. Hardening these web applications is a vital step in preventing intrusions, and should be part of every enterprise service’s overall operation and mitigation strategy.

For MS-SQL Server

i) Ensure the SA account has a very strong password
ii) Remove the SQL guest user account
iii) Remove the BUILTIN\Administrators server login
iv) Do not grant permissions to the public role

Many hackers will use SQL injection to obtain dumps of the database user’s credentials in order to use them to SSH or telnet into the system. Ensure that database user accounts have a different password from any privileged account on the system to prevent this from happening. Ensure that all passwords meet recommended complexity requirements and ensure that you have changed all of the default passwords on your database accounts. Many times, people don’t change even default passwords. [18] If your database is not encrypting, salting, or hashing user’s passwords, this must be fixed immediately. Regardless of which database you are

using, the user account associated with the web application should have the minimum privileges possible.

14. Never forget the basics:

Never forget the basics like changing the passwords of application accounts into the database regularly. This is common sense, but in practice these passwords often stay unchanged for months or sometimes even for years.

15. New defenses for automated SQL injection attacks

In the recent time attackers, have used SQL injection attack methods to quickly find and exploit website vulnerabilities and effectively spread malware. So, in this scenario in order to prevent SQL injections, enterprise security teams must go above and beyond the old SQL defense of testing and patching Web application code.

Companies must not only build defenses and practice secure coding best practices, but also develop an in-depth understanding of how SQL injection attacks work and how the threat has evolved the earlier SQL injection attacks didn't have the vulnerability detection capabilities of contemporary attacks -- as well as learn how to find, [19] isolate and address websites infected with malware on a website. In this tip a security expert Michael Cobb explains how the SQL injection threat has evolved, what types of defenses, such as toolkits and vendor products, are available today to help thwart the threat and best practices for protection from SQL injection attacks of the future.

16. Buy better software:

Make code writers responsible for checking the code and for fixing security flaws in custom applications before the software is delivered. [20] SANS also suggests you incorporate terms from this sample contracting to your agreement with any software vendor.

VI. CONCLUSIONS

SQL injection attack is not new this attack poses a serious security threat over the Internet or over web application from the day we started DBMS. In SQL injection attacks, hackers can take advantage of poorly coded Web application software to introduce malicious code into the organization's systems and network. The vulnerability exists when a Web application do not properly filter or validate the entered data by a user on a Web page. Large Web applications have hundreds of places where users can input data, each of which can provide a SQL injection attack opportunity. [21] Attackers/hackers can steal/edit/delete confidential and critical data of the organization with these attacks resulting loss of market value of the organization. This paper presented an effective survey of SQL Injection attack, detection and prevention techniques. We also analyzed some existing techniques to detect attack and mitigate risk associated with these attacks.

We presented a system for preventing SQL injection attacks against web servers. The main intuition is that by using a randomized SQL query language, specific to a particular CGI application, it is possible to detect and abort queries that include injected code. By using a proxy for the de-randomization process, we achieve portability and security gains: the same proxy can be used with various DBMS back-end, and it can ensure that no information that would expose the randomization process can leak from the database itself. Naturally, care must be taken by the CGI implementer to avoid exposing randomized queries (as is occasionally done in the case of errors). We showed that this approach does not sacrifice performance: the latency overhead imposed on each query was at most 6.5 milliseconds.

With the help of this paper we tried to provide taxonomy of methods for prevent and detect SQL injection attacks. We first define vulnerabilities in web application and how these vulnerabilities may cause SQL injection attacks. Then, we present a classification of SQLIA based on vulnerability. Afterwards, divide the SQL injection and prevention methods to three different categories: static, dynamic and hybrid approaches. These approaches different in the time which are counteracting to possibility of SQLIA. The paper discusses different SQL detection and prevention techniques for a given attack which recently been proposed. Furthermore, we evaluated these techniques, with respect to deployment requirements. For all the negative impact of SQL injection vulnerability, the countermeasures are surprisingly simple to enact. The first rule, which applies to all Web development, is to validate user-supplied data. SQL injection payloads require a limited set of characters to fully exploit vulnerability. Web sites should match the data received from a user against the type (for example, integer, string, date) and content (for example, e-mail address, first name, telephone number) expected. We believe that each detection or prevention technique cannot provide complete protection against SQLIA, [13] but a combination of the presented mechanisms will cover a wide range of SQL injection attacks which will culminate in a more secure and reliable database which is protected against SQL Injection Attacks.

VII. REFERENCES

[1] https://www.acunetix.com/websitesecurity/sql-injection/
[2] http://cis1.towson.edu/~cssecinj/modules/other-modules/database/sql-injection-introduction/
[3] http://www.ijcce.org/papers/244-E091.pdf
[4] https://www.cs.columbia.edu/~angelos/Papers/sqlrand.pdf
[5] Parveen Sadotra (CEH), “Hashing Technique - SQL Injection Attack Detection & Prevention” International Journal of Innovative Research in Computer and Communication Engineering, Vol. 3, Issue 5, May 2015 pg. 4356 – 4365.
[6] https://www.uscert.gov/sites/default/files/publications/sql200901.pdf
[7] http://www.cisco.com/web/about/security/intelligence/sql_injection.html
[8] https://www.defcon.org/images/defcon-10/dc-10-presentations/dc10-spett-sqlinjection/dc10-spett-sqlinjection.pdf
[9] https://blog.udemy.com/sql-injection-tutorial/

[10] http://www.dbnetworks.com/form/Ponemon_SQL_Injection_Threat_Survey.htm
[11] http://www.ponemon.org/local/upload/file/DB%20Networks%20Research%20Report%20FINAL5.pdf
[12] http://research.ijcaonline.org/volume114/number17/pxc3902007.pdf
[13] Parveen et al., "A New Proposed Method for Detection and Prevention of SQLIA (SQL Injection Attack)"International Journal of Advance Research in Computer Science and Management Studies, Volume 3, Issue 5, May 2015 pg. 68-75.
[14] http://www.w3schools.com/sql/sql_injection.asp
[15] https://blog.udemy.com/sql-injection-tutorial/
[16] http://www.sqlinjection.net/risks/
[17] https://www.owasp.org/index.php/SQL_Injection
[18] http://www.owasp.org/index.php/Main_Page
[19] http://www.enterprisenetworkingplanet.com/netsecur/article.php/3866756/10-Ways-to-Prevent-or-Mitigate-SQL-Injection-Attacks.htm
[20] http://searchsecurity.techtarget.com/tutorial/SQL-injection-protection-A-guide-on-how-to-prevent-and-stop-attacks
[21] https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet

