



# Using Extensive Database Firewall to Secure Relational Database Management System against SQL Injection Attack

Surya Pratap Singh\*, Avinash Singh, Upendra Nath Tripathi, Manish Mishra

Research Scholar, Dept. of C.S., DDU Gorakhpur University, Gorakhpur India

Research Scholar, Dept. of C.S., DDU Gorakhpur University, Gorakhpur India

Assistant Professor, Dept. of C.S., DDU Gorakhpur University, Gorakhpur India

Assistant Professor, Dept. of Electronics, DDU Gorakhpur University, Gorakhpur India

**ABSTRACT:** The security of Database is very important in this fast growing environment because database is now used by every running application and web programs. The Relational Database Management System is most widely used Database. In the recent past various types of attacks and security breaks are done on RDBMS. The most serious security problem in RDBMS arises due to SQL Injection Attack. In this type of attack the attacker might gain unrestricted access to the database and its confidential data and information, this can be done by submitting the malicious SQL queries to change the intended working of the application and trick the server to gain improper authentication and access the database by using SQL queries. Different researches are done to protect the database from SQL injection Attack but none of the approach is fully able to protect the database.

In this paper we first discuss different ways by which SQL Injection attack can break the security of Distributed database. We also analyze various available approaches to protect the database against SQL Injection attack. In this paper we also propose the use of Extensive Database Firewall by which we can guard the Relational Database against SQL Injection attack.

**KEYWORDS:** Database Security; RDBMS; threats to database; SQL Injection Attack; Extensive Database Firewall.

## I. INTRODUCTION

In the modern environment the rapid development of internet, online systems and services that use the web application to present the services and use the web methodologies are becoming a very productive strategy for software development companies.[1] These urgency leads to design of those applications which can potentially used by norms amount of users from simple web client systems. As the use of web applications are increasing there is an extensive need to protect the database and provide an effective security mechanism on web applications.[3]

There are various types of attacks done to lure the security of database, in which most dangerous type of attack is SQL injection attack.[4]The SQL injection refers to a class of code injection attack. Code injection is a type of utilization caused by processing invalid user inputs. In this type of attack the user is included in an SQL query in such a manner that some portion of the input provided by the user is treated as SQL code. Such types of attacks may be performed by adding or piggybacking malicious characters into data values in the web form or parameter's argument in the URL.

The SQL injection attack or SQLIA is a type of code injection attack which consists of injection of malicious SQL statements and queries from the client to the application that are later passed to the instance of the Relational database for execution and affects the execution of predefined SQL commands.[6] These attacks can become very costly security threat to any Web application as well as Database systems that receives input from users and process it into SQL queries. Most of the Web applications available these days perform their functions in similar fashion so they are vulnerable to SQL injection attack.[8]

There are various ways by which we can prevent or confront with these attacks. For instance the database designer or administrator can use different techniques in the application development phase which contains uses parameterized queries, different account level permissions, least privileges, customized error messages etc. these techniques can



# International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 2, February 2016

prevent the relational database from SQL injection but their application remains vulnerable in practice. The techniques can cause problems because of human errors and cannot be completely and rigorously applied as an automated model of work against SQLIA.[7][9]

Various researchers have suggested a wide range of techniques and methodologies to help the developers of the database application to overcome from the above problems, but as the problem is dynamic in nature none of the approach is able to fully protect the Relational database.

## II. SQLIA AND VULNERABILITY

Vulnerability is a flaw or shortcoming in the application which can be caused by improper design or can be an implementation bug. An attacker can use these vulnerabilities to cause harm to the Relational database system. Some of the major vulnerabilities which are targeted by most of the attackers are –

- a) SQL injection Attack
- b) Session Management Drop holes
- c) Cross Site Scripting
- d) Cross Site Request Forgery
- e) Broken Authentication etc.

SQL injection Attack vulnerability occurs when an attacker changes the intended way of SQL query by inserting some malicious code. By using these vulnerabilities an attacker could send his commands directly to web application's underlying database and destroy functionality or confidentiality. SQLIA can be classified into five basic classes based on vulnerabilities in web applications. The classification of vulnerabilities is elaborated in the followed sections

A. **Getting Knowledge of Database Fingerprinting:** This attack is considered as pre-attack preparation by an attacker. 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.

B. **Bypassing Web Application Authentication:** This is the most common usage taken by most of the attackers to bypass authentication procedure, used in web based database and applications. Here in this category an attacker exploits an input field that is used in a query's condition part in 'where' statement.

C. **Remote execution of stored procedures:** This category of attack is conducted by executing the procedures, stored previously by the web application developer.

D. **Damaging with additional injected query:** 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.

E. **Injection with UNION query:** In this kind of attack, an attacker extracts data from some database object like 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.

## III. SQL INJECTION MECHANISM

The malicious SQL statements can be injected into a vulnerable application using various input mechanisms. The most common mechanism of SQL injection is explained bellow –

A. **Cookies based Injection:** Cookies are data set from websites and are stored in user's web browser to store the state information when the user is browsing or performing any web related actions. When a client returns to a Web application, cookies can be used to restore the state information sent by the client's. Since the client has control over the storage of the cookies, here a malicious client can tamper the contents of cookies. If a Web application uses the cookie's contents to build SQL queries, an attacker could easily submit an attack by embedding it in the cookie.

B. **User input based Injection:** In such methods, attackers inject SQL commands by providing suitably which crafted as user input. A Web application can read user input in several ways based on the environment in which the application is deployed. In most SQLIA's that target Web database and applications, here user's input typically came from submission of forms that are sent to the Web application via HTTP GET or POST requests. Web applications are generally able to access the user input contained in these requests as they would access any other variable in the environment.

C. **Server variables based Injection:** Server variables are a collection of variables that contain HTTP, network headers, and environmental variables. Web applications use these server variables in a variety of ways, for example



# International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 2, February 2016

identifying browsing trends and logging usage statistic. If these variables are logged to a database without sanitization, this situation can create vulnerability of SQL injection[5]. Because attackers can forge the values that are placed in HTTP request/ response and network headers, the attackers can exploit this vulnerability by embedding or appending an SQLIA directly into the headers. When the query to log the server variable is issued to the database, the attack in the forged header is then triggered.

D. **Second-order injections:** In second-order injections, attackers provide malicious inputs into a system or database to indirectly initiate an SQLIA when that input is used at a later time. Second-order injections are not trying to cause the attack to occur when the malicious input initially reaches the database. Now the attackers can rely on knowledge of where the input will be subsequently used and craft their attack so that it occurs using that usage.

**For example** – suppose a user registers on a website providing a username such as “admin’ --”. Here the application escapes the single quote in the input before storing it in the database properly, preventing its potentially malicious effect. Now if the user changes his/her password, the operation now involves the followings

- Validating that the user knows the current password, and
- Changing the password if the validation is successful.

To perform this operation, the Web application might construct an SQL command as follows:

```
queryString="UPDATE users SET password='" +newPassword +" WHERE userName='" + userName + "' AND password='" +oldPassword + """
```

Here the newPassword and oldPassword are the new and old passwords and userName is the name of the user currently logged-in say “admin’--”. Therefore, the query string that is sent to the database is UPDATE users SET password='newpwd' WHERE userName= 'admin’--' AND password='oldpwd'

Because “--” is the SQL comment operator, everything after it is ignored by the database. Therefore, the result of this query is that the database changes the password of the administrator (“admin”) to an attacker-specified value.

## IV. REVIEW OF LITERATURE

Marco Cova [1]: presents Swaddler, a novel approach to the anomaly-based detection of attacks against web applications. They analyze the internal state of a web application. It works based on both single and multiple variables and shows an impressive way against complex attacks to web applications. First the approach describes the normal values for the application’s state variables in critical points of the application’s components. Then, during the detection phase, it monitors the application’s execution to identify abnormal states.

Almgren, M [2]: presents an intrusion-detection tool aimed at protecting web servers, and justify why such a tool is needed. They describe several interesting features, such as the ability to run in real time and to keep track of suspicious hosts. The design is flexible and the signatures used to detect malicious behavior are not limited to simple pattern matching of dangerous cgi scripts. The tool includes mechanisms to reduce the number of false alarms.

W. G. Halfond [3]: address the problem of SQL injection attack and present an extensive review of the different types of SQL injection attacks known to date. For each type of attack, they provide descriptions and examples of how attacks of that type could be performed. They also present and analyze existing detection and prevention techniques against SQL injection attacks. For each technique, they discuss its strengths and weaknesses in addressing the entire range of SQL injection attacks.

G. T. Buehrer [4]: describe a technique to prevent SQL injection vulnerabilities. Their technique is based on comparing, at run time, the parse tree of the SQL statement before inclusion of user input with that resulting after inclusion of input. Their solution is efficient, adding about 3 ms overhead to database query costs. In addition, it is easily adopted by application programmers, having the same syntactic structure as current popular record set retrieval methods. For empirical analysis, they provide a case study of our solution in J2EE. They implement our solution in a simple static Java class, and show its effectiveness and scalability.

## V. PROBLEM IN EXISTING DATABASE SYSTEM

The SQL Injection attack is one of the most dangerous security threats to the database systems. Various database researchers proposed different methodologies but the database systems available till date still have some serious problems discussed below-

In the modern database system the authentication of the user is done by username and password validation mechanism, here once the user get pass to the authentication process can do the intended work without any problems. Now day the

# International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 2, February 2016

database and application level security can be achieved by the firewall which can enhance the security of the database systems to a good extent. But the firewall available till date is only able to protect the data from normal malicious activities. It can't protect the database from SQLIA. The firewall can only filter the data for malicious code but it can't detect malicious SQL queries. This situation is explained in the fig 1.

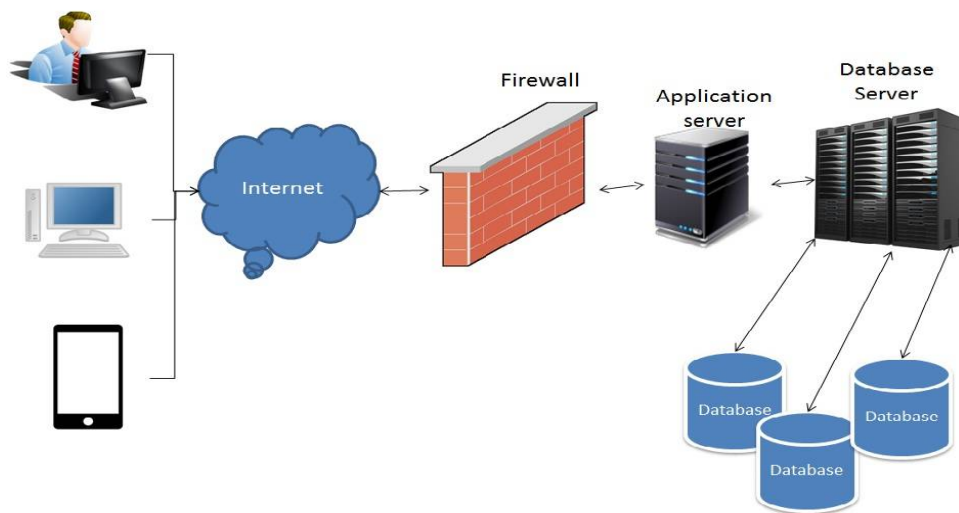


Fig 1. Use of firewall to secure data

From the above fig 1, it is clear that the clients can access the data from the web server, and it is controlled by Firewall but the normal firewall can't protect against malicious SQL queries. From the above fig when the user provide any input it is filtered by firewall for malwares and then passed to the application server for processing but the firewall cannot filter/detect the SQL queries.

The second problem with the existing approach is that the attacker might attempts to modify the present SQL statement by adding elements to the where clause. For example- suppose someone wants to find the record of employee by a search application. The application takes the emp\_id of employee as input and provides the matched value as an output. In this case the web application like this may run the following query  
Select \* from employee where emp\_id=' <input from the user >'

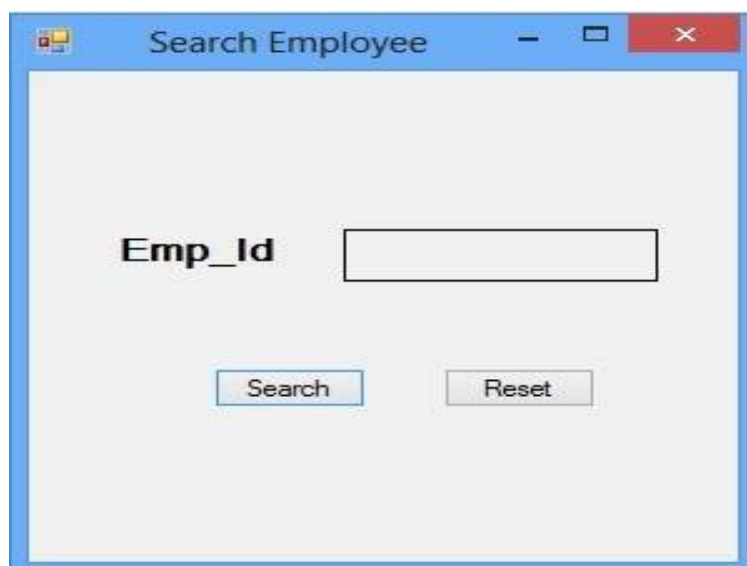


Fig 2. User Input to Search Record

# International Journal of Innovative Research in Computer and Communication Engineering

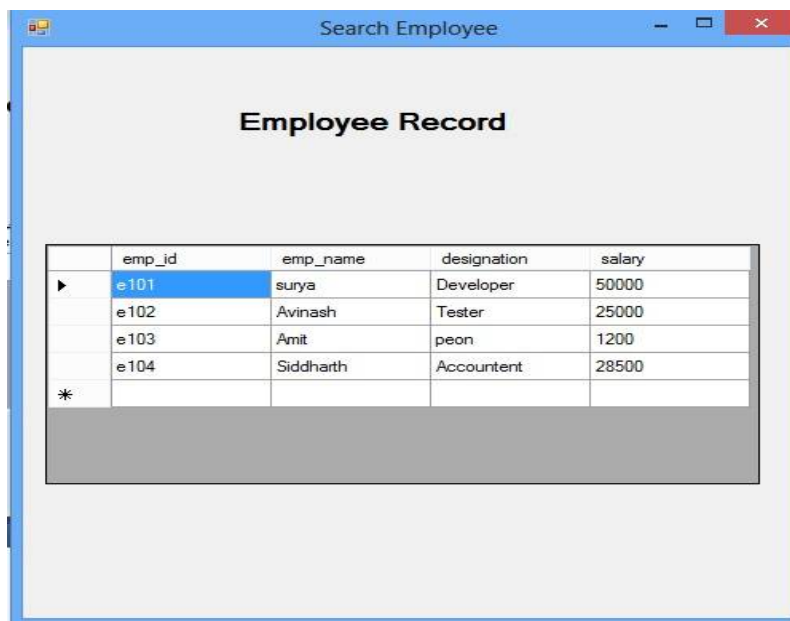
(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 2, February 2016

The Fig 2, Demonstrate the snapshots of Search Employee form of a web application. Here if the user wants to find the record of a particular employee he/she can pass the Emp\_id of such employee in the employee id input box and then press search button. This application now searches the record in the database of employees and if the matching is found then the record is displayed on the users screen. This application is vulnerable to SQL Injection attack. If the user provides the following query –

**Select \* from employee emp\_id ='1' = 1 or ' '**

In such case the record of all employees listed in the database is shown to the user because the above queries matches to all the emp\_id stored due to SQL anomaly. The output of the above query is displayed in the Fig 3.



emp_id	emp_name	designation	salary
e101	surya	Developer	50000
e102	Avinash	Tester	25000
e103	Amit	peon	1200
e104	Siddharth	Accountant	28500
*			

**Fig 3. Search Result on the basis of malicious Input**

From the fig 3 We can see that when the user provide the query like which is given above all the data containing in the database table is displayed, this leads to a serious logical problem which have to be tackled .

## VI. PROPOSED METHODOLOGY

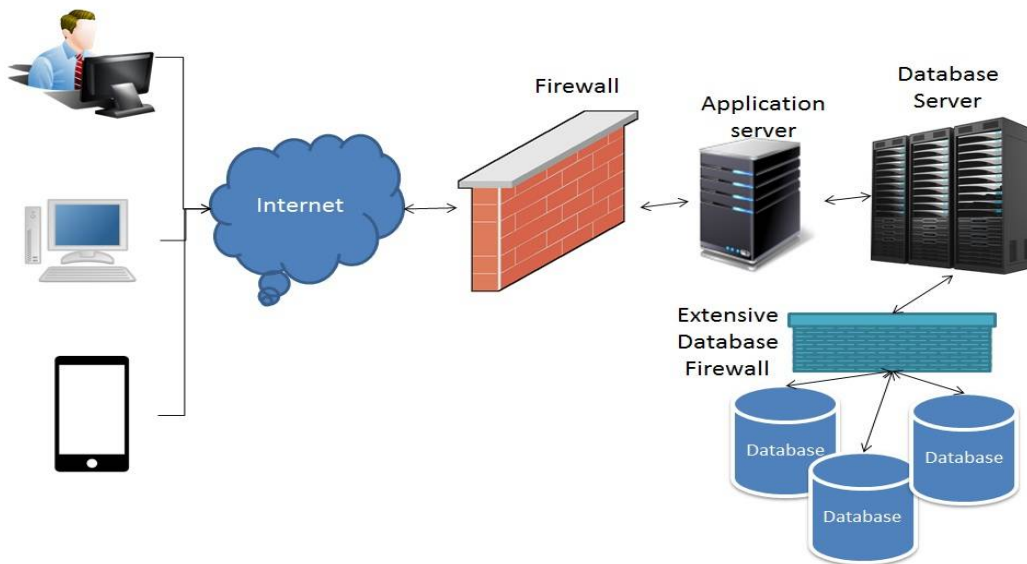
The problems of SQL Injection are very serious and needs a lot attention than others. To overcome from the above mentioned problems we propose the following methodologies-

**Use of Extensive Database Firewall** – we propose the use of extensive database firewall which is designed in such a manner to protect the database by filtering SQL queries. The higher level of security can be achieved by the use of extensive database firewall which is placed between the database server and database instances. The proposed extensive Database firewall filters all the database related queries and then passes to the database for fetching and storing record. In this manner by the use of extensive database firewall we can protect the database from SQL Injection attack.

# International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 2, February 2016



**Fig 4. Use of Extensive Database Firewall**

From above fig 4 it is clear that the extensive database firewall acts as a bridge between Database server and actual database and provides an effective mechanism of protection against SQLIA. Here when the user wants to login to the database server the following steps are performed

*Step 1* – User pass the login credentials.

*Step 2* – The login details passed by user reaches to the application server.

*Step 3* – On the application server side the data is filtered by the use of Firewall.

*Step 4* – The application server pass the filtered data to database server.

*Step 5* – Now the data is filtered and tracked by our proposed Extensive Database Firewall, and then after it reaches to the actual database.

The schematic work of Extensive Database Firewall is shown in the Fig 5. Here the working procedure of Firewall and Database Firewall is described. Form the fig 5. When the user provides the login credential it is bundled in the HTTP request here the security assessment on application server is done by the traditional firewall for security threats other than SQLIA. If any such threat found the firewall denies the access, it is shown in the fig 5. Invalid Access, now when the credential are verified then security assessment on database server is done by our proposed Extensive Database Firewall which is designed to filter the queries and data against malicious SQL statements like SQLIA. Here if data and queries don't contain any threat the normal process and access to the database is continued but on the other hand if any suspected SQL statement is found the Extensive Database Firewall triggers an alarm and denies the access of database. Hence in this manner Two Level security of the database is achieved.

# International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 2, February 2016

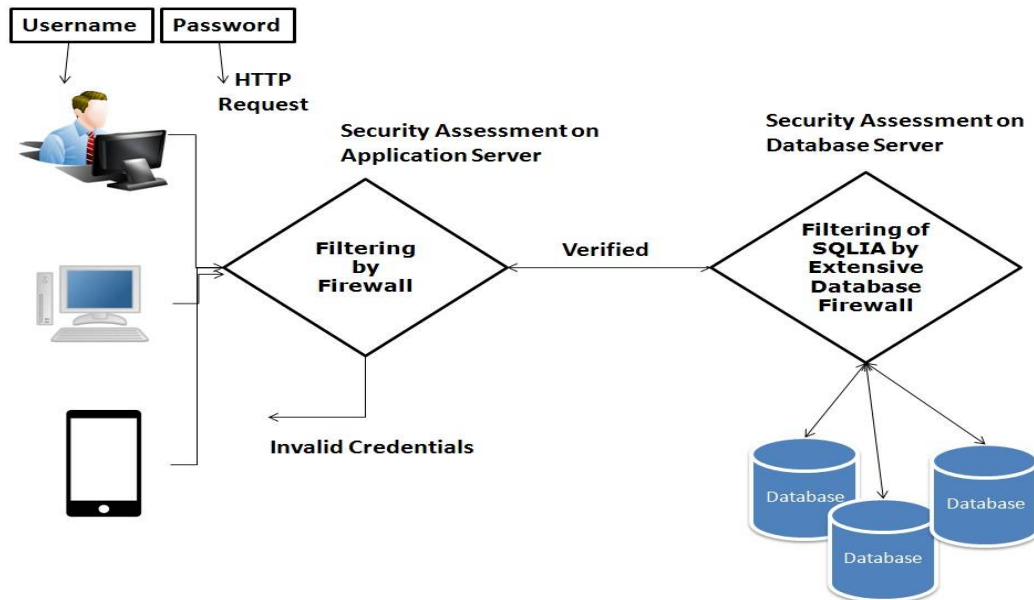


Fig 5. Schematic work of Extensive Database Firewall

## VII. CONCLUSION AND FUTURE WORK

The SQL Injection attack is the most vulnerable security issue to the Relational database because the database hackers and intruders are trying everything to break the security of database so there is an urgency to protect the database from these kinds of vulnerabilities. Various database researchers proposed different approaches to protect the Relational database from these problems but the approaches need some improvement.

In this paper we discuss the SQL Injection attack and its injection process. We explained the way by which the database is affected by SQL injection attacks, we also categorized the SQLIA on the basis of vulnerabilities and shows the basic problems that are incurred in the current database systems. To overcome from these problems, in this paper we proposed the use of Extensive database firewall to protect the database from SQLIA. The proposed methodology is able to secure the database against such SQL injection attacks which wishes to do illegal login to the database server, but there may be other cases of SQL Injection on database too. In future we tries to cope those situations which corresponds to illegal uses of database.

## REFERENCES

1. Marco Cova, Davide Balzarotti. Swaddler, 'An Approach for the Anomaly-based Detection of State Violations in Web Applications', Proceedings of Recent Advances in Intrusion Detection, Vol.4637 pp. 63-86, 2007.
2. Almgren, M. Debar, H., Dacier, M, 'A Lightweight Tool for Detecting Web Server Attacks'. In Proceedings of the Network and Distributed System Security Symposium (NDSS), San Diego, CA pp.253-261, February 2000
3. W. G. Halfond, J. Viegas, and A. Orso, 'A Classification of SQL injection Attacks and Countermeasures', in Proceedings of the International Symposium on Secure Software Engineering, pp. 32-41. 2006.
4. G. T. Buehrer, B. W. Weide, and P. A. G. Sivilotti, 'Using Parse Tree Validation to Prevent SQL Injection Attacks'. In International Workshop on Software Engineering and Middleware (SEM), pp. 25-38 2005.
5. A. Christensen, A. Moeller, and M. Schwartzbach. 'Precise analysis of string expressions'. In Proceedings of the 10th International Static Analysis Symposium, 2003.
6. W. G. Halfond and A. Orso, 'Combining static analysis and runtime monitoring to counter sql-injection attacks', In Online Proceeding of the Third International ICSE Workshop on Dynamic Analysis (WODA 2005), St. Louis, MO, USA, 2005.
7. Y. Huang, S. Huang, T. Lin, and C. Tsai. 'A Testing Framework for Web Application Security Assessment'. Journal of Computer Networks, Vol. 48 Issue 5, Pp. 739-761, 2005.
8. Surya pratap singh, Avinash singh, UpendraNathTripathi, 'Proactive mechanism of security against SQL injection attack' in International journal of scientific research and management (IJSRM), Vol. 3 Issue 5 may pp. 2786-2789, 2015.
9. K. Amirtahmasebi, S. R. Jalalinia, S. Khadem, 'A survey of SQLInjection defence mechanisms', Proceedings of ICITST 2009, vol.5 pp.1-8, Nov. 2009
10. C. Gould, Z. Su, and P. Devanbu. 'Static Checking of Dynamically Generated Queries in Database Applications'. In Proceedings of the 26th International Conference on Software Engineering (ICSE 04), pp. 645-654, 2004.



# International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 2, February 2016

## BIOGRAPHY



“Surya Pratap Singh is MCA and UGC-NET qualified. He is pursuing Ph.D In the department of Computer Science Deen Dayal Upadhyay Gorakhpur University, Gorakhpur (U.P. India) under the supervision of Dr. U.N. Tripathi. His area of research interest is Database Security, Algorithm design and Networking. Surya Pratap Singh has published 20 papers in different national and international conferences/ Journals.



Avinash Singh is M.Sc. Computer Science, M.Tech and M. Phil and pursuing Ph.D. in the department of Computer Science DDU Gorakhpur University, Gorakhpur (U.P. India) under the supervision of Dr. U.N. Tripathi. The area of research interest is Database Security, Networking. Mr.Avinash Singh has published 23 papers in different national and international conferences/ Journals.



Dr.Upendra Nath Tripathi is Assistant professor in Department of computer science DeenDayalUpadhyay Gorakhpur University, Gorakhpur (U.P. India). He has 14 years of teaching and research experience. He has published more than 40 papers in various National and International Journals/conferences. His area of research interest is database systems and networking.



Dr. Manish Mishra is Assistant professor in Department of Electronics DDU Gorakhpur University, Gorakhpur (U.P. India). He has 14 years of teaching and research experience. He has published more than 45 papers in various National and International Journals/conferences. His area of research interest is Computer Technology, fast processor design.