International Journal of Scientific Research in Computer Science and Engineering
Vol.5, Issue.5, pp.41-46, October (2017)

Research Paper

E-ISSN: 2320-7639

# Effectiveness of SQLI Countermeasures

**Daljit Kaur[1*], Parminder Kaur[2]**

[1*]Dept. of Computer Science, Lyallpur Khalsa College, Jalandhar, India
[2] Dept. of Computer Science, Guru Nanak Dev University, Amritsar, India

[*]*Corresponding Author: jeetudaljit@hotmail.com, Tel.: 9779326951*

**Available online at: www.isroset.org**

Received 10th Sep 2017, Revised 24th Sep 2017, Accepted 17th Oct 2017, Online 30th Oct 2017

***Abstract*—** In the recent times web applications has become increasingly popular with the growth of web. At the same time, there is an increase in number of attacks in web applications. Attacks like injection vulnerabilities such as SQL Injection, Cross site Scripting, Cross site Request Forgery(CSRF) are common. This paper specially focuses on countermeasures of SQL Injection vulnerability. Here, we have implemented various attacks on a Giftshop web application and also classified SQL Injection countermeasures with respect to Software Development Life Cycle and tested them for their effectiveness with the help of vulnerability scanners. Finally, the result of vulnerability scanners are shown and analyzed before and after the implementation of known SQL Injection countermeasures.

**Keywords—** *SQL Injection; Attacks; Vulnerability scanners; Threats; Web application; Security*

## I. INTRODUCTION

Security has been the critically important part of most of the web applications. Online Services and rapid development of Internet use the web paradigm. But with the growth of World Wide Web and increase in these online services, attacks on web have also grown. Therefore, effective security mechanisms on web applications and addressing them seem to be very important in these days. SQL injection (SQLI) continues to be one of the most predominant web application threats as it has compromised large number of websites including those of some high profile companies. It allows attackers to obtain unauthorized access to the back-end database to change the intended application-generated SQL queries. This type of attack exploits vulnerabilities existing in web applications or stored procedures in the back-end database server. It allows attackers to inject crafted malicious SQL query segment to change the intended effect, so that attacker can view, edit or make the data unavailable to other users, or even corrupt the database server. When an application becomes susceptible to SQLI Attack (SQLIA), attacker can get total control and access to database [1]. A successful SQLIA can read sensitive data from database, modify database data (insert/update/delete), execute administration operations on database (such as shut down DBMS and make it unavailable), recover the content of given file present on DBMS file system and in some cases can also issue commands to operating system. This research paper implements various attacks that can be performed on SQLI vulnerable web applications and also implements existing SQLI countermeasures in Software Development Life Cycle (SDLC) to check the effectiveness of them. Section II reviews the literature for known SQLI Vulnerabilities. Section III gives the brief overview of SQLI vulnerability and its counter measures in development life cycle. Section IV implements various attacks in SQLI vulnerable web application. Section V tests for the effectiveness of countermeasures and result are shown with the help of vulnerability scanners. In this research work, vulnerability scanner OWASP-ZAP is used to confirm the SQLI vulnerability. This vulnerability scanners is available with operating system Kali Linux and also freely available at www.owasp.org. Kali Linux is an open source project that is maintained and funded by Offensive Security, a provider of world-class information security training and penetration testing services. Section VI concludes the result and gives the future directions.

In the literature many countermeasures are suggested by various researchers and techniques are proposed to avoid these attacks but their effect is not presented. This research work is an effort to check the effectuality and potency of the available countermeasures of the SQLI vulnerability.

## II. RELATED WORK

SQLIA is the most popular, challenging and serious threatening attack. Year after Year, it is ranked as the top security vulnerability of the Internet which is responsible for countless data breaches. This vulnerability was first documented by Jeff Forristal in 1998[2]. Since then, lot of

 

research in the field is done by Industry and academic experts. There are many methodologies and algorithms suggested in [3].Several techniques are proposed to provide a solution for SQLIAs. This section gives the brief of the some similar work done in the area.

In 2006, Ke Wei et al. [4] has suggested that by using SQL injection attacks, an attacker could thus obtain and/or modify confidential/sensitive information. They also suggest that an attacker could even use a SQL injection vulnerability as a rudimentary IP/Port scanner of the internal corporate network. They proposed a novel technique to defend against the attacks targeted at stored procedures. This technique combines static application code analysis with runtime validation to eliminate the occurrence of such attacks.

In 2008, Mehdi Kiani et al.[5] describe an anomaly based approach which utilizes the character distribution of certain sections of HTTP requests to detect previously unseen SQL injection attacks and their practical results suggest that the model proposed in this paper is better than existing models at detecting SQL injection attacks.

In 2010, Ivano Alessandro Elia et al. [6] present an experimental evaluation of the effectiveness of five SQL Injection detection tools that operate at different system levels: Application, Database and Network. To test the tools in a realistic scenario, Vulnerability and Attack Injection is applied in a setup based on three web applications of different sizes and complexities. Based on experimental observations they underline the strengths and weaknesses of the tools assessed.

In 2011, Kai-Xiang Zhang et al. [7] suggest SQL injection attacks, a class of injection flaw in which specially crafted input strings leads to illegal queries to databases, are one of the topmost threats to web applications. Based on their observation that the injected string in a SQL injection attack is interpreted differently on different databases, they propose a novel and effective solution TransSQL to solve this problem. TransSQL automatically translates a SQL request to a LDAP-equivalent request. After queries are executed on a SQL database and a LDAP one, TransSQL checks the difference in responses between a SQL database and a LDAP one to detect and block SQL injection attacks. Their Experimental results show that TransSQL is an effective and efficient solution against SQL
injection attacks.

In 2012, RamyaDharam et al. [8] present a framework which can be used to handle tautology based SQL Injection Attacks using post-deployment monitoring that help to detect and prevent tautology based SQL Injection Attacks.

In 2012, TIAN Wei et al. [9] discuss how to generate more effective penetration test case inputs to detect the SQL injection vulnerability hidden behind the inadequate blacklist filter defense mechanism in web applications. They propose a model based penetration test method for the SQL injection vulnerability. Their Experiments show the penetration test case generated by their method can effectively find the SQL injection vulnerabilities hidden behind the inadequate blacklist filter defense mechanism thus reduce the false negative and improve test accuracy.

In 2013, Amir Mohammad Sadeghian et al. [10] suggest that a successful SQL injection attack interfere Confidentiality, Integrity and availability of information in the database. Based on the statistical researches this type of attack had a high impact on business. Finding the proper solution to stop or mitigate the SQL injection is necessary. To address this problem security researchers introduce different techniques to develop secure codes, prevent SQL injection attacks and detect them. They present a comprehensive review of different types of SQL injection detection and prevention techniques. They criticize strengths and weaknesses of each technique.

In 2013, Amir mohammad Sadeghian et al. [10] first they provided background information on this vulnerability. Next they present a comprehensive review of different types of SQL injection attack. For each attack they provide an example that shows how the attack launches. Finally they propose the best solution at development phase to defeat SQL injection and conclusion.

In 2014, Manju Kaushik and Gazal Ojha [3] discuss and comopare the existing methodologies for detecting and preventing SQLIA in order to design better attack detection and prevention methods in future. For better security they also discuss encryption and decryption techniques.

Also many researchers [11], have classified the SQLIAs based on the goal, motive, intention of the attacker or attack methodology used in order to understand the attack in better way.

## III. SQL I VULNERABILITY AND ITS COUNTERMEASURES

Vulnerability is a weakness in the application which can be a design flaw or an implementation bug. An attacker can use such vulnerabilities, to harm the stakeholders of an application. SQL Injection Attack, Cross-Site Scripting (XSS), Cross- Site Request Forgery (CSRF), Broken Authentication and Session Management are some of the application layer vulnerabilities targeting most of the current web applications [12]. According to reports that are provided by OWASP and WHID , among all these attacks SQLIA and XSS are very common. Also according to our previous research, SQLIA is one of the major attacks after Defacement and followed by XSS, Account Hijacking and DDoS(Distributed Denial of Service) Attack[13]. SQLIA is considered a severe of attack affecting confidentiality, integrity and availability of information. SQL injection vulnerability is a type of attack which adds Structured Query Language code to a web form input box to gain access or make changes to data. By using this vulnerability an attacker can send his commands directly to web application's underlying database and destroy functionality or confidentiality. If the injection is successful, an attacker can even read sensitive data from the database, modify database

 

data (DML Commands, execute administrative operations on database such as shut down the Database), and in some cases issue commands to operating system.

SQLI attacks have been around years now and lot of research in the field has been done by Industry and academic experts. In literature, there are many methodologies, algorithms and techniques proposed to provide a solution for SQLIAs. Analysis of SQLI attacks reveal that they are caused due to improper coding of web applications and inability to filter or sanitize input [14,15]. So, here known SQLI countermeasures and mitigation techniques from various researchers are classified in phases of SDLC in this section.

*Design Phase*

- Use minimum text boxes and try radio buttons/drop down list/check boxes instead [16].
- Use principles of least privileges and disable default accounts and passwords [1,17,18]. Also use Read only views for SQL statements that do not require any modification.
- Choose names for tables and fields that are not easy to guess.
- Identify the list of SQL statements that will be used by application and only allow those.

*Coding Phase*

- Santize/Validate Input by ensuring data is properly typed and does not contain escaped code [16-21]. Validate inputs with Data Type, Data Length and Data Format [1,22].
- Validation of all inputs must be done at both client and server side [1,17].
- Encode string in such a way that all meta-characters are interpreted by the database as normal characters [1,19,21,22].
- Use Stored procedure with static SQL wherever possible[1,17,18,21]
- Use parameterized queries instead of dynamic queries. Use Prepared statements in programming languages like Perl, Java [1,18,20].
- Use POST method instead of GET method for form submission[16]
- Ensure that Error Messages do not disclose any internal database structure, table names, or account names. Use proper error handling mechanism (Custom errors) also keep error messages and usable[1,20,21].

*Testing Phase*

- Conduct penetration tests against applications, servers and perimeter security[1].

*Configuration &Implementation Phase*

- Install the database on different machine than Web server or Application server .
- Update and Patch production servers (including operating system and application) [1,2,20].
- Disable potentially harmful SQL stored procedure calls[16].
- Delete system stored procedures [17].
- Delete/Disable unnecessary stored procedures/prepared statements .

## IV. VARIOUS ATTACKS WITH SQLI VULNERABILITY

All organizations who maintain a web presence are at risk of being attacked. However, the level of risk is different for each organization with respect to intellectual property or personally identifiable information stored by the organization. The purpose of a web based attack is significantly different than other attacks. SQLIA is most commonly associated with extraction of valuable data through web applications. SQLI vulnerability is also utilized as a platform for launching other types of attacks. The other types of attacks include Denial of Service (DoS), Defacement, Account Hijacking and Authentication Bypassing. This section concisely describes the attacks and performs each of them successively on a web application. The consequences of each attack possible with this vulnerability can be visualized from this section.

For the case study of these attacks, a Giftshop web application is selected. The selected web application is vulnerable to the SQLI attacks. Above described attacks are performed on Giftshop web application manually as well as with the help of automated tool SQL Map. SQL Map is an automated tool that performs SQLI attack and is capable of capturing active database management system fingerprint, enumerating entire database and much more [2].

*A. Database Fingerprinting*: This is actually a pre-attack preparation by an attacker and this type of attack is performed by entering some inputs which results in as an illegal or logically incorrect queries. The error messages reveal the table and column names that cause error and attacker can come to know about the database used in the backend server.

For database fingerprinting in Giftshop application, vulnerable columns are found using *'order by '* clause and then database, its version are displayed at vulnerable column. Database, version and tables are revealed with just a single command in automated tool SQL Map as shown in figure 1.

 

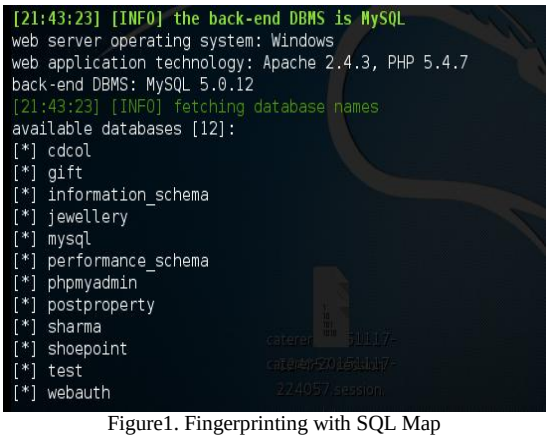


Figure1. Fingerprinting with SQL Map

*B. Authentication Bypass:* In this attack, an attacker exploits an input field that is used in a SQL statement's 'WHERE' conditional part. For example, in login form of the website that takes username and password parameter to enable access to certain section of website by validating entries in the back -end database. When attacker enters the username as 'abcd' or 1=1 - -' and password ='passwd', then the SQL Query becomes:

*Select * From Users WHERE* username = 'abcd' or 1=1 - -' and password ='passwd'

Double hyphens character is interpreted as a comment by the SQL Server, and everything after '- -' is ignored. Since 1=1 is always true so login is always validated. Giftshop application is also found vulnerable and allows authentication bypass.

*C. Injection with UNION Query:* In this attack, an attacker exploits the vulnerable parameter to change the data set returned for a given query. It is extraction of data from table but not as the intention of developer.

Information about the database, its version and users in the backend can also be revealed with the help of UNION query. Also the extraction of the data from table is possible as shown in figure 4.

*D. Account Hijacking:* This attack lets the attacker gain access to administrative or user credentials. This is again extraction of data from table. As figure 6 depicts the extraction of the data from the table, and the credentials received this way with SQLI lead to account hijacking as administrative password can be changed by the attacker after logging or even deleted from the database.

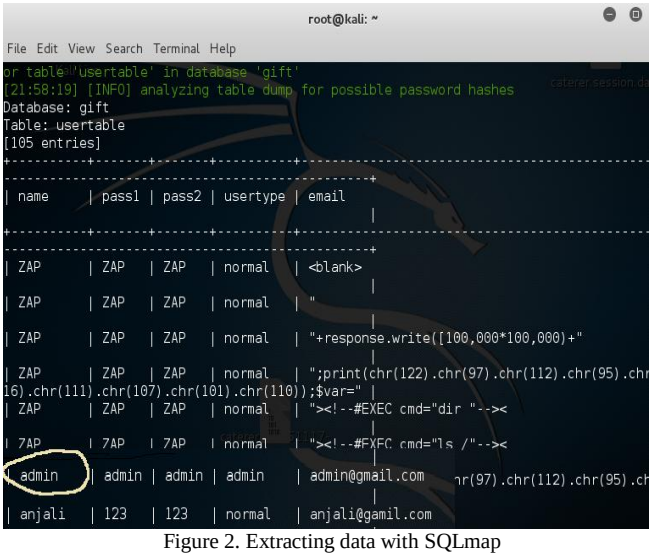


Figure 2. Extracting data with SQLmap

*E. Denial of Service (DoS):*This attack is to halt the web application by shutting down the backend database or consuming precious CPU time by sending database into time consuming loops over lots of data.DoS is most likely the well-known of all application attacks.

For the Giftshop application, Denial of Service attack was performed using two methods: Firstly by shutting down all the applications running on the remote system. The was done by executing *shutdown/f* command in the shell uploaded with the help of SQL map. Secondly by renaming the directory containing giftshop application files. This was again the execution of command on the shell as Shell on remote operating system is the complete takeover of the system/server revealed in figure 3. Thus, in result of both methods, Giftshop application was unavailable.

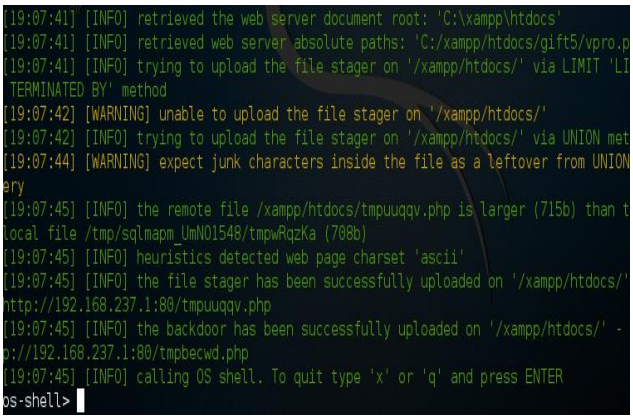


Figure 3. Shell Uploading with SQL map

*F. Defacement:* In this attack, an attacker alters the content of web site with offensive or erroneous graphics and/or text. An attacker can also change the appearance of the page or silently redirect a client to malware hosting server.

 

In the Giftshop, changing the content of web page was also possible through shell as done for previsously for DoS. But here manually a file was created and uploaded to the server with SQLI vulnerable URL (Uniform Resource Locator) as clear from figure8.


Figure 4. Writing File through URL

## V. EFFECTIVENESS OF SQLI COUNTERMEASURES

This section describes the results of the SQLI countermeasures on the Giftshop application. This application has been edited and implemented using the countermeasures specified in section III during the development cycle.

Firstly the Login page related database, table name and privileges has been changed as per design phase countermeasures. Then in the coding phase, input has been properly sanitized and validated at both client and server side. Also parameterized queries and prepared statement has been used as:

*$dbh->prepare("select * from usertable where email=? and pass1=?")*

Error messages has been handled with proper care.Similar changes has been done with other web pages of the Giftshop application. After these changes in design and coding of web pages and related database, againG iftshop application has been tested against SQLI attacks manually and with the help of vulnerability scanners. Result of scanning of web application with OWASP-ZAP (before and after the countermeasures implementation) are shown in figure 5 and figure 6.

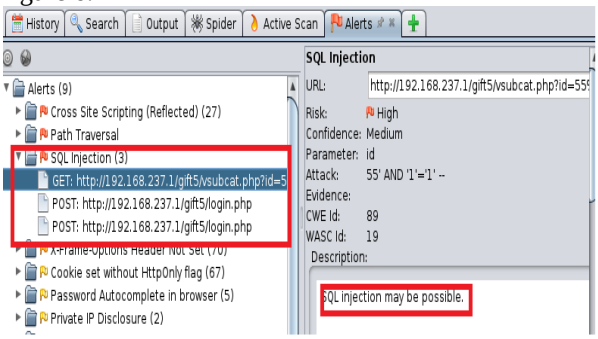


Figure 5. OWASP -ZAP Scan Result(Before)

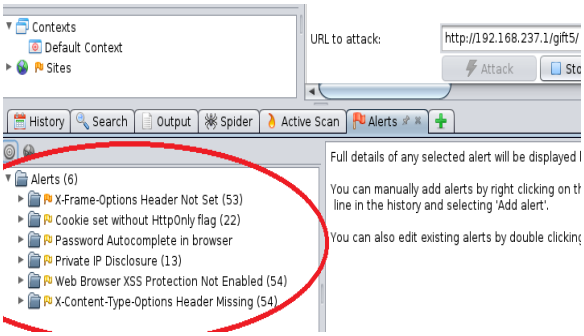


Figure 6. Scan Result with OWASP-ZAP(after)

Also the web application was tested manually against the various attacks like Database fingerprinting, Defacement, Account Hijacking and UNION query but it showed no response to them. Thus web application is now safe from attacks like Database fingerprinting, Account Hijacking, Defacement, Injection with UNION query and DoS due to SQLI vulnerability.

## VI. CONCLUSION AND FUTURE SCOPE

In this paper, we have studied and implemented the various attacks possible with SQLI vulnerability in web applications. Thus the known countermeasures of this vulnerability are classified in the SDLC fashion in section III and their effectiveness is checked with vulnerability scanner. Section IV has performed the various possible attacks on the web application w.r.t SQLI vulnerability. Further the web application is edited in order to implement the countermeasures identified during the development cycle and check the usefulness of these. Result of vulnerability scanner before and after the implementation of respective countermeasures in section V reveal that if applications are developed with security in mind from the beginning of SDLC, then many attacks on web applications can be avoided almost without any extra effort and time. Thus only need of the time is to aware the developers with known countermeasures and their effectiveness. In future, other vulnerabilities and their corresponding attacks can be implemented to make web applications more safe and secure.

## REFERENCES

[1] W.K. Torgby, N.Y.Asabere."*Structured Query Language Injection (SQLI) Attacks: Detection and Prevention Techniques in Web Application Technologies*". International Journal of Computer applications Vol. 71, Issue.11 , Pp 29-40.ISSN: 0975-8887, 2013.

[2] M. Gandhi. and J. Baria. *"SQL Injection Attacks in Web Application".* International Journal of Soft computing and Engineering (IJSCE), Vol2, Issue 6 ( Jan 2013). 189-191. ISSN:2231-2307. 2013.

[3] .Kaushik and G. Ojha." *SQL Injection Attack Detection and Prevention Methods :A Critical Review*", International Journal of Innovative Research in Science, engineering and Technology (IJIRSET), Vol3, Issue 4 .pp 11370-11377. ISSN: 2319-8753, 2014.

 

[4] K.Wei, M.Muthuprasanna and S.Kothari."*Preventing SQL injection Attacks in stored Procedures"*. In Software Engineering Conference , Australia,2006.

[5] I.A.Elia, Fonseca,Vieira,"*Comparing SQL Injection Detection Tools Using Attack Injection: An Experimental study*" in IEEE 21st International Symposium on Software Reliabiliry Engineering(ISSRE).pp 289-298,November 2010.

[6] K.X.Zhang, C.J. Lin, S. Chen, Y. Hwang. "*TransSQL:A translation and Validation based solution for SQL Injection attacks",* In first international conference on Robot, Vision and Signal Processing, pp248-251.November 2011.

[7] R.Dharm, Shiva.,"*Runtime monitors for tautology based SQL injection attacks*", In international conference on *cyberSec,pp.* 253-258. June 2012.

[8] T. Wei,Y.J.Feng,X.Jing. " *Attack Model Based Penetration Test for SQL Injection Vulnerability*", In IEEE 36th annual Computer Software and Applications Conference Workshops,pp. 589-594. July 2012.

[9] A.Sadeghian,Zamani, Manaf.,"*A Taxonomy of SQL Injection Detectionand Prevention Techniques.*", In International Conference on Informatics and Creative Multimedia.pp. 53-56. September 2013.

[10] Aldar C.F.Chan, "A Security Framework for Privacy Preserving data aggregation in wireless sensor networks", ACM Transactions on sensor networks. Vol 7, Issue 4, 29-40. DOI: 10.1145/1921621.1921623

[11] R.Piplode,P.sharma and U.K.Singh,"*Study of Threats, Risks and Challenges in Cloud Computing",* International Journal of Scientific Research in Computer Science and Engineering, Volume 1, Isuue 1, 2013.

[12] M. Shema. "*Seven Deadliest Web Application Attacks",* Elsevier Inc., pp47-69. ISBN-9781597495431,2010.

[13] D. Kaur, P. Kaur. "*Empirical Analysis of Web Attacks*". In Procedia of Computer Science. Elsevier Publications. Volume 78, pp. 298-306. DOI:10.1016/j.procs.2016.02.057, 2016.

[14] S. Junaid. "*Analytical Study of Common Web Application Attacks*". International Journal of Advanced Research in computer engineering & Technology (IJARCET)", Vol.3, Issue3, 611-617.

[15] G. Parmar, K.Mathur. "*Proposed Preventive measures and strategies Against SQL injection Attacks*". Indian Journal of Applied Research, Vol.5, Issue 5,pp 664-671. ISSN- 2249555X, 2015.

[16] S. Madan, S. Madan. "*Bulwark Against SQL Injection attack – An Unified Approach*". International Journal of Computer Science and Network Security(IJCSNS), Vol. 10 No.5.pp 305-313. 2010.

[17] Mahapatra and S. Khan. "*A Survey of SQL Injection Countermeasures*", International Journal of Computer science &engineering(IJCSES) Vol.3, No.3,pp.55-74. DOI : 10.5121/ijcses.2012.3305 55, June 2012

[18] William, Jeremy and Alessandro. "*Comparing SQL Injection Detection Tools Using Attack Injection: An Experimental study*" in IEEE 21st International Symposium on Software Reliabiliry Engineering(ISSRE).pp. 289-298 , 2010.

[19] S. Kalaria and M.Vivekanandan. "*Dark Side of SQL Injection*". In the proceedings of ASAR International Conference, Banglore, Pp 67-72. ISBN: 978-81-927147-0-7. April 2013.

[20] D.Gollmann. "*Securing Web Applications*".Article *in* ELSEVIER Information Security Technical Report Volume 13 Issue1. Elsevier Advanced Technology Publications Oxford, UK. 1-9.DOI: 10.1016/j.istr.2008.02.002

[21] U.Aggarwal, M.Saxena,K.S. Rana." *A Survey of SQL Injection Attacks*". International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE), vol.5, Issue 3. 286-289. ISSN:2277128X., March 2015.

[22] M.Kiani,Clark,Mohay, "*Evaluation of Anomaly Based Character Distribution Models in Detection of SQL Injection attacks*". In 3rd International conference on Availability,Reliabilty and Security, pp 47-55, 2008.

## Authors Profile

Daljit Kaur pursed M.Sc(Networking and Protocol Design) from Guru Nanak Dev University in 2006. She is currently pursuing Ph.D. and currently working as Assistant Professor in Department of Computer Science, Lyallpur Khalsa College,Jalandhar, affiliated to GNDU, Amritsar since 2009. She has published more than 15 research papers in reputed international journals including Elsevier, Thomson Reuters (SCI & Web of Science) and conferences including IEEE and it's also available online. Her main research work focuses on Web applications Security, Software Security and Secure development based education. She has 8 years of teaching experience and 5 years of Research Experience.

Dr. Parminder Kaur, is presently working as an Assistant Professor in the Department of Computer Science, Guru Nanak Dev University, Amritsar, India. She has completed her Ph.D. in the research area of Software Engineering from Guru Nanak Dev University Amritsar in the year 2011. She has a teaching experience of about twenty three years. Her research interests include OpenSource Software, Web Usability and Optimization, Web Services, Software Security, Software Evolution, Version Management in Distributed Environment and Image Processing. She has around 40 publications in International/National Journals as well as Conferences. She has contributed six book chapters. She is a life member of Punjab Science Congress and Computer Society of India.

 