Detection and Prevention of SQL Injection Attacks By Request Receiver, Analyzer and Test Model

Submitted By:

Nosheen Kanwal

512/FBAS/MSCS/F08

Supervision:

Dr. Muhammad Sher

DEAN, Faculty of Basic and Applied Sciences

Department of CS and SE

Faculty of Basic and Applied Sciences

International Islamic University Islamabad

Accession No. IH-13614

K

[illegible]

size
300dpi
- ocr
- small pdf.

MS
004

[illegible] D

[illegible] Science

Computer System

Dedication

Dedicated to the Holy Prophet Muhammad (Allah's grace and peace be upon Him). And also my beloved parents, teachers and my friends who support me a lot and their prayers pay the way to success in my life.

ABSTRACT

During the last few decades use of web and database applications extend to a large extent. With the increase in use of the web based applications there was also increase in use of online database applications. When there is growth in one technology the associated problems also arises. Out of many problems and threats to the database applications one potential problem is of SQL injection. Many solutions were also proposed with passage of time. In this work I also proposed solution which is based on Request, receiver and analyzer model. The proposed solution is a novel approach to detect and block SQLIA in web applications. The proposed solution would also implement in a tool named RAT that would be less costly in term of resource usage and efficient in terms of time and applicability against different techniques of SQLIA.

Key Words: SQLIA, RAT

ACKNOWLEDGMENT

All praises to be on *Almighty Allah*, the most "REHMAN" the most "RAHEEM", the provider of hope, guidance and knowledge. Without HIS remembrance surely could not have overcome our moments of despair.

It is said in the Holy Quran:
"Does man think that he will be left uncontrolled, (without purpose)? Was he not once a drop of ejected semen? Then he became a clot, so He created and fashioned him and made him into two sexes, male and female. Is He who does this not able to bring the dead to life?" [Surah alQiyama: 36-40]

I thank to **The Holy Prophet Muhammad** (Allah's grace and peace be upon him). Without His blessings, I was unable to complete the thesis. I am also indebted to every one whose efforts helped me to make this thesis possible and I express my cordial and humble thanks for their untiring help and cooperation in completing this thesis.

I cordially regard the inspiration, prays, encouragement, and financial support of our loving and affectionate parents and family for their motivation in every aspect of my life.

FINAL APPROVAL

It is certified that we have read the thesis report, titled "**Detection and Prevention of SQL Injection Attack by Request Receiver Analyzer & Test Model**" submitted by **Nosheen Kanwal, 512/FBAS/MSCS/F08**. It is our judgment that this thesis is of sufficient standard to warrant its acceptance by International Islamic University, Islamabad for the degree of **MS in Computer Science.**

Committee:

External Examiner:
Dr. Nadeem Javaid
Assistant Professor CIIT
Department of Electrical Engineering
COMSATS , Check Shehzad Islamabad

[illegible]

Internal Examiner:
Mrs. Zareen Sharf
Assistant Professor
Department of CS and SE,
Faculty of Basic and Applied Sciences,
International Islamic University Islamabad

[illegible]

Supervisor:
Dr. Muhammad Sher
DEAN,
Faculty of Basic and Applied Sciences,
International Islamic University Islamabad

[illegible]

TABLE OF CONTENTS

ABSTRACT I

ACKNOWLEDGMENT II

FINAL APPROVAL III

TABLE OF CONTENTS V

LIST OF TABLES XIII

LIST OF ABBREVIATIONS XIV

CHAPATER 1: INTRODUCTION 1

1.1 Motivation and Challenges 1

1.2 Background 3

1.2.1 WEB Applications 4

1.2.2 WEB Security 7

1.2.3 Input Validation vulnerabilities 10

1.2.4 Mostly targeted organizations 13

1.3 Problem statement 14

1.4 Proposed Solution 15

1.5 Thesis outline 15

1.6 Research Methodology 16

1.6.1 Tool used 17

2.1 Introduction 18

2.2 Related Research 18

2.3 Limitation 24

2.4 Summary 24

CHAPTER 3: REQUIREMENT ANALYSIS 25

3.1 Problem Domain 25

3.1.1 SQL Injection Attack (SQLIA) Process 25

3.1.2 Consequences of SQL injection attacks 28

3.1.3 Most common vulnerabilities for SQLIA 29

3.2 Problem Scenario 30

3.2.1 Tautology Attacks 30
3.2.2 Union Attack 31
3.2.3 Logically Incorrect query Attack 32
3.2.4 Piggybacked Query 32
3.2.5 Alternate Encoding 33
3.2.6 Inference 33
3.2.7 Stored Procedures 33
3.3 Focus of Research 33
3.3.1 Status of SQLIA defense techniques 34
3.3.2 SQLIA detection and prevention tools and techniques analysis 36
3.4 Summary 50
4.1 Introduction 51
4.2 Request Receiver, Analyzer and Tester Architecture (RAT) 51
4.2.1 Request Receiver 51
4.2.2 Analyzer 51

4.2.3 Tester 52

4.3 Algorithm 53

4.4 Flow Chart 54

4.5 Limitation 54

4.4 Summary 55

CHAPTER 5: TESTING AND PERFORMANCE EVALUATION 56

5.1 Introduction 56

5.1.1 Simulation Model 56

5.1.2 Analysis 59

5.2 Test Scenarios 59

5.3 Performance and evaluation 61

5.4 Summary 65

CHAPTER 6: CONCLUSION AND FUTURE WORK 66

6.1 Conclusion 66

6.2 Achievements **67**

6.3 Future Work **68**

BIBLIOGRAPHY **69**

LIST OF FIGURES

Figure 1 : OWASP SQL injection ranking 1

Figure 2 : A simple web application scenario 5

Figure 3 : Interactions among 3-tier application 6

Figure 4 : A typical World Wide Web infrastructure 7

Figure 5 : The most widespread vulnerabilities in web applications (% Vulns ALL) . 8

Figure 6 : The probability to detect the most widespread vulnerabilities in web applications (% Sites ALL) 9

Figure 7 : Percent of vulnerabilities out of total number of vulnerabilities (% Vulns BlackBox & WhiteBox) 10

Figure 8 : Different class of vulnerabilities by percentage from 2001 to 2002 11

Figure 9 : Reported web attacks in 2008 12

Figure 10 : No of incidents by no of record leakages 13

Figure 11 : Mostly attacked organizations in 2008 14

Figure 12 : Percentage of vulnerability by class 3

Figure 13 : How web applications works 26

Figure 14 : Simple php code snippet that generate dynamic query for a log in input. 26

Figure 15 : Benign user input on a typical log in form... 27

Figure 16 : Malicious user input on the log in form.. 27

Figure 17 : Work flow 17

Figure 19 : No of defense techniques emerged over the time 34

Figure 20 : %age of detection principle used 34

Figure 21 : Detection principle usage over the time 35

Figure 22 : Percentage of analysis methods used... 36

Figure 23 : Analysis methods used over the time 36

Figure 24 : Home page of RAT implementation... 56

Figure 25 : Receivers page for evaluation of the model... 57

Figure 26 : Analysis result 58

Figure 27 : Administration rule module 58

Figure 28 : No of attacks detected by different in evaluation period 61

Figure 29 : No of attack protected by different tools during evaluation period 62

Figure 30 : Time taken for detection by different tools 62

Figure 31 : Time taken by different tools to prevent SQLIA 63

Figure 32 : Types of attacks detected by different tools 63

Figure 33 : Types of attacks blocked by different tools 64

Figure 34 : No of database supported by different tools 64

LIST OF TABLES

Table 1 : Evaluation data set.. 60

LIST OF ABBREVIATIONS

API	Application Program Interface
ACM	Association for Computing
AMNESIA	Analysis and Monitoring for
ANN	Artificial Neural Network
ASCII	American Standard Code for
ASP,	Active Server Pages
CSSE	Context-Sensitive String
DBMS	Database Management System
DOAJ	Directory of Open Access
DOM	Domain Object Model
DoS	Denial of Services
DREAD	Damage Potential,
D-WAV	Detection of Web Application
FCD	Frequency Character
HTML	Hyper Text markup Language

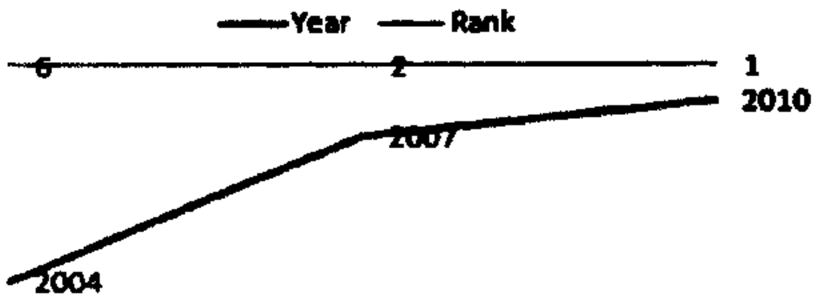
IEEE	Institute of Electrical and
IP	Internet Protocol
JDBC	Java Database connectivity
JSP	Java Server Pages
JVM	Java Virtual Machine
MS	Microsoft
MUSIC	Mutation-based SQL Injection
OWSAP	Open Web Application Security
PHP	Hypertext Preprocessor
phpBB	PHP Bulletin board
PQL	Program Query Language
RAT	Request Receiver, Analyzer and
SAFELI	SQL Injection Scanner Using
SQL-IDS	SQL Injection Detection System
SANIA	Syntactic and Semantic Analysis
SCC	Same Character Comparison

SPDL	Security policy description
SQL	Structured Query Language
SQLIA	SQL Injection Technique
CVSID	Software Code Vulnerabilities
USA	United States of America
WASC	Web Application Security
WASP	Protecting Web Applications
WAVES	Wave Application Vulnerability
WWW	World Wide Web
WebSSARI	Web application Security by
XSS	Cross-site scripting

CHAPATER 1: INTRODUCTION

During the last few years most of organization preferred to have web based application to have access to global market place and availability and access for all the time from anywhere. Despite having many advantages there is also numerous risks associated with the web applications. Web applications have to face many input vulnerabilities including the SQL injection attacks (SQLIA). SQLIA have become more popular among intruders due to improvements in its techniques over the years. Over the last few years SQLIA improved in term of vulnerability and emerged as one the serious threats to the web based data driven applications. In fact the open Web Application Security Project has placed SQLIA in top ten vulnerabilities for a web based application [1]. The next figure shows the SQLIA ranking in top ten over the years.

Figure 1 : OWASP SQL Injection ranking [2]



Similarly most of the software development companies stress the need to address the issue of SQL injection attacks. Due to attractive nature of database SQLIA attacks increasing day by day.

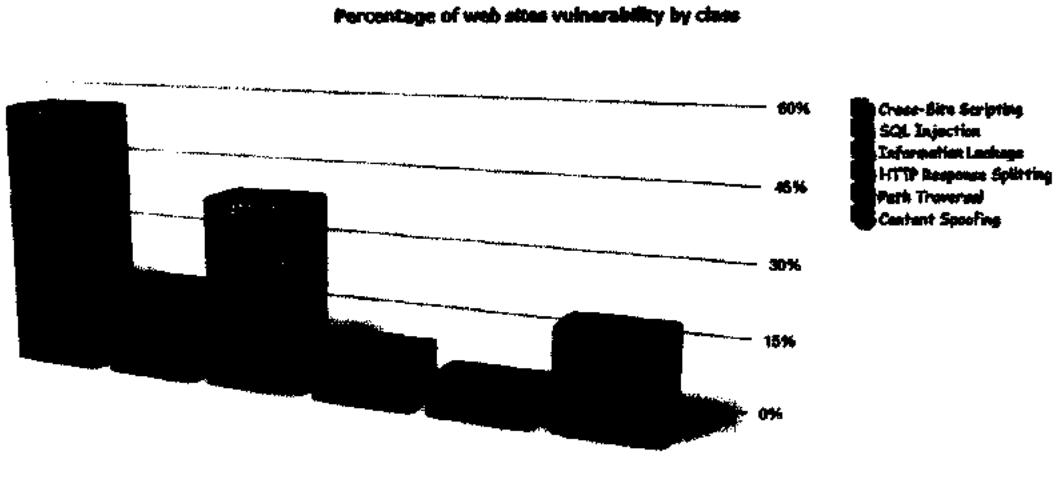
1.1 Motivation and Challenges

We are living in information age where information is most crucial and important for business existence and survival in competitive market place. Like other assets appropriate care also have to make for safety and security of information. Among many others SQL injections also proved to be one of major threats to the important assets of the today's organizations where data and information are very crucial for the survival of the organizations. Recently it is observed that SQL injections attacks and vulnerabilities increased to a large extent and increased to 156% up to the year 2012 [3]. Moreover SQLIA are easy to exploits that's the reasons they are used by predominantly used for different reasons. The most malicious use of SQLIA is in applications related to finance, confidential data, sabotage, and terrorism or simply for fun. Domain of application affected applications increasing days by day. With more and more types of application being get infected there is also improvements in sophistication of techniques employed for SQL injections and it is also becoming more and more common approach for many intruders.

With the increase in commonality, techniques and more application being affected there is need to develop a strategy and proper measure against such attacks. This is the reason why SQL injection detection and preventions is the agenda of modern research and focus of this study. In order to achieve this purpose many techniques have been developed and implemented but none of the solution is comprehensive enough to deal with the problem in an appropriate way. Most of the proposed solution suffer with performance or require change in source code that is a big problem in itself. I feel that there is a need of a solution that dealt with the problem in an accurate way but at the same time cost effective in term of time and resource usage with minimum modification in source code.

Among many other web vulnerabilities SQL injection attacks is among the top most vulnerabilities. The next stats from the OWSAP show the web application vulnerabilities percentages of different classes.

Figure 2 : Percentage of vulnerability by class [4]



From the above figure it is clear that SQLIA is one of the top three. This ranking may vary in different web security vulnerability ranking by different organization but at everywhere it is among the top ten and most of the time it is among the top five. Due to easy access to web and high rewards for attacks due to database access SQLIA is the proffered choice for the attackers. The next stats clearly show that SQLIA is also among the top when SQLIA opt to choose a technique for attack.

1.2 Background

Computer security also known as information security relates to preventing unauthorized use of data or data protection against unauthorized access. The information security is a broad concept and applied in contextual manner. In simplest form computer security is the detection and prevention of unauthorized use of computer. Prevention relates to preventing intruders for having access to any part of computer and detection relates to determining the whether someone attempt to breach the security and break into system being independent of attempt was successful or not and level of damage.

From very early days of computer and information the security of information is challenging task. It was proved to be very difficult to ensure that system do for what they intends to do. Now we are living in information age where most of the daily routine tasks are performed with the assistance of technology. One common misconception is relating to information security concerned with only limited to information processing with computers. This is not true. The information security definition describe that it concerned with all aspects of acquisition, processing, transmitting and storing of information whether it is through electronic means or on paper. According to U.S. National information Security glossary information security is;

"the protection of information system against unauthorized access or to modification of information whether in storage, processing or transit and against the denial of services to authorized users or provision of services to unauthorized users, including those measures necessary to detect, document and counter such threats." [5]

Now a day's most of information stored and transmitted electronically the information security aspects of computers have gained much attention. Now we are connected more in term digital world with assistance of computers. Now there are far more chances of intrusions and malicious attacks of wide variety. Now it is more crucial for use to be in a reliable and safety is more concern for us as compared to past. According to U.S. National information Security glossary Computer Security is;

"Measure and controls that ensures confidentiality, integrity and availability of the information processed and stored by a computer" [5]

But we use computer and information security in a more general way.

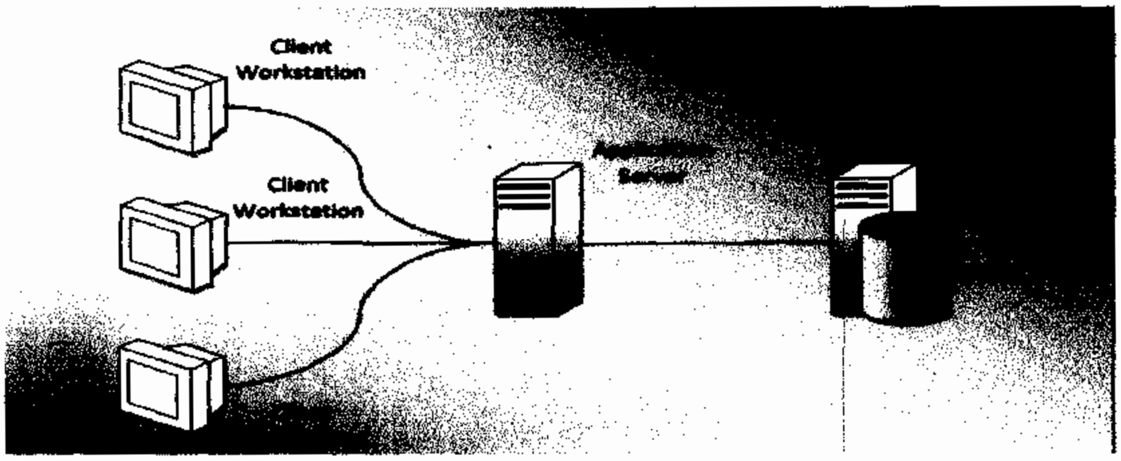
1.2.1 WEB Applications

Due to rapid growth of Internet and Web technologies organizations across the world rely on this medium to gain competitive edge and to attract global market place. Web technologies proved to be efficient, flexible and economical as compared to any other existing communication technology. Web technologies also efficient and flexible enough to improve inter departmental work across the world, worldwide exchange of information and services.

We are living in information age where Internet is indispensable part of our life. From government offices to kid's entertainment we rely in Internet and web technologies. Very important data stored and transmitted over the Internet. Web application is accessible by web through a network. There is trend in shifting from desktop to web based applications. Most of the application which was standalone has shifted to the web. Web based application differ from desktop application that they are accessible from anywhere through the web. Web applications are placed on a server and can be accessed from anywhere. [5]

A web application comprises of set of related web pages and program logic with associated database. User of that program can access the web application from anywhere. A web application is usually based on 3-tier model. 3- Tier application is the interface the second is the program logic and the third is the database. In web application the first tier is the browser on the client side, the second tier is the program logic reside on the server in the form of different scripts and program logic. The third tier is the database that accessed from a database server. A typical 3-tier application is shown in the next figure. Data moved across the tier through appropriate interfaces as shown in the next figure.

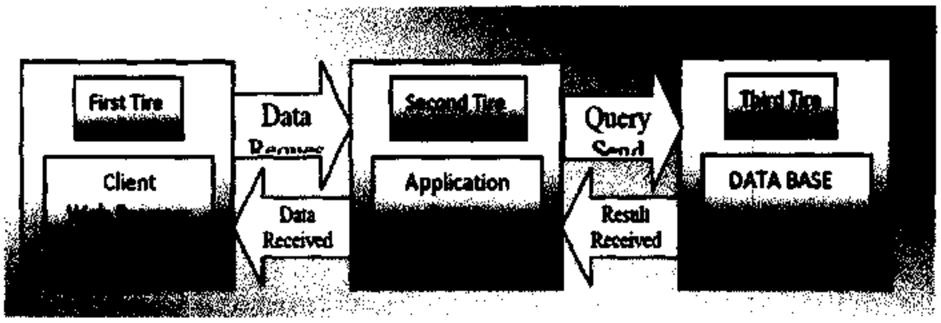
Figure 3 : A simple web application scenario [5]



The browser acts as interface for the user for interaction with the application. The web application server manages the business logic and act as intermediary between user requests and database access. The web server receives the input from both other tiers in the form strings. This flow of data among different tiers gives raise problem of input validation. The web application or middle tier also has to check the input from other tier in order to validate the input otherwise there may be severe security threats for the application. Failure to validate or sanitize the input can create serious security loopholes for intruders to have illegal access to the important assets of organization.

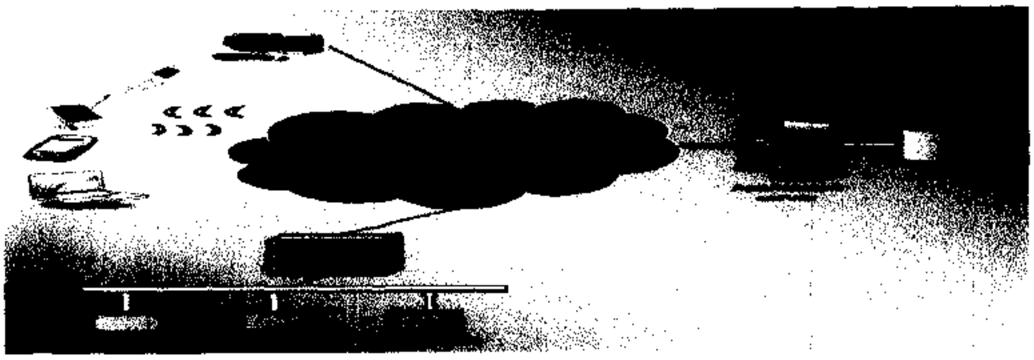
As we mentioned above there is trend in web application and desktop based application are being replaced by web based application. There are many reasons behind this shifting. For most and important is accessibility from anywhere and at all time. One more important reason is the availability of heterogeneous types of access devices in the form of mobile phones, PDA and laptop. In order to be compatible for providing access to these heterogeneous ranges of devices web platform is appropriate.

Figure 4 : interactions among 3-tier application [5]



But the major force behind the web based application is accessibility all around the globe for all the time. It help the organization to attract the clients across the world and be exploits the global market place. In order to place the web application World Wide Web (WWW) infrastructure is used as shown in the next figure;

Figure 5 : A typical World Wide Web infrastructure [5]

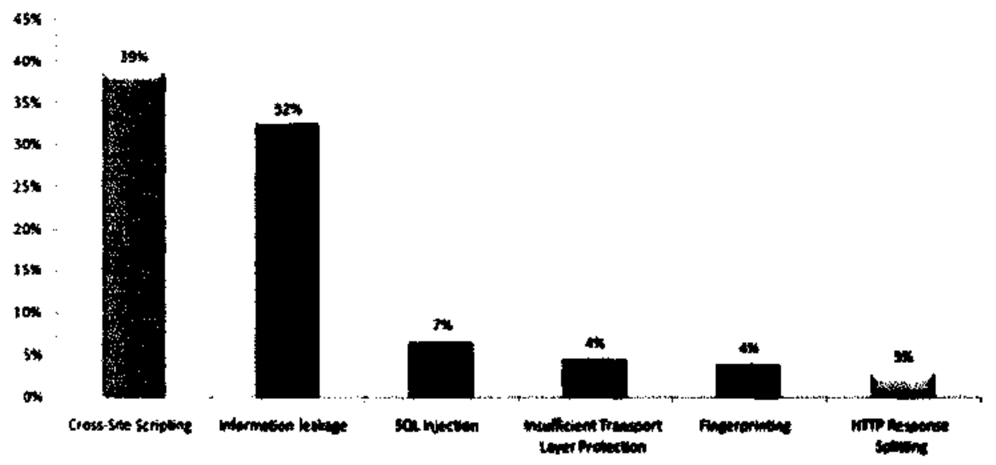


When there are numerous advantages of going for web based application there are also increase chances of vulnerabilities attacks of different types.

1.2.2 WEB Security

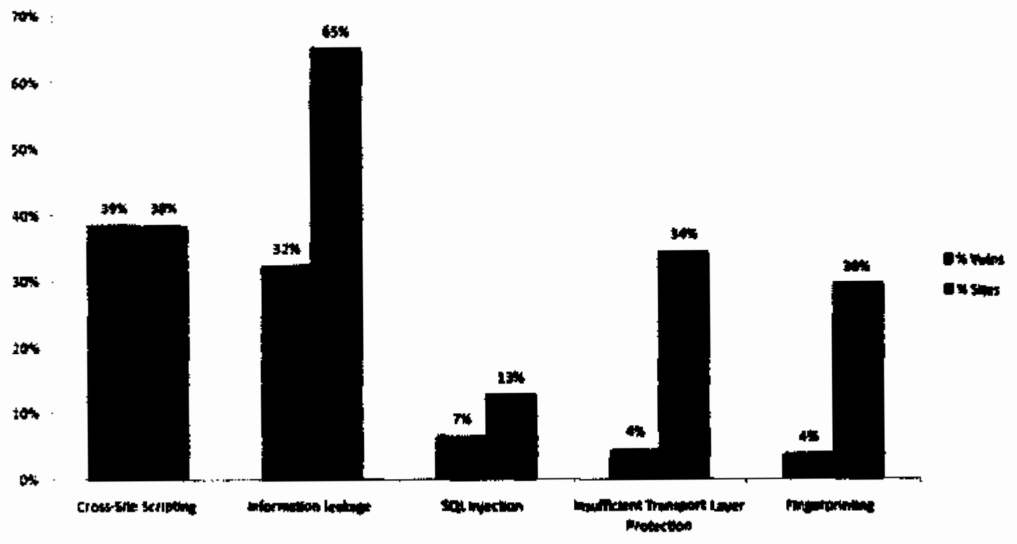
Over the last two decades there is tremendous growth of websites. From governments department to different types of organizations, agencies, banks and even Small to medium enterprises rely on web based applications for smooth running of their processes and transactions. Websites has become the important information release centers that manage large amount of data for sharing among billions of users over the Internet.

Figure 6 : The most widespread vulnerabilities in web applications (% Vulns ALL) [4]



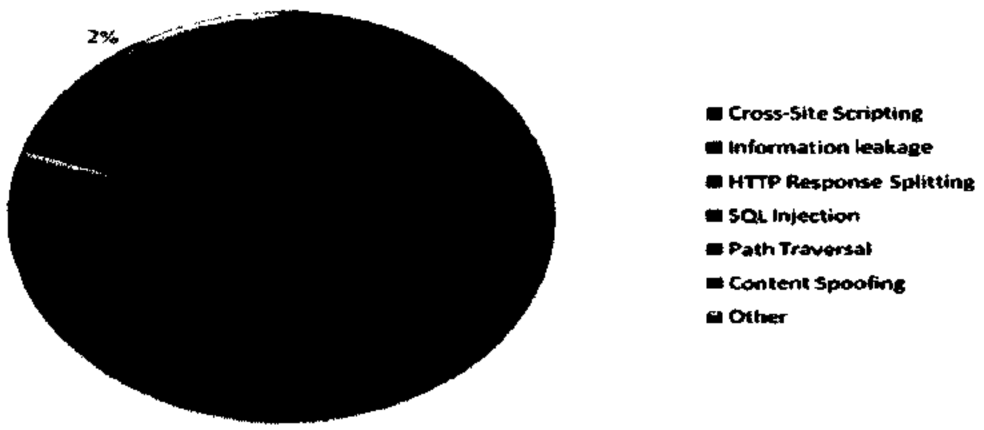
In order to ensure smooth running of the any organization that rely on web application the security of the web application is very critical. Though Internet and web technologies developed rapidly but due to complexity of network and varieties of vulnerabilities the appropriate level of security development does not took place. Wide varieties of threats still problem to development in this field. Though web security gained attention in last few years but a lot more have to done. Web security is mostly overlooked aspect in data security.

Figure 7 : The probability to detect the most widespread vulnerabilities in web applications (% Sites ALL) [4]



According to Web Application Security consortium (WASC) out of 12186 web applications of different types 97554 contains vulnerabilities with different levels of risks. According to WASC 49% of websites contains vulnerabilities of high risk levels. [4]

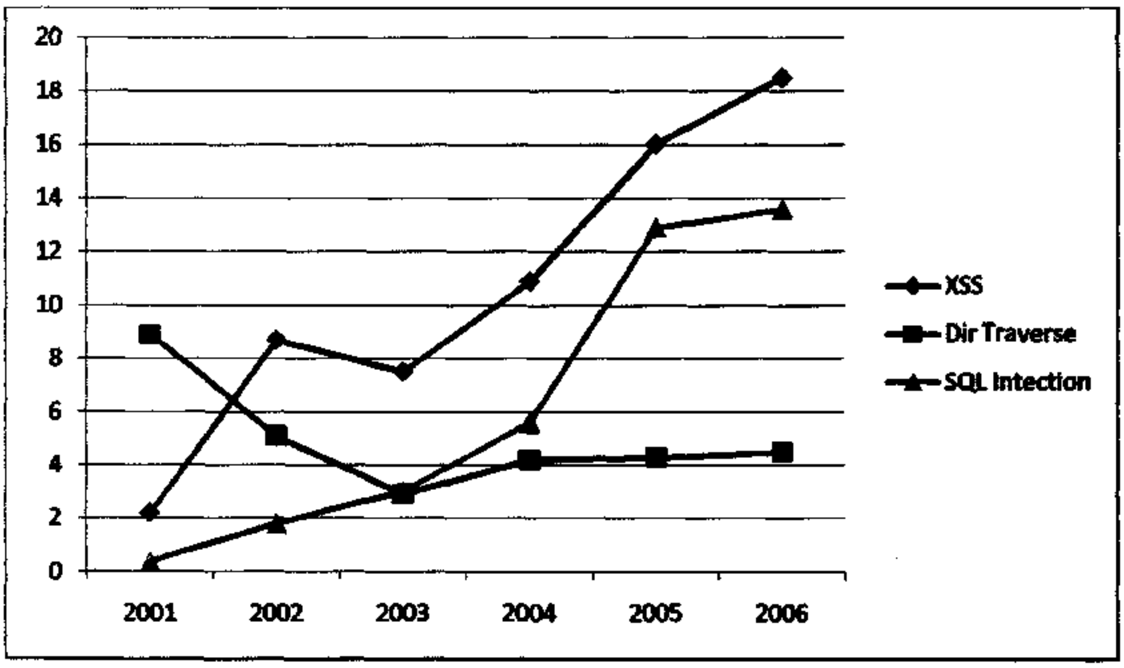
Figure 8 : Percent of vulnerabilities out of total number of vulnerabilities (% Vulns Black Box & White Box) [4]



1.2.3 Input Validation vulnerabilities

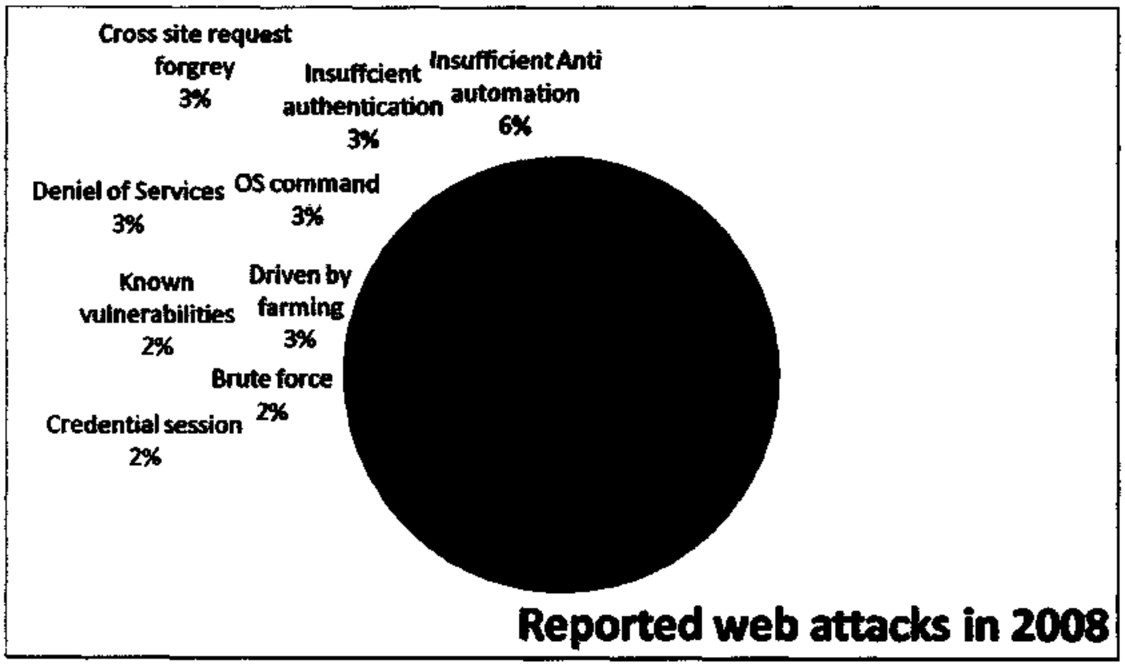
SQL injections attacks are among the most input validation attacks. In SQL injections the intruders want to access those HTML documents and database information that are not intended for them. This is possible because of available of low level API's with the help of low level string manipulations that are treated as isolated lexical entities. Especially in case of PHP where there is lack of sophisticated API and string is used for data and code representation. Some paths in applications code may exist that allow user unmodified or unchecked user input that can manipulate the database or HTML document in an unauthorized manner. Sometimes constraints are not strong enough that block illegal user inputs. In short SQL injections are integrity violations in which low level low integrity data is used in a high integrity channel, where the browser or database run a code that is intended for untrusted user but do so with the orders of application logic. In order to highlight the growth of web and associated problems the next figure describes a clear story. From 2001 to onwards different types of vulnerabilities escalates to high level including the SQL injection attacks (SQLIA).

Figure 9 : Different class of vulnerabilities by percentage from 2001 to 2002 [5]



In the above figure these classes of vulnerabilities are shown that had are more relevant to web security. SQLIA is consistently near the top of all vulnerabilities. Because of the easy access to the web and importance of database web security analyst speculate that the actual SQLIA vulnerabilities are far more than the reported. The next figure shows that this hypothesis is correct to a large extent. The data on the next figure shows the percentage of web attacks for the year 2008. [3]

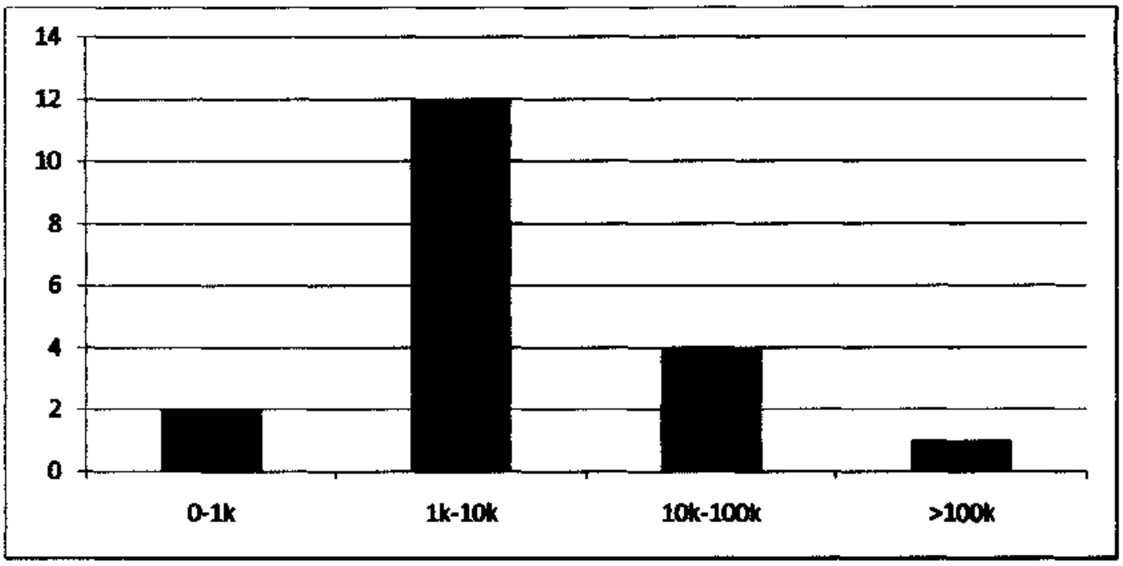
Figure 10 : Reported web attacks in 2008 [3]



Though many others types of vulnerabilities still unreported but important to note here is that SQLIA are 30% in year 2008. SQLIA can be severe as they intend to access the database and result in important information. Leakage, by pass authentications or even can add unauthorized accounts to the database. According to the Web hacking incident database fallowing ranges of data leakages against the number of incidents shown in next figure.

In order to ensure smooth running of the any organization that rely on web application the security of the web application is very critical. Though Internet and web technologies developed rapidly but due to complexity of network and varieties of vulnerabilities the appropriate level of security development does not took place. Wide varieties of threats still problem to development in this field. Though web security gained attention in last few years but a lot more have to done. Web security is mostly overlooked aspect in data security.

Figure 11 : No of incidents by no of record leakages [6]

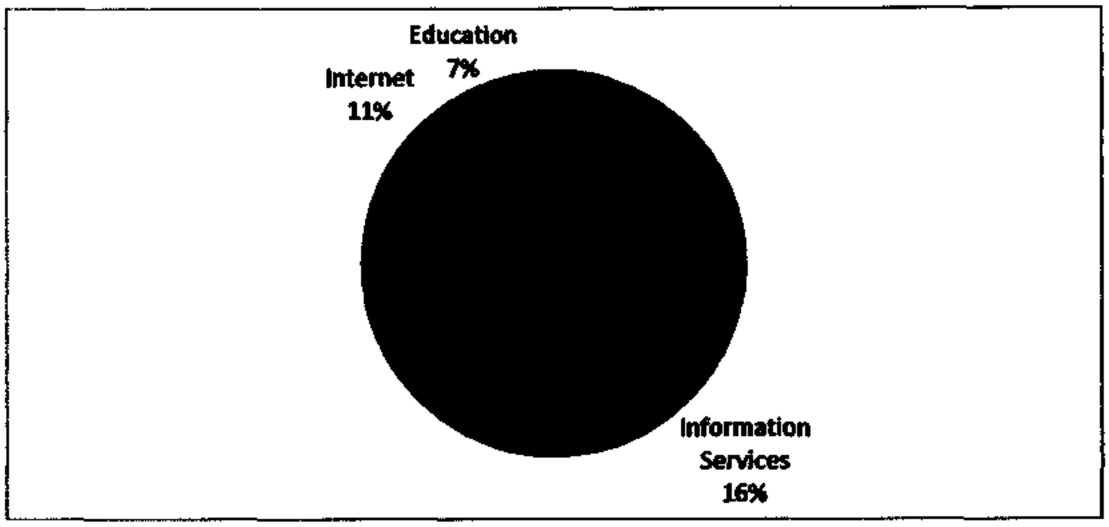


It is observed that about 1.5 million pages being get affected by the SQLIA. [6] On 25th April 2008 Washington post write that hundreds and thousands of web sites being get hacked and many of them were from USA. Hackers exploit the security flaws of Windows and succeeded to seed malicious code. According to Finnish ;(Maker of F-Secure antivirus) about half a million of web pages hacked and serving malicious code. Most of attacks leaked one thousand records and even one leaks more than 100000 records. This reveals the severity of the issue and the number of user being got affected.

1.2.4 Mostly targeted organizations

Mostly government and associated websites are the targets. According to the Web Hackers Incident Database government organizations are the primary targets with educations at number six. There may be some ideological reasons behind this. One major factor may be that public disclosure requirements of governments are much broader than the commercials organizations.

Figure 12 : Mostly attacked organizations in 2008 [6]



On the commercial side the e-commerce websites, media and pure Internet services providers are the main targets.

1.3 Problem statement

From the introduction we can easily infer that SQLIA is one of the most profound web security issues and it tends to rise with the increase in data driven web based application over the Internet. Due to easy access of the web and potential of database access SQLIA rate tends to rise from 2000 to till dates. Though different solutions proposed with time but many suffered with performance issues or require a lot of changes.

There is a need to make the web application more secure so that increasing number of organization that opts for web application feel secure. In recent years research work focused on the issue with the emergence of new defense techniques and solutions. On the contrary the SQLIA techniques also tend to more sophisticated resulting into still an issue. Keeping in view of importance of the subject we try to investigate and answers fallowing research questions;

RQ.1. What are the SQLIA techniques employed by the attackers and what are the existing defense mechanism?

RQ.2. How proposed solution work against different techniques of SQLIA?

RQ.3. Up to what extent the proposed solution effective as compared to existing solutions?

1.4 Proposed Solution

We proposed a model that is RAT (Request Receiver, Analyzer and Tester) is a model based approach that does not require change in source code. It has certain advantages over the previous develop techniques and model that it can be implemented in any language on any sort of platform. It also has advantaged that it can be used for any sort of database.

1.5 Thesis outline

I divided this work into seven chapters starting with introduction. The first chapter introduction set the foundation of study with basics about security issues in general and SQL injections in particularly. Introduction chapter also describe the motivation behind this work, scope of the study, significance and problem statement.

The second chapter named "literature review" includes the previous work in the domain to streamline our direction and pave our way in right direction.

The third chapter named "Research ***methodology***" sets our research basics pillars. It describes the methods used for research and data interpretations method.

The fourth chapter "Review of existing solutions" includes the short review of existing solutions and associated problems.

The fifth chapter named "***Proposed solution***" is the bread and butter of our study. It discusses the proposed solution in details.

The sixth chapter named "***Evaluation***" deals with the performance and accuracy evaluation of the proposed solution. Proposed solution after implanting evaluated against different criteria's for efficiency and accuracy.

The seventh and last chapter named as "***Conclusions and future work***" concludes our findings and also recommends future work in these lines.

This chapter deals with schema for our research and study. We want to use a comprehensive literature review and experiments for our research. This chapter would include the source of literature review and experiment design for the research, the tools used and analysis basis for our study.

1.6 Research Methodology

This study would base on extensive literature review and experiment. We would start with a preliminary literature review to have insight into the problem in order to clearly understand the problem domain and its available solution. Literature review would be an ongoing activity throughout the research period in order to support our findings and to pave our way in the right direction.

Literature for our study would be selected against strict criteria regarding the relevancy to the problem being studies and from reliable source in shape of world renowned journals like ACM, DOAG, Science direct and IEEE journals with high impact factors. Most of the selected articles and research paper would be in the range of last five years publications. Relevancy to the subject is would be yardstick that guides our search and selection of articles for literature review.

Figure 13 : Work flow

In order to evaluate the proposed solution we take help from experimentation of the proposed solution against different criteria in term of time taken, no of SQLIA detected and blocked and types of SQLIA detected and blocked.

In order to implement the solution experiments also part of the work with widely deployed tools like SQL server, ASP and HTML pages.

1.6.1 Tool used

I would run simulator on windows 7 environment with visual studio version 10.

1. Microsoft Windows 8
2. Visual Studio version 2012
3. .Net Frame work version 3.5
4. MS SQL Server
5. ASP. Net

In order to evaluate the propose solution we would use fallowing parameters

1. Efficiency in term of time taken to detect and block SQLIA
2. No of SQLIA detected and blocked
3. Types of SQLIA detected and blocked

CHAPTER 2: LITERATURE SURVEY

2.1 Introduction

Literature review would be essential part of our study. It set the foundation of our work to have deep insight into the subject matter. The major source of publication would be science direct, DOAG, IEEE explore and ACM library.

2.2 Related Research

This section of study contains literature review in order to have deep insight into the problem and issue related to our work. This section provides a foundation for our study by providing deep knowledge about the problem domain. In order to meet objectives of the study a reviewed number of article selected against the criteria mentioned in the research methodology. I tried to concentrate more on article from journal like ACM, IEEE etc.

W.G. Hal fond et all contributed a lot in the field of SQL injection. They review different SQLIA techniques and also analyze different defense techniques with strength and weakness of each technique. They proposed a model based technique to detect and prevent SQLIA on web. Their technique named as AMNESIA use both static and dynamic analysis and also evaluate their proposed solution.

XuePing-Chen [11] discuss the SQLIA attack process and its implementation. The author suggests the use of strong authentications and coding technique in order to avoid SQLIA.

M. Bravenboer et al. [12] present a natural style code that resist injections. This technique embeds the SQL into java in order to generate automatic code. This is a generic technique as host and guest language can be combined in an easy way.

T. M. Chen and J. F. Buford exploit the application layer to create "honey pot" in order to gain insight about different types of SQLIA. This "honey pot" helps to prevent the attacker to have access to the database or operating system. This technique emulates the common defense technique against the SQLIA.

A. Ciampa et al. [15]present an approach and a tool named "V1p3R ("viper")" that test the web applications for SQLIA injections. The proposed approach relies on pattern matching and output produced by the test.

D. Das et al. [16] classify different types of SQLIA techniques employed and on the type and extent of different types of vulnerabilities by different SQLIA.

M. Ficco et al. [17]propose a systematic technique based on the anomaly model. This technique combines different techniques to get data from different sources.

W. G. J. Halfond et al. [9]Proposed a model based technique to detect and prevent SQLIA on web. Their technique named as AMNESIA use both static and dynamic analysis and also evaluate their proposed solution.

Sangita Roy et al. [18] suggest a new technique that use filter to check SQLIA. In this approach URL filter used to validate the user inputs. This technique used java serve lets for the filters and is claimed to be more efficient and scalable.

Xiang Fu and Kai Qian[19] present evaluation of a toolset called "SAFELI" in order to detect vulnerabilities in SQL injection. Their technique use static analysis with the help of symbolic instruction in order to detect security vulnerabilities.

Z. Jan et al. [20]present a technique based on use of parameterized cursor that help to detect and prevent SQLIA. This approach can be used in any database.

M. Junjin [21] Proposed an approach detect the input vulnerabilities with the help of analysis, and runtime testing. This is implemented in SQLInjectionGen.

K. Kemalis and T. Tzouramanis [22] propose a new approach that use syntax of the query. Proposed technique is effective against all types of attack along with no need to make change in the source code.

M. Kian et al. [23]proposed a anomaly based technique in which HTTP request character distribution used to detect novel SQLIA attacks

Kieyzun et al. [24] propose "ARDILLA" named technique for PHP. This technique use the input mutation and dynamic taint programming in order to find the any problem in input that can cause SQLIA. ANDRILLA can effectively detect the aging based attacks.

N. Lambert and K. S. Lin [25] proposed a technique named based on query tokenization with the help of query parse method. In this approach array of the tokens of the query are compared for same size in order to detect the injection attack.

Prithvi Bisht and Pal Singh [26] present a technique that based on automatic change in the code and use PREPARE statement. Their proposed approach works well with legacy applications against the SQLIA.

J. C. Lin et al. [27]Propose SQL proxy based blocker technique to prevent SQLIA known as (SQLProb). This technique dynamically detects and extracts the inputs for any undesirable input sequence. This technique proved to prevent all types of SQL injection attacks.

M. Monga et al. [28] present PHan or PHP Hybrid Analyzer that use off line and online analyzer with the help of static and dynamic analysis.

Moosa, A.[29] Propose a technique that is accurate and efficient in term of time usage. They present an approach known as keyword approach that has higher accuracy and efficiency for testing the bad characters input into the input by a malicious user.

A.Razzaq [30] proposed a technique that can safeguard against the web attack at different layers. They claim that their proposed technique has low false positive rate. The proposed technique is capable of detection of different types of known and unknown vulnerabilities including the SQLIA. The technique is also efficient as it use the control flow graph.

H. Shahriar and M. Zulkernine [31] present a testing technique for SQLIA that is mutation based. With the help of this technique test data is generated that can be used for revealing the SQLIA. The author proposed a mutation based SQL Injection vulnerability checking tool named as "MUSIC". The proposed "MUSIC" technique automatically creates mutants for applications written in JSP that is used in subsequent analysis.

S. V. Shanmughaneethi et al. [32] proposed a methodology for detection SQLIA vulnerabilities. In order to validate the input the static structure of query is used. This technique use parsing of the query in order to perform the static analysis.

S. T. Sun, K. Beznosov[33] review different types of SQLIA and also classify them on different basis. This study helps to have deep insight into the problem domain.

A. Tajpour and M. Shooshtari [6][3] present a review of different types of approaches against the SQLIA and also evaluate the performance of different technique.

S. Thomas et al. [34] study present an algorithm for preventing the SQLIA vulnerabilities. Their proposed technique suggests an algorithm for prepared statement which is a static technique. The proposed techniques help to prevent to logical structure of the prepared statements.

Z. Zhang et al. [35] present technique named as "code Auditor" based on tool "BLAST". This technique use static analysis and model checking. This technique usefully checks the buffer overflow vulnerabilities in the web applications.

Prithvi Bisht, and Pal Singh [26][36] suggest a technique that dynamically investigate the query for any threat. CANDID is the name of technique that they proposed in which queries are evaluated dynamically for being an input candidate. They also present a tool named "CANDID" that defends java application for SQLIA. This solution is implemented in JVM that do not require modification in the developer code.

Gregory T. and Buehrer et al. [37] in "Using Parse Tree Validation to Prevent SQL Injection Attacks"(2005) present SQL parse tree validation technique.

Zhendong Su and Gary Wassermannin"The Essence of Command Injection Attacks in Web Applications" (2006) present a technique to test and sanitize query. His proposed technique is known as SQLCHECK. SQLCHECK use grammar to validate query. [38]

Stephen Thomas and Laurie Williams in article named "Using Automated Fix Generation to Secure SQL Statements" (2007 IEEE) comprehensively discuss vulnerability replacement method against the SQLIA.

Ettore Merlo et al. in article named "Automated Protection of PHP Applications against SQL-injection Attacks" (2007) present a method to automatically safeguards applications against the SQLIA. [34] Their propose technique use static and dynamic analysis along with automatic code reengineering.

Ke Wei et al in "Preventing SQL Injection Attacks in Stored Procedures" present a new technique to prevent and detect SQLIA in stored procedures. Their proposed method use static code analysis and runtime checks to safeguard against the SQLIA. [39]

R. McClure et al. the SQL DOM technique [40] and W.R. Cook Safe Query Object [41]use encapsulation of queries in order to have a safe access to the database. This technique use query building process to prevent SQLIA. In fact these methods are code defense mechanism.

Atefeh Tajpour et al. [42] analyze different SQLIA detection and prevention techniques along with their evaluations. They also present an effective evaluation technique.

Wassermann and present a technique with the help of static analysis and automated reasoning in order to assure that input does not contain a tautology.

Huang et al. [43] Specify white box scheme for errors relating to input validation. This approach is limited in its application as it assumes that important functions can be stated ahead of time that is not possible at all time.

Livshits and Lam [44]in their detect PQL language vulnerabilities using static analysis. Their proposed technique use PQL to generate vulnerability signature and from this vulnerability static analyzer is created. This technique can detect vulnerabilities in code. This technique can be used to improve the code by avoiding vulnerabilities. The proposed technique is limited in sense that it can only detect known vulnerabilities but not new one introduced.

A. Anchlia and S. Jain [45] in their article named "A novel injection aware approach for the testing of database applications" present a different approach to detect test the SQLIA in an application.

Anup Shakya and Dhiraj Aryal [5] presents excellent taxonomy of SQLIA technique to have a deep insight into the problem domain.

Halde and Jagdish present [46] another approach that is very effective for detection and blocking of different types of SQLIA.

Ficco M el al in [47] present a weight age based correlation approach to detect and block SQLIA.

O. Dieste [48] present different strategies for detection of different types of SQLIA that take advantages of different vulnerabilities in a web application.

Sarangi [49] presents a technique for blocking SQLIA in stored procedures that is very effective as well as efficient one.

S. Ali et al in [50] present authentication mechanism that works well against different SQLIA techniques.

2.3 Limitation

All these proposed solution are limited in context of applicability and coverage of wide variety of SQLIA techniques. Some techniques suffer with performance overhead. A complete analysis of different techniques with their strengths and weaknesses described in next chapter.

2.4 Summary

In this chapter we review the material about different proposed solutions and about different work on SQLIA techniques. Aim of this section is to formulate the basics of our work and to streamline our direction towards the comprehensive and effective solution of the problem.

CHAPTER 3: REQUIREMENT ANALYSIS

3.1 Problem Domain

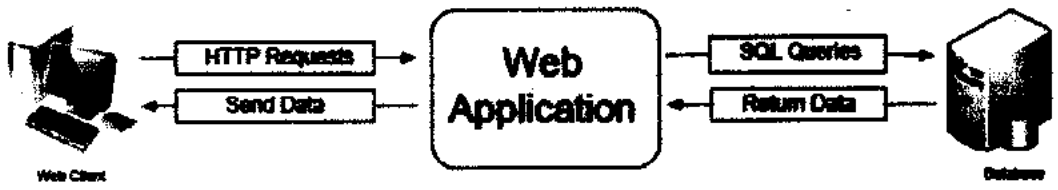
SQL Injection term was first coined in 1998 and its first usage was recorded in 2000 [6]. Since that time SQLIA has a prominent place among the most of the common and severe web security vulnerabilities [3]. It took place when syntax or semantics of legitimate query changed by a malicious user in order to have unauthorized access to the HTML document or database. This technique exploits the syntax and semantics of legitimate query with the help of SQL operators and keywords.

In a simple way we can define SQLIA as the change in intended logic, syntax or semantics of SQL query with the help of SQL operators of keyword. Or in another words SQL injection attack is a hacking technique in which input parameters of a web application used to add malicious code to access the resources that are not intended for them.

3.1.1 SQL Injection Attack (SQLIA) Process

As discussed in the introduction section now a das most of the websites are data driven. Database in a web site is like a black box. Database driven websites use 3-tier architecture where it becomes a black box where requests are received in the form of HTTP and generates SQL statements in response to that requests. The requests may also contains the parameters that are exploited by the attackers to generate SQL query to have access to information they want but are not intended for it. Database in a data driven application with 3-tier architecture receives the query as input from application logic and database response by sending appropriate data. This data is sent back to the user page through the middle tier as shown in the figure.

Figure 14 : How web applications works



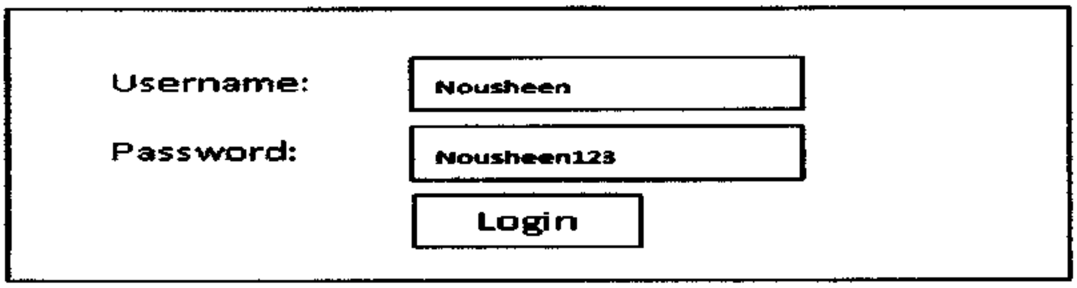
The most vulnerable page for the SQL injection attack is perhaps the log in page. In order to show the how the attack works we take a simple example of log in page and php code snippet. Following is the simple php code snippet that uses a dynamic generated query.

Figure 15 : simple php code snippet that generate dynamic query for a log in input

```
// connect to a database
mysql_connect(servername,username,password);
// store user input in the variables collected from the user input login form
$username= $_POST ['username'];
$password= $_POST ['password'];
// dynamically build the query from the user input
$query = "SELECT * FROM tbl_users WHERE username = '$username' AND password = '$password';
// execute a query
$result = mysql_query($query);
if($result)
     return true;
else
     return false;
```

In the next figure benign user input is shown and very next to it the dynamic query generated in response to that input is shown.

Figure 16 : Benign user input on a typical log in form



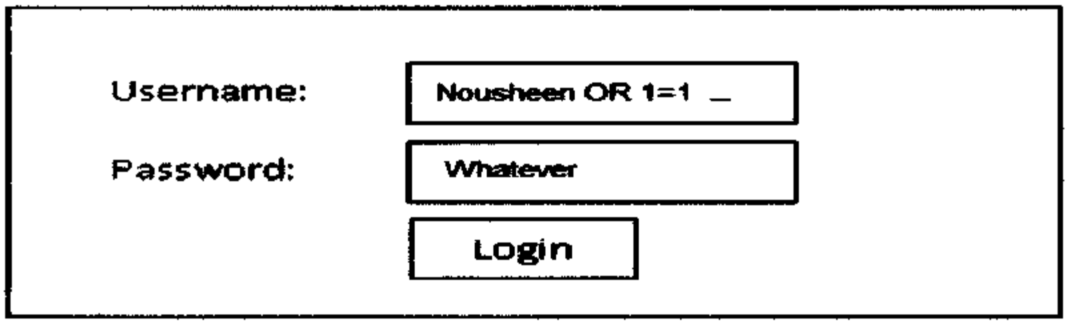
Username:	Nousheen
Password:	Nousheen123
	Login

The query generated by code in response to this input is as;

```
SELECT * FROM tbl users WHERE username = 'Nousheen' AND password = 'Nousheen123';
```

Now in the next scenario we show the malicious user input on the same flo in form and with same code that generate dynamic query in response to the previous code snippet.

Figure 17 : Malicious user input on the log in form



Username:	Nousheen OR 1=1 _
Password:	Whatever
	Login

The dynamic query generated in response to this is shown here;

```
SELECT * FROM tbl users WHERE username = 'Nousheen' OR '1'='1'- - 'AND password='whatever
```

Malicious user very tactfully try to ignore the password by using the comments operator – of sql result into everything would be ignored after the --. In order to validate the user name OR operator used because it result into 1 whether the user exist or not since result of 1=1 would always 1 whether user 'Nousheen' exist or not. This is the simple technique and we can easily deduce that there could be many techniques for this sort of authentication.

The problem may extend when some sort of vulnerabilities even when provide checks on input. For example some web sites do not allow use of quotation but SQLIA still possible with the use of numeric expressions. The core of the problem is that input is taken as isolated lexical entities.

3.1.2 Consequences of SQL injection attacks

SQLIA is one of the most prominent threats to web applications and gained much attention over the few years. SQLIA allow the attackers to have access to the database is the main reasons for using this SQLIA technique. Database is the hub of information resource for any organization and can affect much more than any other resource. Database usually contains sensitive information that can be exploited by the attacker for many reasons. Every SQLIA has intentions after being successful. This intention is the goal of the attack. The most intended intentions of the SQLIA are retrieving data, uploading files, retrieving database schemas etc.

The typical intentions of the SQLIA are;

1. In order to perform database finger printing. Fingerprint information is the information about type and version of database. Every successful SQLIA attack must know about certain information in the form of version of SQL language used. Different data base management system use different version of SQL language. Few example Oracle use PL/SQL and MS SQL Server use T-SQL.

2. Run a command to get all the information about user and password
3. In order to determine the database schema. A database schema is the structure of the database with name of fields, tables and logical relationships. In order to load a successful subsequent attack these information are very crucial.
4. In order to extract and modify data. This is the core aim of the SQLIA and most abundant and common types of attacks.
5. In order to shut down database, locking or dropping certain tables in the database. Result into DoS.
6. Certain attacks also aim to evading attacks in order to avoid system detection technique.
7. Files upload in order to replace existing files or files with wrong information
8. In order to bypass authentication and to gain status of a privileged user. Authentication is the mechanism to allow access to the information to only legitimate users. Matching user name and password is the most common authentication mechanism in web applications. With the help of SQLIA the attacker want to have access to the database in the form of un authorized user.
9. In order to execute certain types of remote commands. Remote commands are used to run arbitrary program on the server that can assist the attacker for subsequent attacks or to archive other purpose.
10. Using through look up and retrieve IP addresses to attack them.
11. Interacting with underlying operating system
12. Online shop lifting for example changing the price of items
13. Changing the account balance in the debit, credit or online account

TH-13619

Most of the attacks (about 90%) are related to financial loss but in the long run most damage is on the repute of the organizations. According to a survey about 60% of customer stop using services after a security breach into the organization.

3.1.3 Most common vulnerabilities for SQLIA

The most common type of vulnerabilities for SQLIA attacks are "insufficient input validations", existence of "privileged accounts" and "extra functionalities". Input validation is the technique to filter user input to prevent malicious attempt from users. Improper or weak authentication mechanism of user and password matching on the log in page is the most common initiation of SQLIA. When user input is allowed without proper verification the attacker is free to execute code for his/her malicious intents. Attacker usually exploits the weak or improper input authentication to induct malicious code to get desired results for malicious intents.

Database also allows certain accounts with empowered privileges to make changes in the database extensions or evens into the intention of the database. These privileged accounts are also fertile source for the SQLIA once the attacker gets this information. With the access of the privilege account like administrator account empower the attacker to perform any transactions.

Extra functionality in the modern DBMS also helps or empowers the attacker to exploits the sources for his malicious intents. . For example xp_cmdshell is used to execute certain operating system commands but it may empower the attacker to gain access to the operating system.

3.2 Problem Scenario

3.2.1 Tautology Attacks

Tautology attacks aim to insert malicious code with the conditional statements to order to make their evaluation always to true. The consequences of this attack vary to certain extent and totally depend upon the intention of the attacker. The most common intent is to bypass the authentication process and to extract data for the intended data. The inject-able field is exploited that is used in WHERE clause of the query. The transformation of all the conditions into tautology, results into return of all the rows of the intended table.

In order to be successful for the tautology based attack. Attacker must have some know how about inject able or vulnerable parameters and up to some extent the coding constructs that evaluate the query and return results. Tautology based attacks considered successful when all the results are displayed by the code or at least one record that allow some other actions. The example explained in the SQLIA process the query generated response to malicious user input is;

```
SELECT * FROM tbl users WHERE username = 'Nousheen' OR '1'='1'- - 'AND password='whatever
```

Malicious user submits the [1=1 --] with the user name. – causes the impact of remaining query ignored. This code change WHERE clause into a tautology results into true for each row and return the entire row of table.

3.2.2 Union Attack

In this technique safe query is joined with injected query with the keyword UNION in order to get data from other tables. In this sort of technique a vulnerable parameters is used to return the result from other tables.

In this sort of attack the attacker try to get result from the tables different from the table specified by the developer. Using syntax UNION SESLSECT <injected query> with a legitimate query helps to retrieve results from other tables in contrast to table specified by the developer. With this sort of query the result returned from the first query and result from the second part of the query is returned to the attacker. As the illegitimate part of the query is in complete control of the attacker he or she can access the data from any table. A simple use of UNION is shown in the next query. For example if the user insert the fallowing query into the login field;

['UNION SELECT pwd FROM user-info WHERE id='abc' AND pwd='']

It would join with the original query and the resultant query generated in response to this query is shown in the small table.

```
SELECT * FROM users WHERE id=' ' UNION SELECT pwd FROM user_info WHERE id='abc' -- AND pwd=' '
```

In this query there would no result returned from the first query but the second query return the password from the user info table for the abc user.

3.2.3 Logically Incorrect query Attack

This type of attack used to get information about the structure of database. This sort of attack help to get basic type of information that can be used for subsequent attacks. Information returned by the server in response to an error is used for making other attacks. The error page returned by the server contains information that can be exploited by inject able parameters. In this technique attacker tries to generate syntax, logical or type conversion information. Syntax errors are used to gain information about injects able parameters. Type conversions errors help to gain information about data type and logical errors help to gain information about the tables and fields.

3.2.4 Piggybacked Query

In this technique illegal queries joined with legitimate query with the help of certain delimiters like ";", "," etc. Additional queries injected into original query. In this case injected queries piggy backed on the original query. In case of successful attack database execute multiple illegitimate queries after the legitimate one. In this sort of technique database receive multiple queries. Legitimate query execute in a normal fashion whereas inserted query can prove to be very harmful as any SQL command can be executed including the table drop and deleted as well insert of new data. For example if the user insert fallowing input into the password field "[‘; drop table users - -] the query generated in response to this is shown here.

```
SELECT * FROM USERS WHERE id='Nousheen' AND pwd="; DROP TABLE users --'
```

After executing the first query the second query would be executed in response to the input. In response to second query the user table may drop result into loss of information. This could hit the organization in a severe manner.

3.2.5 Alternate Encoding

Illegitimate queries injected by changing the coding schema in the form of Unicode, ASCII or hexadecimal format. Using encoding for the queries the injected code can bypass the filter or scanner used to test the legitimacy of the query. For example attacker can use char (44) which is the code for quote that may be the bad character for the scanner.

3.2.6 Inference

In this types of technique the attacker try to change the behavior of the database. Build injection and timing attack are based on inference principle. In blind injection the attacker uses series of true and false questions from the database. When developers hide the errors details the SQLIA become difficult but not impossible. Blind injection techniques are used in this sort of scenarios. In timing attack response delays from the database observed in order to inject queries.

3.2.7 Stored Procedures

Stored procedures are used to add another layer of abstraction. Depending upon the logic of executing the stored procedure SQLIA can be used in different ways.

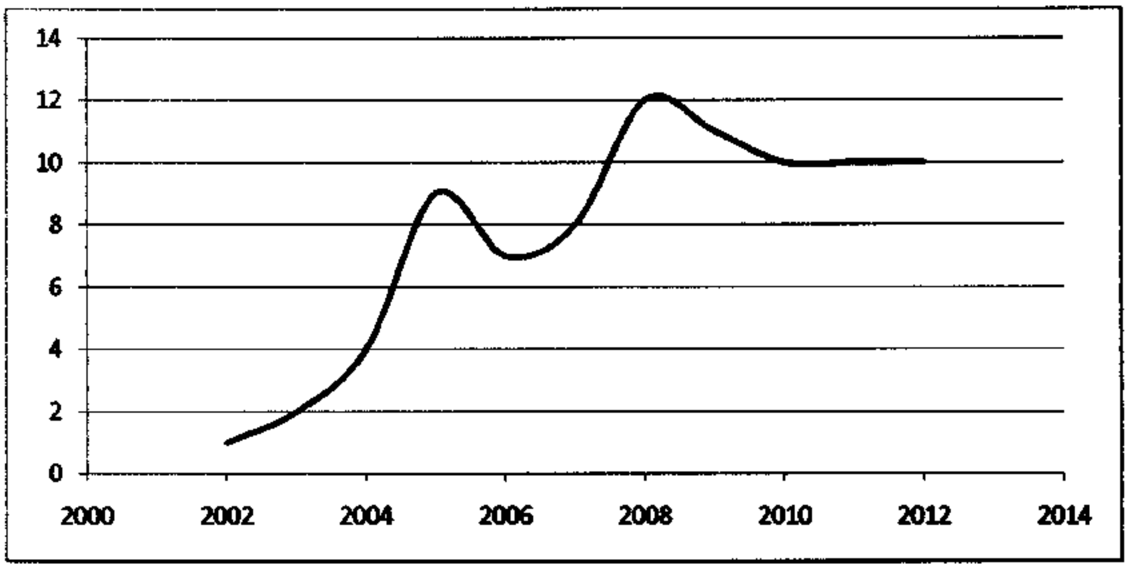
3.3 Focus of Research

This chapter describes the basics of SQLIA, the basic techniques and defense techniques along with the proposed and existing solutions to the issue.

3.3.1 Status of SQLIA defense techniques

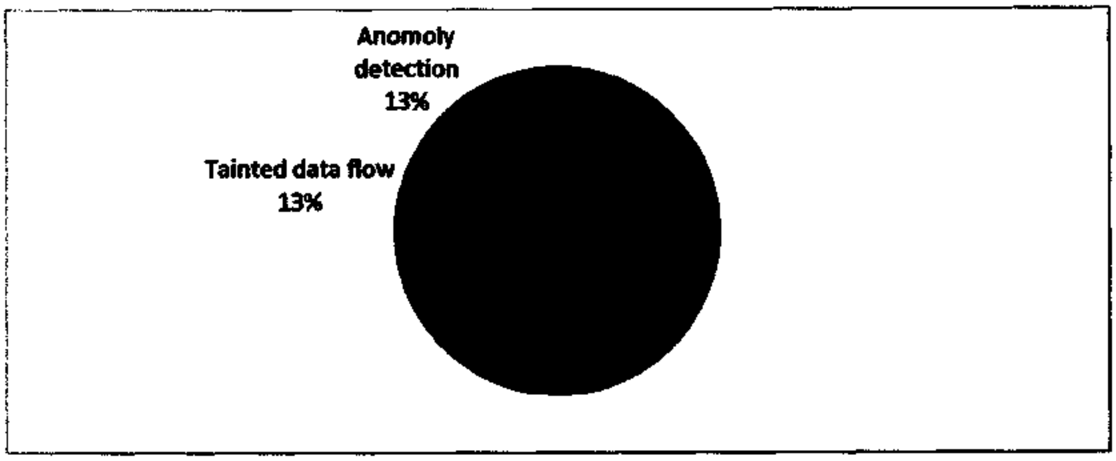
As we mentioned above the SQL injection has gained the attentions of researcher and different defense techniques emerged over the last few years and this rate emerged to a large extent. The next figure depicts the story of increasing defense techniques emerged over the last few years and a rising trend is very obvious in the figure.

Figure 18 : No of defense techniques emerged over the time [5]



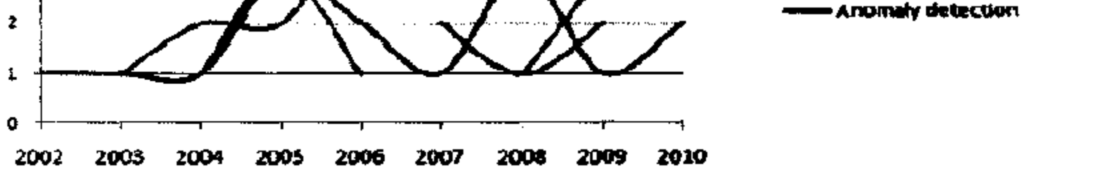
These techniques use different detection principles. Out of the emerged techniques over that last few years thirteen percent use tainted data flow, thirteen percent anomaly based, twenty five use signature based and forty nine percent use grammar based technique. It is depicted in next figure.

Figure 19 : %age of detection principle used [5]



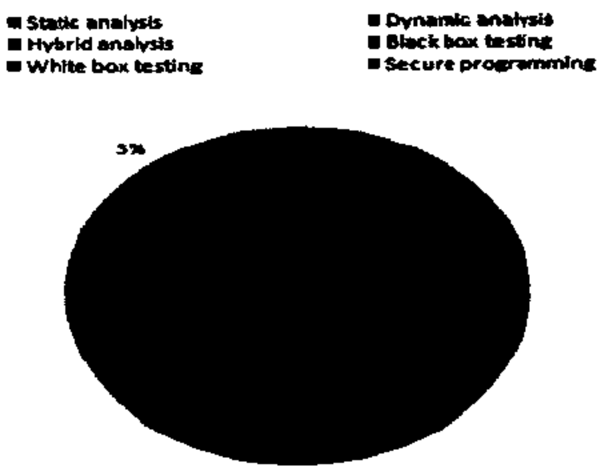
In the next figure we show the emergence of detection principles over the year. In figure it is clearly evident that grammar based approach are used in a dominant way and next are signature based with tainted data flow and anomaly based are used at equal rates.

Figure 20 : detection principle usage over the time [51]



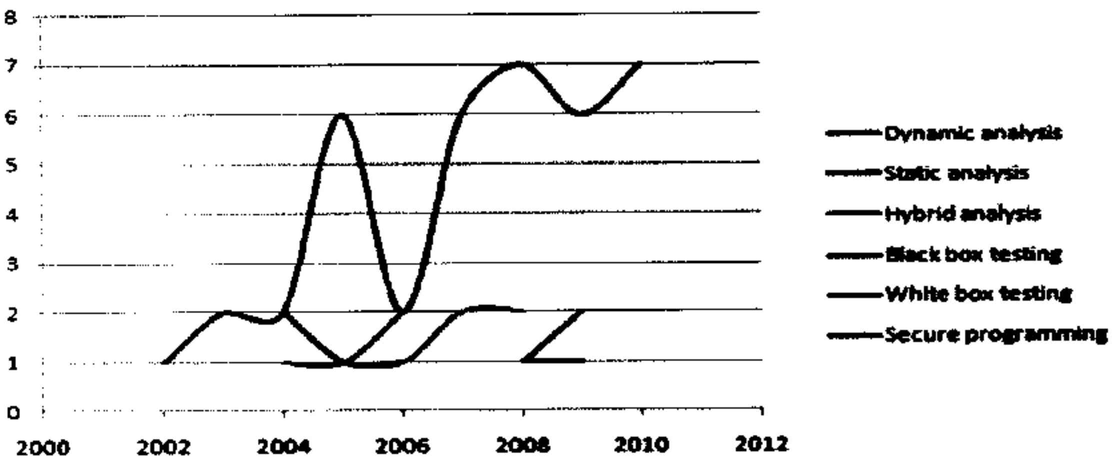
Different detection techniques also use analysis methods. The next figure shows the use of different analysis methods over the last few years. The mostly used analysis method is the dynamic analysis and least used analysis method is white box testing. Hybrid method is at second number in the analysis method used in different techniques developed over the last few years

Figure 21 : percentage of analysis methods used



It is very interesting to observe that use of analysis method used over time is very different as shown in the next figure.

Figure 22 : Analysis methods used over the time



3.3.2 SQLIA detection and prevention tools and techniques analysis

3.3.2.1 Abstracting Application-Level Web Security

In this technique authors proposed structural techniques for the web developers to implement security policies for web applications. Authors believe that these techniques help to protect several types of web attacks. Their proposed techniques based on three components including the Security policy description language (SPDL), policy compilers, and security gateways. Application validations rules are defined in SPDL where transformation rule decide what have to do when a malicious attack performed on the attack. Policy compilers transform the rule into code. Whereas security gateways between the client and server acts as firewall to implement the specific security policy. [52]

3.3.2.2 Web Application Security Assessment by Fault Injection and Behavior Monitoring

This technique works well for SQLIA as well as for XSS attacks. It detects web vulnerabilities with the help of web crawlers. It also uses different types of attacks technique as use machine learning approach to defend against these attacks. It uses machine learning approach therefore considered as best than other techniques. It is implemented into WAVES (Wave Application Vulnerability and Error Scanner) tool by the author.[53]

3.3.2.3 An Automated Universal Server Level Solution for SQL Injection Security Flaw

This technique is used to safeguard the web server against the SQL injection techniques. Author claimed that this technique can be used to protect any type of web server from SQLIA. AUSELSQI intercept the HTTP messages for any malicious or suspicious contents and rejects the malicious contents. It is implemented this technique on an ISAPI filter on IIS web server where it can also act as firewall.[1]

3.2.3.4 SQLRand: Preventing SQL Injection Attacks

In this technique the author applies the randomization of instructions set of SQL language. SQL statements are tried to change with the help of a random number that make it very difficult for the attacker to predict the syntax of SQL. This technique known as SQLRand implemented using a proxy that sit between the server and client that translate back the randomized SQL statements to actual code. When a malicious user make an attempt the proxy would reject the queries as it would not be recognized by him.[54]

3.2.3.5 JDBC Checker: A Static Analysis Tool for SQL/JDBC Applications

In this technique named as JDBC checker is used that checks the completeness of the dynamic built query and identifies any problem with the dynamic generated query for java web application. This technique is use context free language for semantic checking of the SQL queries. [55]

3.2.3.6 Securing Web Application Code by Static analysis and Runtime Protection

This technique uses the information flow analysis in order to detect any input validation types vulnerabilities. It also uses static flow for taint flow. Main drawback of this technique is that it assumes that ample preconditions for suspicious inputs can be articulated.

This technique implemented in WebSSARI (Web application Security by Static analysis and Runtime Inspection) tool. [43]

3.2.3.7 An Analysis Framework for Security in Web Applications

This technique use static analysis that determine the SQL queries as finite automata and then checks the finite automata for security problems. [38]

3.2.3.8 Using Parse Tree Validation to Prevent SQL Injection Attacks

This technique uses the parsing of SQL query and compares it with the expected query for possible malicious inputs. This query checking is performed in a dynamic fashion with the help of a model query that is used for comparison at run time. Queries are compared both before and after input. Author has implemented this technique in Java. [37]

3.2.3.9 SQL Injection Protection by Variable Normalization of SQL Statement

In this technique normalization of variables used to safeguard the web application against the SQLIA. Creators have proposed the variable standardization strategy to ensure web provision against SQL infusion assaults. Initially, the system use virtual database connectivity to concentrate essential structure of SQL. Furthermore, this data is used to check the legitimacy of SQL statements. In the event that the SQL is diverse, it will be blocked.

The idea of this system is to utilize variable standardization to change the variables and protect the structure of the SQL statements. Despite the fact that client supplied variables (which essentially contrast assuredly) are utilized to develop SQL explanation yet the fundamental structure of SQL remains same and if the supplied variables are noxious, it will change the structure of SQL statements and we will ready to discover it. [56]

3.2.3.10 Dynamic Taint Propagation for Java

In this technique the user inputs are tagged and tracked in a dynamic manner. Unreliable user input is tainted. This technique has advantages that it does not require modification in the source code. This is implemented in JVM in order to taint the suspicious user inputs and tracks these inputs for life time. This technique use source, propagation and sink and use the heuristic approach to taint the unreliable user input. [57]

3.2.3.11 AMNESIA: Analysis and Monitoring for Neutralizing SQL Injection Attacks

Author proposed AMNESIA technique that is based on dynamic and static analysis that safeguards various types of web servers. In this technique the vulnerable portion of page are identified known as hotspot. This technique based on four steps. In the very first step the application is analyzed for hot spots. In the next part all the dynamic generated compared with the legitimate queries that are created with the static part. This technique also does not require modification in the code. This techniques shown good results for efficient and effectiveness.[9]

3.2.3.12 Finding Security Vulnerabilities in Java Applications with Static Analysis

In this technique static analysis tool is used for various types of web vulnerabilities detection. This technique use PQL (Program Query Language) that resembles with java. Main use of this technique is to taint the susceptible user inputs. [44]

3.2.3.13 Finding Application Errors and Security Flaws Using PQL: a Program Query Language

This technique use PQL (Program Query Language) that is a run time check for the web application security. This technique is effective against the SQLIA, XSS and path vulnerabilities. This technique taints and sanitizes the strings rather that the characters in the user input. This technique has certain drawback as it is poor in defense against the numeric field injections and identification of all tainted input and its propagation is a difficult. [58]

3.2.3.14 SQL DOM: Compile Time Checking of Dynamic SQL Statements

SQLDOM technique enforces the use of strongly type's database. This technique use API to enforce or implement the technique. It is implemented in C and dot Net framework. In this technique abstract object model and Sqldomgen is used. Abstract object model is used to generate legal query at run time whereas Sqldomgen is used for API. This technique is known as SQL DOM (SQL Domain Object Model).[40]

3.2.3.15 A Learning-Based Approach to the Detection of SQL Attacks

This is anomaly based technique that considered anomalous behaviors of the database as attack to the system. This technique uses different model to taint the query that can create the anomaly in the system. [59]

3.2.3.16 Pixy: A Static Analysis Tool for Detecting Web Application Vulnerabilities

It is a static analysis technique that can detect SQLIA and XSS automatically. It use data flow analysis for taint analysis. In order to enhance the correctness and precision literal and alias analysis is also used. This technique is implemented in pixy tool that is open source tool that only target XSS. [60]

3.2.3.17 SecuBat: A Web Vulnerability Scanner

Author developed a scanner named as "SecuBat" that use white box testing for identification of possible vulnerabilities. This technique relies on three components named crawling, attack and analysis components. This technique is implemented in C and MS SQL server database. [61]

3.2.3.18 Automatic Revised Tool for Anti-Malicious Injection

Writer believes that input validations are the main source of vulnerabilities for SQL and XSS attacks. This technique first checks for threats input areas in the HMTL form, cookies etc. This technique then generates automatic validation technique. This technique based on four components named as spider, analyzer, function producer and tester.[62]

3.2.3.19 Eliminating SQL Injection Attacks – A Transparent Defense Mechanism

This technique use validation and run time checks to safeguard the application against different types of attacks. This technique has advantage that it can be merged with existing application and also do not require any modification in source code.

This technique relies on string analysis for static analysis and builds the SQL graph. At the run time checks the input is validated against the SQL graph built at the static analysis phase. [63]

3.2.3.20 Defending Against Injection Attacks through Context-Sensitive String Evaluation

The technique named as Context-Sensitive String Evaluation (CSSE) use metadata information and context sensitive string evaluation function. This technique also does not require source code modification and programmer attention. This technique is implemented in PHP and use context sensitive. [64]

3.2.3.21 D-WAV: A Web Application Vulnerabilities Detection Tool using Characteristics of Web Forms

This method is an automated testing methodology which detects web vulnerabilities, for example, SQLIA and XSS. It gets a target web structure with the assistance of given URL.

It makes test suites which consider the confidence of each one test with evaluation. At last, these test suites are executed and comparing in order to make conclusion for HTML code investigations. A Web Application Vulnerabilities Detection Knowledge Repository is utilized to figure out if the vulnerabilities exist or not. This technique implemented into D-WAV. [65]

3.2.3.22 X-LOG Authentication Technique to Prevent SQL Injection Attacks

In these technique three filtrations schemas are used named as vulnerability guards, X Log authentication and stored procedures used. This technique is used against many types of attackers for effectiveness and efficiency and proved to be an excellent one. [66]

3.2.3.23 The Essence of Command Injection Attacks in Web Applications

SQLCHECK is a technique to safeguard against the SQLIA. In the same article author also defined the SQLIA prevention technique. SQLCHECK technique based on context free grammar and query parse tree structure that is used to validate the user inputs. This technique is used to validate the syntax of dynamic generated query that is checked in a run time manner. This technique also uses the Meta data for be beginning and end of query.

3.2.3.24 Preventing SQL Injection Attacks in Stored Procedures01

This technique uses the data base layer to safeguard the attacks whereas most of techniques discussed use application layer. This technique is used to safeguard the stored procedures against the SQLIA. This technique relies on static analysis and run time checks. One major advantage of this technique is that it can be used automatically. [39]

3.2.3.25 Swaddler: An Approach for the Anomaly based Detection of State Violations in Web Applications

This technique also uses to protect stored procedures against the SQLIA named as "Swaddler". This technique also uses static analysis and run time checks of validations. This technique parse the SQL query and compares the user input query with the original SQL query to identify any problem with the input query. [67]

3.2.3.26 SMask: Preventing Injection Attacks in Web Applications by Approximating Automatic Data/Code Separation

SMask is a technique that used for detection of SQLIA and XSS. This technique use string masking for syntactical analysis for differentiating the legal query and malicious one. This technique use pre and post processor for query validation. [68]

3.2.3.27 Sania: Syntactic and Semantic Analysis for Automated Testing against SQL Injection

This is a novel technique that tries to identify the possible vulnerabilities for SQLIA at development and testing phase. This technique use syntactic and semantics of the queries for possible detection of vulnerability for SQLIA attack. This technique identifies the point where malicious user can exploit for SQLIA. This technique implemented in a tool named "sania". [69]

3.2.3.28 Automated Protection of PHP Applications against SQL-injection Attacks

This technique based on static and dynamic analysis for identification of SQLIA in a PHP code. This technique also relies on code reengineering to protect legacy applications. This technique tested in phpBB (PHP Bulletin board) and produced amazing results. [70]

3.2.3.29 Using Automated Fix Generation to Secures Statements

This technique automatically eliminates the SQLIA vulnerabilities from the java code. This technique uses the prepared statement and change the vulnerable part of the query to the prepared statement. Programmer can change the vulnerable part of the code with the automatic generated code. [71]

3.2.3.30 A Method for Detecting Code Security Vulnerability based on Variables Tracking with Validated-tree

This technique analyzes the code for possible vulnerabilities. It uses the parse tree for validation of variables used in program script. This technique considers the manipulation of the variables from the input. On the basis of variables manipulations the final report is prepared for possible vulnerabilities. This technique is implemented in a tool named "Software Code Vulnerabilities of SQL Injection Detector (CVSID)" [72]

3.2.3.31 Evaluation of Anomaly Based Character Distribution Models in the Detection of SQL Injection Attacks

In this technique HTTP requests are checked for any anomaly that may be source of threat. This technique also detects and handles unknown SQLIA techniques. This technique parses the HTTP request in order to identify any threat. This technique also does not require change in source code. Author also presented two characters distribution model named as FCD (Frequency Character Distribution) and SCC (Same Character Comparison).[23]

3.2.3.32 Automated Fix Generator for SQL Injection Attacks

This technique automatically detects and eliminates the SQLIA in order to change the vulnerable queries into safe one. This technique also uses the prepared statements to detect and fix sql queries that can exploit the validation related vulnerabilities.[73]

3.2.3.33 SAFELI: SQL Injection Scanner Using Symbolic Execution

SAFELI is tool that detects the possible vulnerabilities into a code. SAFELI stands for 'Static Analysis Framework for discovering SQL Injection vulnerabilities'. This tool analyzes the byte code of java application and use static analysis for any possible vulnerability. This tool also use string solver for identification of possible attack. [19]

3.2.3.34 WASP: Protecting Web Applications Using Positive Tainting and Syntax-Aware Evaluation

This technique safeguards the web application against the SQLIA using the positive tainting. It tracks the data for being trusted or not. At the next phase it use dynamic tainting for trusted only trusted data and only pass the trusted data for further processing. This technique also performs static analysis of the syntax of the query for possible vulnerabilities. It is implemented in WASP (Web Application SQL-injection Preventer). [8]

3.2.3.35 SQL-IDS: A Specification-based Approach for SQL-Injection Detection

This technique use specification for detection of SQLIA vulnerabilities. It checks the SQL statements that could breach the security. This technique uses the specification for identification of vulnerabilities in developer's specified queries. This technique also use a filter that process the queries between server and web application for possible problem with query generated in response to user input. This is independent of any DBMS and also do not require modification in source code. This technique is implemented in a tool named "SQL-IDS (SQL Injection Detection System)". This tool monitors the traffic in java applications. [22]

3.2.3.36 MUSIC: Mutation-based SQL Injection Vulnerability Checking

This is mutation based approach in which nine operators capable of SQLIA are identified. These operators are also capable of change in the source code of the application. This approach use mutants to create test cases to reveal more and more SQLIA. This technique is implemented in a tool named as "MUSIC" that detects SQLIA in JSP applications. [31]

3.2.3.37 Design Considerations for a Honey pot for SQL Injection Attacks

This technique does not defend SQLIA rather attract for SQLIA in order to have maximum knowledge about the technique employed are developed over time. This is an application layer technique where honey pot is created that attracts the attackers. The major aim of this technique is to track the attacker and safeguard the operating system that attacker does not propagate to the operating system or strong defense technique can be developed in response to this attack. [13]

3.2.3.38 A Weight-Based Symptom Correlation Approach to SQL Injection Attacks

This approach works at different layer especially network and application layer. In this technique several different anomaly detection techniques are used and data from various sources are used for different types of information. In case of any anomaly detection an error is raised that inform the system about any possible vulnerability. [17]

3.2.3.39 SQLProb: A Proxy-based Architecture towards Preventing SQL Injection Attacks

This technique named as "SQLprob" is proxy based approach in which prevents SQL attack. This technique also does not require changes in source code. Another advantage of this approach is that it is independent of any programming language and can be applied in any existing suit. [74] This technique based on fallowing four components as;

1. The Query Collector:
2. The User Input Extractor
3. The Parse Tree Generator
4. The User Input validator

3.2.3.40 Shielding against SQL Injection Attacks using ADMIRE Model

Author proposed ADMIRE model for detection and prevention of SQLIA. ADMIRE model is a step wise approach that shield the database against the attacks. The six steps of ADMIRE model are;

1. Analyze the security objectives
2. Divide the application
3. Mark the vulnerabilities
4. Identify the threats
5. Rank the threat
6. Eliminate the threat

In order to identify the vulnerabilities in the code Microsoft STRIDE model is used in this approach. In order to categorize the risk Microsoft DREAD (Damage Potential, Reproducibility, Exploit-ability, Affected Users and Discover-ability) model is used. This approach helps to developer to understand the threats and develop a strategy to combat against it. [75]

3.2.3.41 SBSQLID: Securing Web Applications with Service Based SQL Injection Detection

This technique detects SQLIA vulnerabilities based on the service based approach. This approach also considers the syntactic structure of the query and validations based vulnerabilities for user inputs. This technique comprises of input validation, query analyzer and error services. This technique also based on the syntax and semantics checks of the query. [32]

3.2.3.42 CANDID: Dynamic Candidate Evaluations for Automatic Prevention of SQL Injection Attacks

This technique use dynamically infers programmer SQL query structure for actual query. This technique makes sample inputs known as candidate inputs and compares it with actual inputs. Then benign and actual queries are compared for malicious contents. In this manner actual queries rejected if it contains malicious contents. This technique is implemented in tool known as "CANDID".[36]

3.2.3.43 TAPS: Automatically Preparing Safe SQL Queries

In this technique the queries are changed to reliable queries using prepared statements. This technique believes that with using of prepared statements chances of SQLIA drops to significant amount. This technique is implemented in a tool named as TAPS, Tool for Automatically Preparing SQL queries. TAPS tool would change the vulnerable queries into safe prepared statements that safeguards against the SQLIA.[76]

3.2.3.44 A Heuristic-based Approach for Detecting SQL-injection Vulnerabilities in Web Applications

This technique implements a tool names "viper" for testing web applications. This technique uses pattern matching for valid input from the user. It requires extensive knowledge base that can inspect the generation of sql queries. This tool based on four components named as information gathering, input parameters identification, attack generation and reporting of results. [77]

3.2.3.45 Artificial Neural Network based Web Application Firewall for SQL Injection

This is application layer technique that acts as firewall based on Artificial Neural Network (ANN). This technique use training of the character recognition that can learn automatically. This technique trains the ANN for use in web application. Two filtering technique keyword based and characters based" are used in this technique. This technique is implemented in ANNWAF.[78]

3.4 Summary

In this chapter we explore the about SQLIA and different techniques to exploit the SQLIA. Along with different techniques we also analyzed the situation and status of the problem. In the last section we provide a comprehensive overview of different proposed solution with strength and weaknesses of each solution.

CHAPTER 4: SYSTEM DESIGN AND IMPLEMENTATION

4.1 Introduction

We proposed a model that is RAT (Request Receiver, Analyzer and Tester) is a model based approach that does not require change in source code. It has certain advantages over the previous develop techniques and model that it can be implemented in any language on any sort of platform. It also has advantaged that it can be used for any sort of database.

4.2 Request Receiver, Analyzer and Tester Architecture (RAT)

Proposed approach RAT (Request Receiver, Analyzer and Tester) will perform the following steps:

4.2.1 Request Receiver

User can connect to the site from anywhere in the inter cloud. Whenever a user requests a page its request is received and analyzed in RAT. The RAT first mark the pages contained in the site. If request is for the pages on which vulnerability are marked then they are passed to Analyzer. The pages that are having no server side script or possible attack involve are pass to storage tier and request details are provided to Tester.

4.2.2 Analyzer

The task of the analyzer is to filter the A pages from A' pages. If page is a then combination of all such passes on which SQLIA is possible can be represented as

$$A = \sum_{i=1}^{n} Ai = A1 + A2 + A3 \cdots\cdots An$$

Similarly, the pages on which no server side scripting is involve and client can only see those pages can be considered as **A'**

$$A' = \sum_{i=1}^{n} A'i = A'1 + A'2 + A'3 \cdots\cdots A'n$$

If the client requests for a page that can result for attack then this request is filter by Analyzer, and Analyzer checks the possible attack by the help of its knowledge base. Here knowledge base is updated as schedule task or on the request of administrator. The updating includes adding of new rules to the knowledge base for checking attacks. It also keeps track of requests that were made by user for pages having server side scripts. This tracking is also recorded in the knowledge base if it is from a suspicious client. The Knowledge Base contains rules that can be represented as

$$R = \sum_{i=1}^{n} r'i = r1 + r2 + r3 \cdots\cdots rn$$

4.2.3 Tester

When request details are provided to Tester, it waits for the results from the storage tier, and on receiving it compares that response with expected response by using its knowledge base. If the response is unexpected then it will be a possible attack. In such case the knowledge base is updated and request of the client is stopped.

If a user's request is allowed by the analyzer, then it creates a storage checkpoint first and then passes to the tester. Meanwhile the request was passed to the storage tier. The tester will test result received from the response and the expected response from the inputs of the users. If the response from the storage tier and expected was matched then result is given to user at presentation tier else the changes made by that request is brought back to its original position in storage tier.

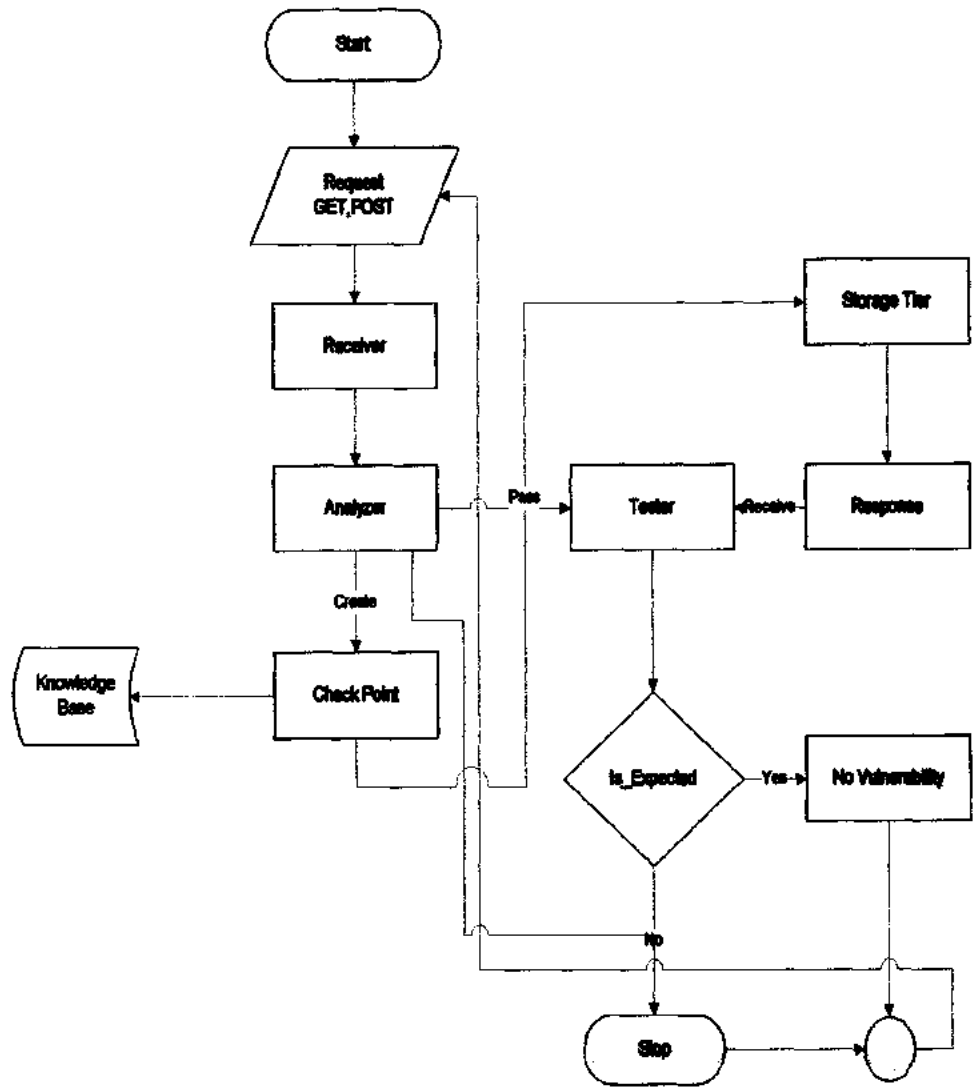
4.3 Algorithm

Figure 23 : RAT algorithm

```
function Receiver()
input(page)
inout(requestDetail)
MarkResponse = mark(page)
if MarkResponse contains vulnerability then
      Analyzer(page)
else if MarkResponse contains (no server side or possible attack) then
      store(page)
      Tester(requestDetail)
end if
end function

function Analyzer(Page)

if page is A then
      represent all combinations

end function

function Tester(requestDetail)
      reply = checkInStore(requestDetail)
      if reply is unexpected then
            updateKB(requestDetail)
            return false
      end if
      return true

end function

function updateKB(requestDetail)
      'Code to update Knowledger Base
      'KB can be created easily in prolog
end function

function store(page)
      'code to store page
end function

function checkInStore(requestDetail)
      'Code to check request detail and return respose from Knowledge
base
end function
```

4.4 Flow Chart

Figure 24: RAT Flowchart



4.5 Limitation

Proposed model implemented for ASP Dot net application using MS source code analyzer tool for possible SQLIA vulnerabilities in code. This tool has to identify the page that contains the server side script and possible vulnerabilities. Overall performance coverage of the whole system depends upon the performance of the tool. In future it is proposed that different types of analysis tools can be configured for the application.

For some application that require high response rate may feel performance issue due to extra overhead of analysis and testing. This side of model needs to be improved with the help of implementation tool to minimize the time.

We personally feel that knowledge base should be updated using sophisticated machine learning approach about SQLIA. Heuristics and machine learning approach would enhance the applicability of the proposed solution.

4.4 Summary

In this section we present the proposed solution, with algorithm and flow chart. We also present the limitation of the proposed solution.

CHAPTER 5: TESTING AND PERFORMANCE EVALUATION

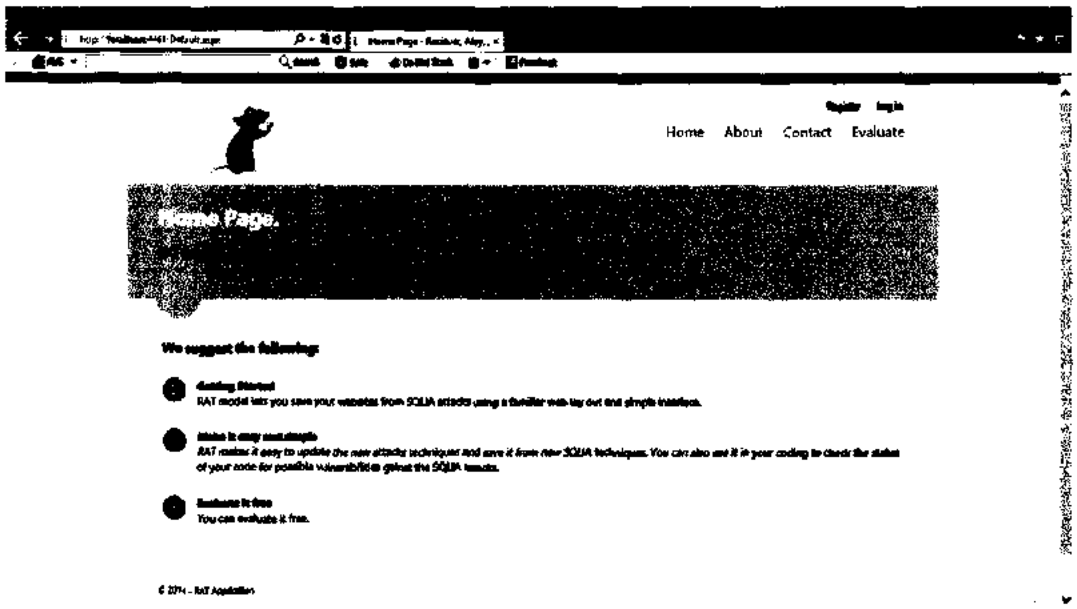
5.1 Introduction

This chapter intends to test the proposed solution against different parameters. The most important is coverage of proposed solution against different types of attacks or effective detection and prevention against different types of attacks.

5.1.1 Simulation Model

By using language likes PHP or ASP.NET we make different classes comprising of rules/functions for validating queries that cause vulnerabilities to occur. First detection of queries is done by storing them in a variable and check is made by defining a function that is applied for testing variable and depending on condition either true or false value is returned to detect the anomalous query causing attack. After that prevention is made for recovery. The collection of classes will be our knowledge base which will provide the injection information for possible attacks.

Figure 25 : Home page of RAT implementation


Register Log in

Home About Contact Evaluate

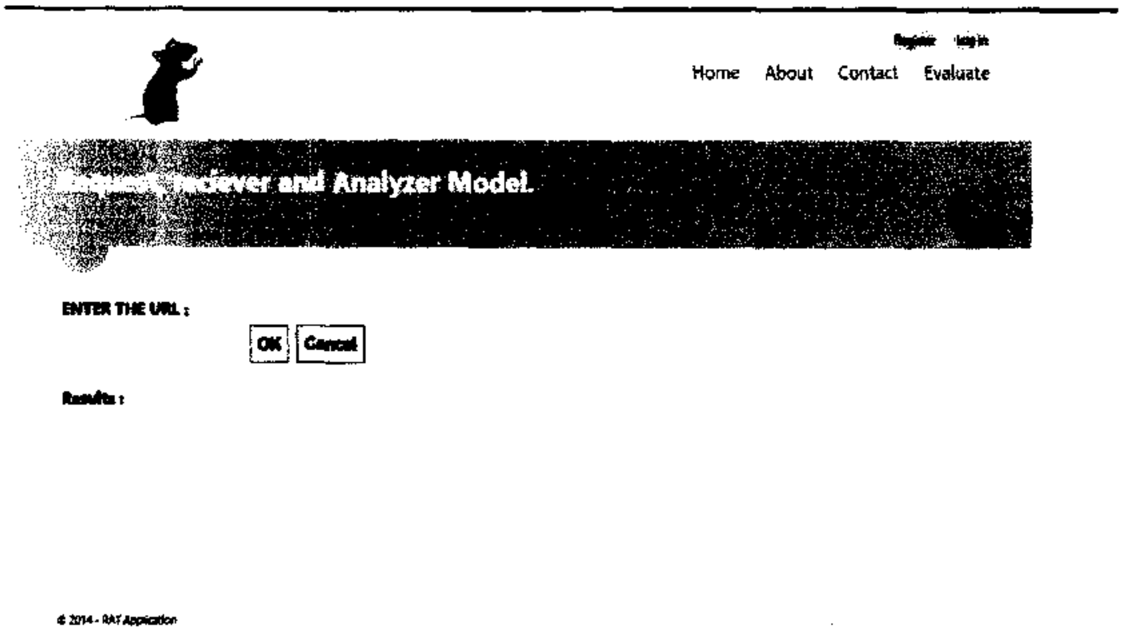
Home Page.

We suggest the following:

- **Getting Started**
  RAT model lets you save your websites from SQLIA attacks using a familiar web lay out and simple interface.
- **Make it easy and simple**
  RAT makes it easy to update the new attacks techniques and save it from new SQLIA techniques. You can also use it in your coding to check the status of your code for possible vulnerabilities gainst the SQLIA threats.
- **Evaluate it free**
  You can evaluate it free.

© 2014 - RAT Application

Figure 26 : Receivers page for evaluation of the model


Register Log in

Home About Contact Evaluate

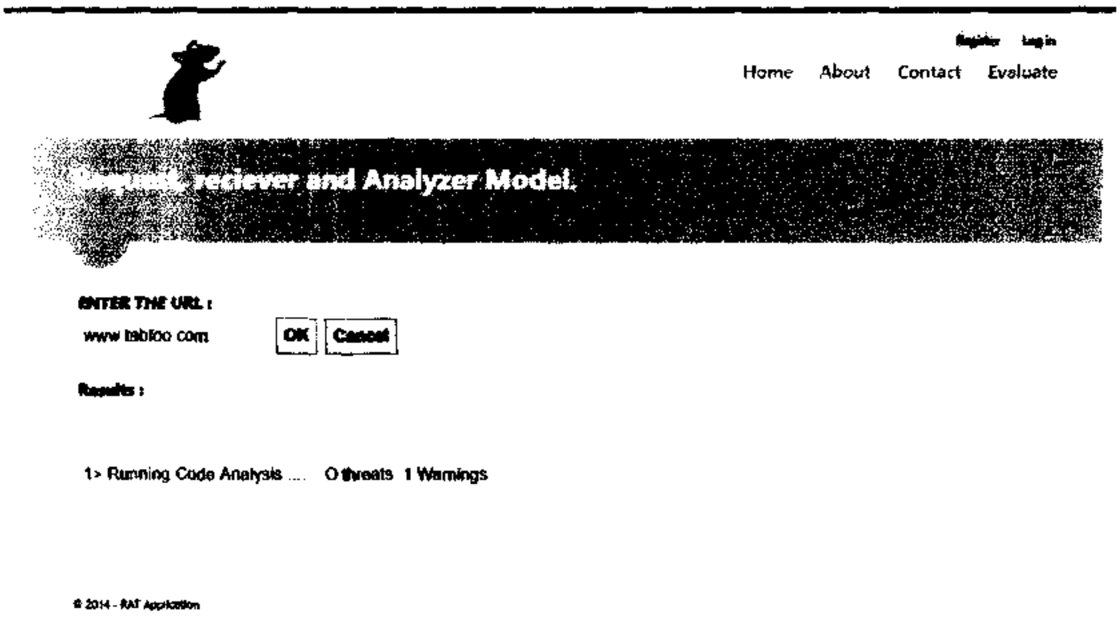
Request, reciever and Analyzer Model.

ENTER THE URL :

Results :

© 2014 - RAT Application

Figure 27 : Analysis result


Register Log in
Home About Contact Evaluate
Request, reciever and Analyzer Model.
OK
Cancel

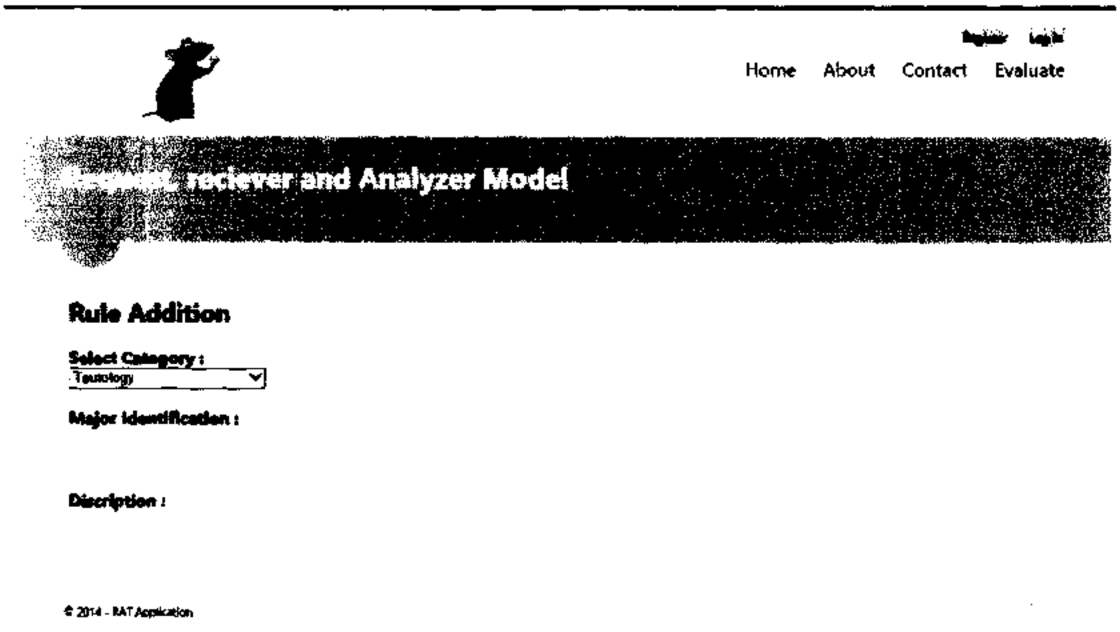
ENTER THE URL :
www.tabloo.com

Results :

1> Running Code Analysis .... 0 threats 1 Warnings

© 2014 - RAT Application

Figure 28 : Administration rule module


Register Log in
Home About Contact Evaluate
Request, reciever and Analyzer Model

Rule Addition

Select Category :
Tautology

Major Identification :

Discription :

© 2014 - RAT Application

The proposed solution has one advantage that any sort of analysis tool can be configured and new SQLIA techniques can be added by administrated.

5.1.2 Analysis

We compare the effectiveness against different tool. Effectiveness of the proposed solution is determined by:

1. No of attacks detected
2. No of attacks blocked or prevented
3. No of types of attacks detected
4. No of types of attacks prevented
5. No of database supported by proposed solution

Along with effectiveness it is also important that proposed solution put minimum overhead on application performance.

5.2 Test Scenarios

The proposed solution is also evaluated against different parameters including the time taken to detect and block the SQLIA Efficiency is measured in term of;

1. Average time taken to detect SQLIA
2. Average time taken to prevent SQLIA.

In order to evaluate the results fallowing data set are used;

Table 1 : Evaluation data set

Applications	No of inputs
Classifieds	130
Portals	115
Online shopping	123
Employee Directories	180
University databases	22

5.3 Performance and evaluation

Figure 29 : No of attacks detected by different in evaluation period

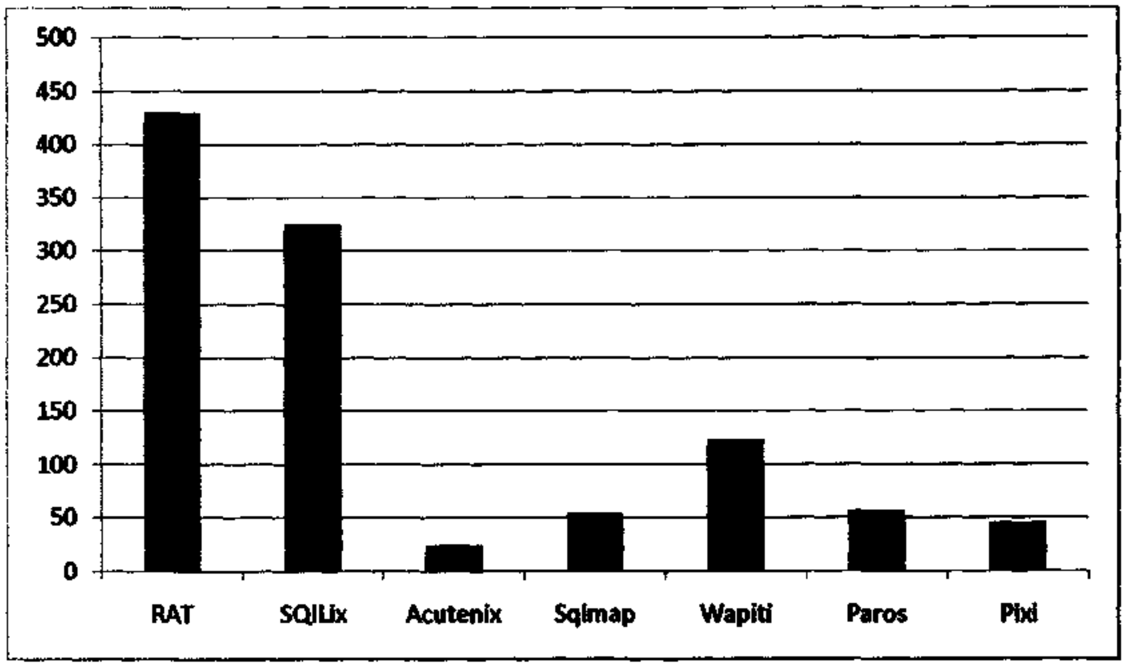


Figure 30 : No of attack protected by different tools during evaluation period

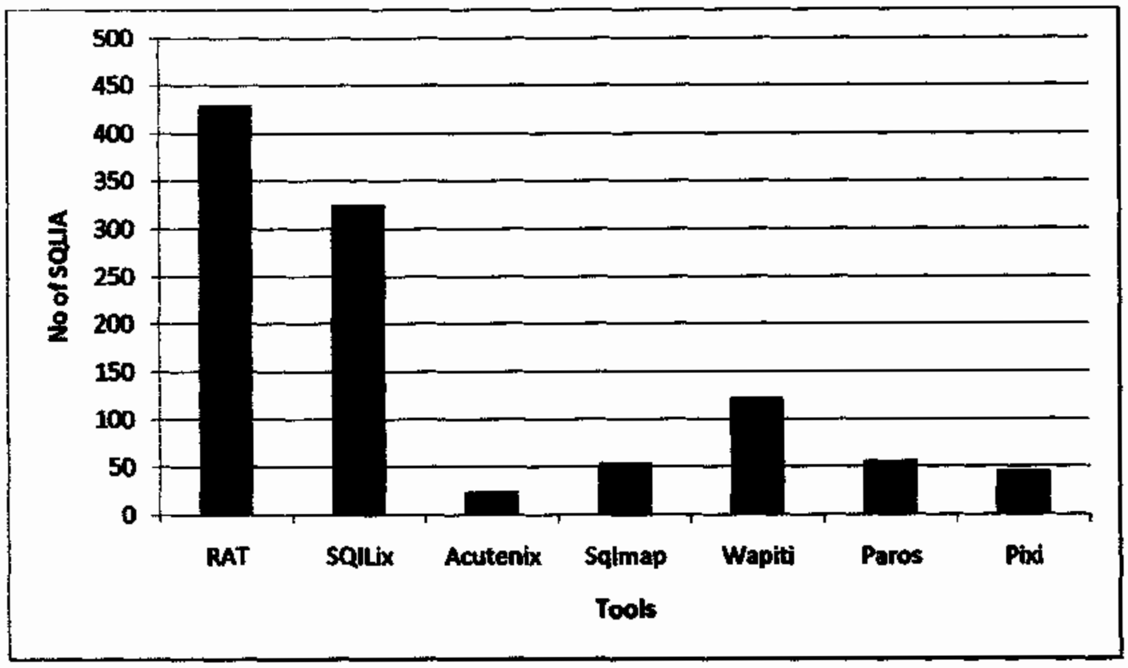


Figure 31 : Time taken for detection by different tools

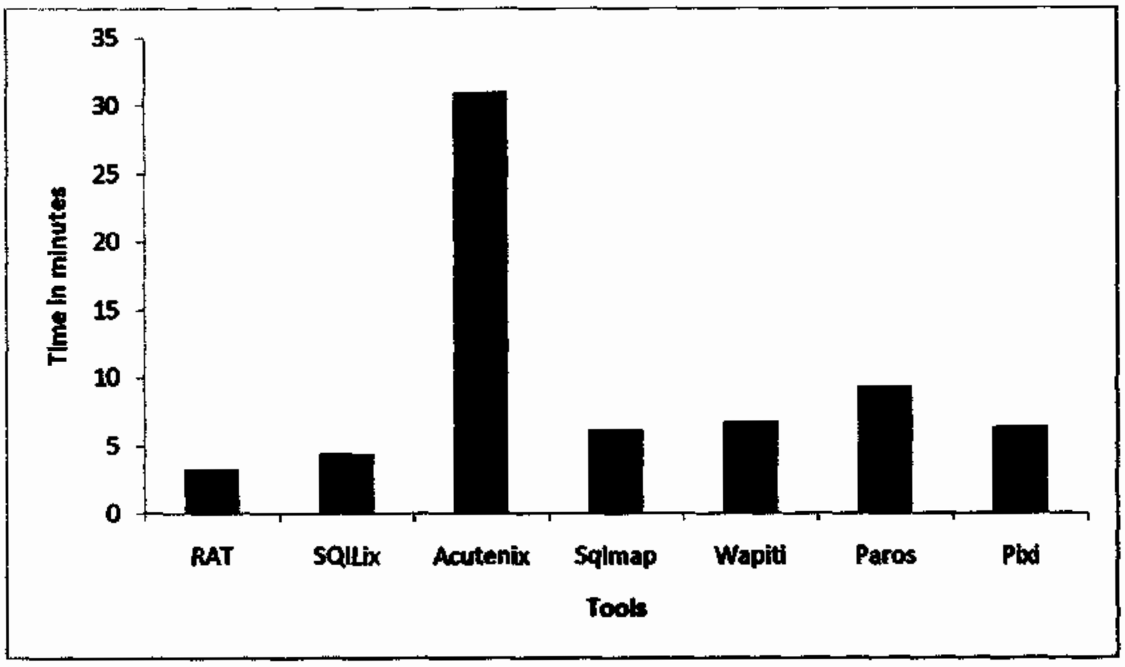


Figure 32 : Time taken by different tools to prevent SQLIA

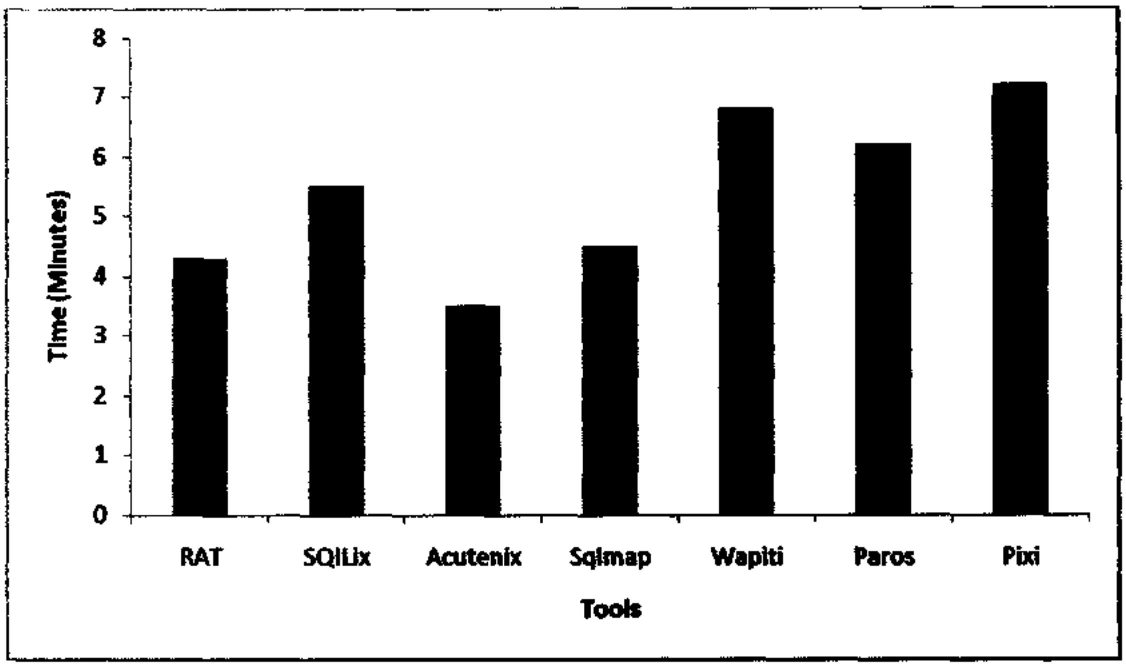


Figure 33 : Types of attacks detected by different tools

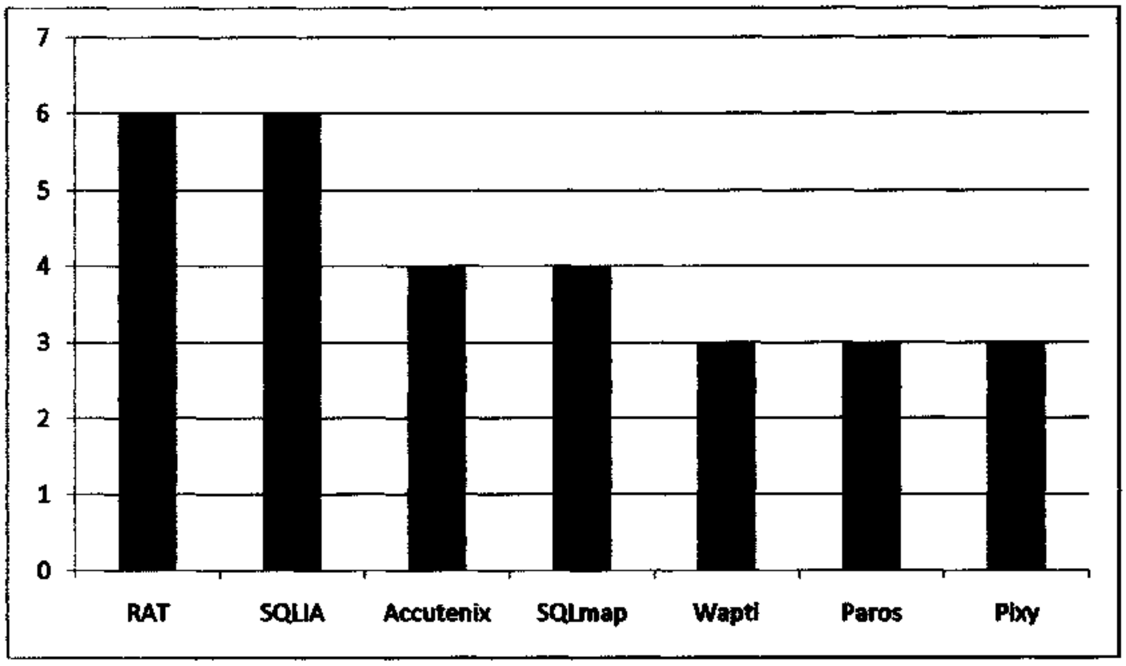


Figure 34 : Types of attacks blocked by different tools

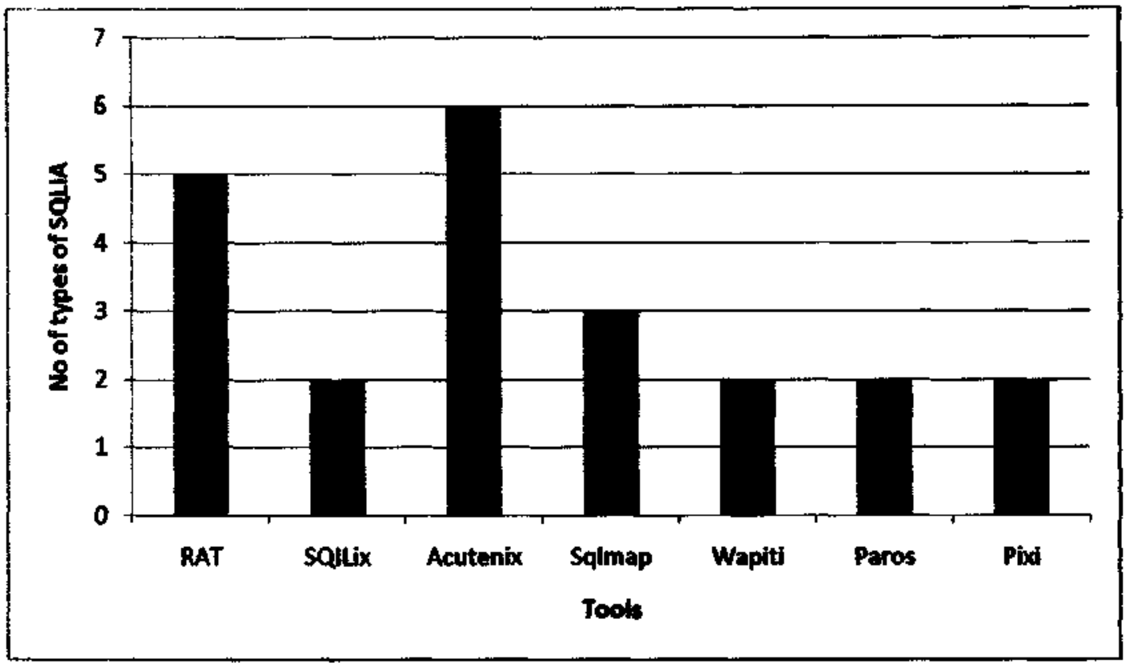
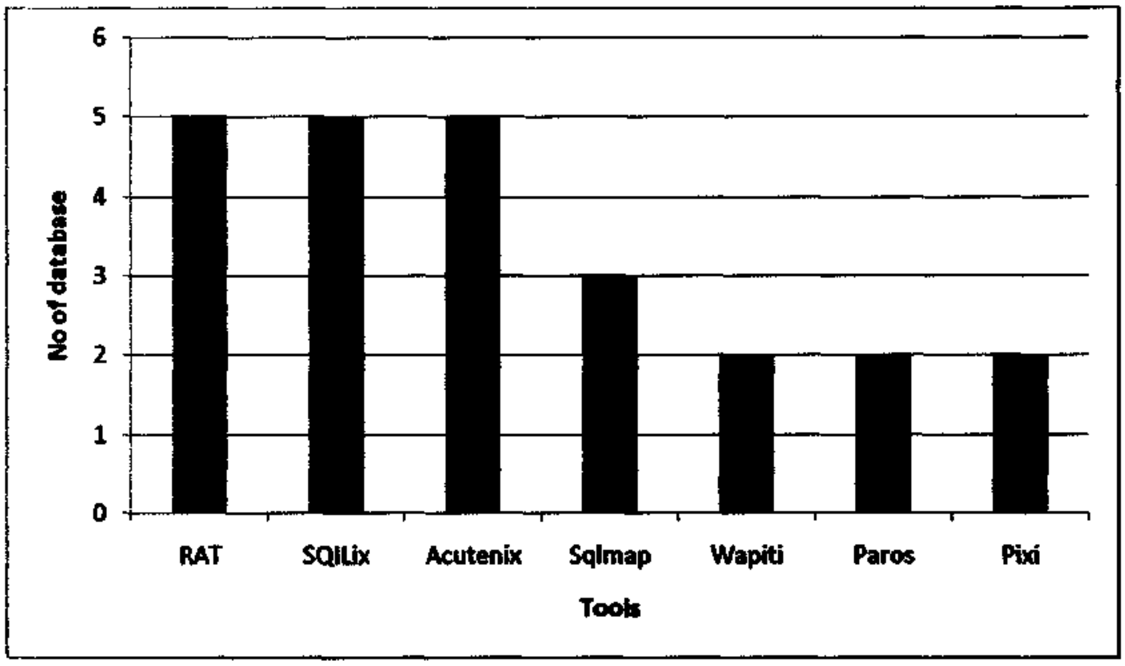


Figure 35 : No of database supported by different tools



5.4 Summary

These results shows that proposed solution is more effective against large number of attacks and more effective against variety of attacks in detection and prevention of SQLIA. The results are clear evident that this technique is more powerful to detect and block variety of attacks.

These results also evaluate the performance of different techniques in term of time overhead by different techniques imposed on the system. Our proposed system also take less time for detection and combat against different SQLIA as compared to others common available .

CHAPTER 6: CONCLUSION AND FUTURE WORK

6.1 Conclusion

With the emergence of world wide web most of the business activities shifted to web around the world. In the recent years with there is tremendous amount of transformations from desktop to web based applications. With the increase in web based applications due to potential of it there are certain threats also associated with the web application. Due to huge growth of web based applications there is also tremendous growth in web security issues.

SQL injection attack is one of the web security issues and has gained attention of the attackers due to access to database. With access of database SQLIA proves to be potential reward for the attacker, use and development of different SQLIA techniques emerged. SQLIA is one of the potential sources of web security vulnerabilities.

Due to potential of the risks associated with SQLIA, this issue has gained the attention of researcher over the last few years. Many efforts in their direction carried out result in emergence of different tools and techniques.

Though different tools and techniques proposed for the issue but many of them have problems as some require source code modification, some limited to certain database and some suffer with performance issues. We also proposed a model based on receiver, analyzer and test model. This model is superior to previous proposed solution in term of handling of wide range of SQLIA techniques including new ones. This technique also works independent of database used and requires little modification in code.

The proposed solution has wide applicability against different types of SQLIA but also put little overhead over the client application. Proposed solution shown excellent results when evaluated for average time taken for each attack detection and blocking.

6.2 Achievements

Major characteristics and achievements of proposed model are listed below.

1. This technique requires nominal change in source code.

2. Proposed solution can be implemented for any sort of programming language any platform.

3. Proposed solution works for all types of database. It works independent of database.

4. This technique also allow administrator o add rules against new as well as existing SQLIA techniques.

5. This technique also proposed a learning approach about the SQLIA techniques as it maintains a knowledge base about different techniques.

6. This technique shows effectiveness against many types of SQLIA and has the capability to have high coverage.

7. Different analysis tools can be configured depending upon the application requirements.

8. Other tools of SQLIA detection tools can be configured with the model for analysis and test.

6.3 Future Work

This technique is implemented in ASP.Net with the help of MS SQLIA tool. This scheme needs to be implemented for other platforms especially that can work independent to platform.

The proposed solution use MS SQL analyzer for possible vulnerabilities detections and page marking. The tools need to improve in such a way that any sort of analyzer can be configured for analysis.

Knowledge base maintains the techniques and knowledge about different attacks. Knowledge base should be updated using different machine learning approaches.

Bibliography

1] A.A Alfantookh, "An automated universal server level solution for SQL injection security flaw," in *Electrical, Electronic and Computer Engineering, 2004. ICEEC '04. 2004 International Conference* , 2004, pp. 131-135.

2] (2010, April) Top 10 2010-Main. [Online]. https://www.owasp.org/index.php/Top_10_2010-Main

3] M. Shooshtari A. Tajpour, "Evaluation of sql injection detection and prevention techniques," in *Computational Intelligence, Communication Systems and Networks (CICSyN), 2010 Second International Conference*, 2010, pp. 216 –221.

4] Sergey Gordeychik. (2013, December) Web Application Security Statistics.

5] Dhiraj Aryal Anup Shakya, "A Taxonomy of SQL Injection Defense Techniques," School of Computing Blekinge Institute of Technology, Karlskrona Sweden, Technical 2011.

6] M. Massrum, M.Z. Heydari A. Tajpour, "Comparison of sql injection detection and prevention techniques," in *Education Technology and Computer (ICETC), 2010 2nd International Conference*, vol. 5, 2010, pp. 174-179.

7] A. Orso W. G. Halfond, "Using Positive Tainting and Syntax Aware Evaluation to Counter SQL njection Attacks," in *14th ACM SIGSOFT international Symposium on Foundation of software engineers ACM*, 2006, pp

8] A. Orso, P. Manolios W. G.J. Halfond, "Wasp: Protecting web applications using positive tainting and syntax-aware evaluation," in *IEEE Trans. Softw. Eng*, 2008, pp. 34:65.

9] J. Viegas, A. Orso W. G. J. Halfond, "Amnesia: analysis and monitoring for neutralizing sql-injection attacks," in *20th IEEE/ACM international Conference on Automated software engineering*, New York, 2005, pp. 174–183.

10] Alessandro Orso William G.J. Halfold, "Detection and Prevention of SQL injection aattacks,".

11] Xue Ping-Chen, "SQL injection attack and guard technical research," in *Procedia Engineering*, 2011, pp. 4131-4135.

12] E. Dolstra, E. Visser M. Bravenboer, "Preventing injection attacks with syntax embeddings," *Sci. Comput. Program.*, vol. 75, pp. 473–495, July 2010.

13] J. F. Buford T. M. Chen, "Design considerations for a honeypot for sql injection attacks," in *LCN'09*, 2009, pp. 915-921.

14] Buford J. Chen T. M., "Design considerations for a honeypot for sql injection attacks," in *LCN'09*, 2009, pp.

915-921.

15] C. A. Visaggio, M. D. Penta A. Ciampa, "A heuristic-based approach for detecting sql-injection vulnerabilities in web applications," in *In Proceedings of the 2010 ICSE Workshop on Software Engineering for Secure Systems, SESS '10*, New York, NY, USA, 2010, pp. 43–49.

16] U. Sharma, D.K. Bhattacharyya D. Das, "An approach to detection of sql injection attack based on dynamic query matching," *International Journal of Computer Applications*, vol. 1, no. 25, pp. 28–34, February 2010.

17] L. Coppolino, L. Romano M. Ficco, "A weight-based symptom correlation approach to sql injection attacks.," in *In Proceedings of the 2009 Fourth Latin-American Symposium on Dependable Computing, LADC '09*, Wasington DC, 2009, pp. 9–16.

18] Avinash Kumar Singh, Ashok Singh Sairam Sangita Roy, "A Novel Approach to Prevent SQL Injection Attack Using URL Filter," *International Journal of Innovation, Management and Technology*, vol. 3, no. 5, October 2012.

19] Kai Qian Xiang Fu, "SAFELI – SQL Injection Scanner Using Symbolic Execution," in *Workshop on Testing, Analysis and Verification of Web Software*, July 21, 2008.

20] M. Shah, A. Rauf, M.A. Khan, S. Mahfooz Z. Jan, "Access control mechanism for web databases by using parameterized cursor," in *Future Information Technology (FutureTech), 2010 5th International Conference,* 2010, pp. 1-6.

21] M. Junjin, "An approach for sql injection vulnerability detection," in *Sixth International Conference on Information Technology: New Generations*, Washington, DC, USA, 2009, pp. 1411–1414.

22] T. Tzouramanis K. Kemalis, "Sql-ids: a specification-based approach for sql-injection detection," in *ACM symposium on Applied computing, SAC '08*, New York, NY, USA, 2008., pp. 2153–2158.

23] A. Clark, G. Mohay M. Kiani, "Evaluation of anomaly based character distribution models in the detection of sql injection attacks," in *Third International Conference on Availability, Reliability and Security*, Washington, DC, USA, 2008, pp. 47–55.

24] P. J. Guo, K. Jayaraman, M. D. A. Kieyzun, "Ernst. Automatic creation of sql injection and cross-site scripting attacks," in *31st International Conference on Software Engineering, ICSE '09*, Washington, 2009, pp. 199–209.

25] K. S. Lin N. Lambert, "Use of query tokenization to detect and prevent sql injection attacks," in *Computer Science and Information Technology (ICCSIT) 2010 3rd IEEE International Conference*, vol. 2, July 2010, pp. 438 –440.

26] Pal Singh Prithvi Bisht, "Improving Web Security by Automated Extraction of Web Application Intent," University of Illinois at Chicago, Chicago, Illinois, Technical 2011.

27] J. M. Chen, C. H. Liu J. C. Lin, "An automatic mechanism for sanitizing malicious injection ," in *The 9th International Conference for Young Computer Scientists*, Washington, 2008, pp. 1470–1475.

28] R. Paleari, E. Passerini M. Monga, "A hybrid analysis framework for detecting web application vulnerabilities," in *Workshop on Software Engineering for Secure Systems, IWSESS '09*, 2009, pp. 25-32.

29] A. Moosa, "Artificial neural network based web application firewall for sql injec," in *World Academy of Science, Engineering and Technology*, vol. 3, 2012, pp. 12–21.

30] A. Hur, N. Haider, F. Ahmad A. Razzaq, "Multi-layered defense against web application attacks," in *Sixth International Conference on Information Technology: New Generations*, Washington, DC, 2009, pp. 492–497.

31] M. Zulkernine H. Shahriar, "Music: Mutation-based sql injection vulnerability checking," in *The Eighth International Conference on Quality Software*, Washington, DC, USA, 2008, pp. 77–86.

32] C. E. Shyni, S. Swamynathan S. V. Shanmughaneethi, "Sbsqlid: Securing web applications with service based sql injection detection," in *web applications with service based sql injection detection Telecommunication Technologies, ACT '09*, Washington, DC, pp. 702–704.

33] K. Beznosov S. T. Sun, "Sqlprevent: Effective dynamic detection and prevention of sql injection," Technical report, March 2009. 2009.

34] L. Williams, T. Xie S. Thomas, "On automated prepared statement generation to remove sql injection vulnerabilities," *Inf. Software Technology*, vol. 51, pp. 589–598, March 2009.

35] Q. Zheng, X. Guan, Q. Wang, T. Wang Z. Zhang, "A method for detect code security vulnerability based on variables tracking with validated-tree," *Frontiers of Electrical and Electronic Engineering in China*, vol. 3, pp. 162–166, 2008.

36] P. Madhusudan , V. N. Venkatakrishnan Prithvi Bisht, "CANDID: Dynamic Candidate Evaluations for Automatic Prevention of SQL Injection Attacks0," *ACM Transactions on Information and System Security (TISSEC)* , vol. 13, no. 2, February 2010.

37] Bruce W. Weide , Paolo A. G. Sivilotti Gregory Buehrer, "Using parse tree validation to prevent SQL injection attacks," in *SEM '05 Proceedings of the 5th international workshop on Software engineering and middleware* , New York, 2005, pp. 106-113.

38] Zhendong Su Gary Wassermann, "An analysis framework for security in Web applications ," in *In Proceedings of the FSE Workshop on Specification and Verification of Component-Based Systems (SAVCBS 2004*, 2004.

39] Muthuprasanna, M. Ke Wei and S. Kothari, "Preventing SQL injection attacks in stored procedures," in *Australian Software Engineering Conference, 2006.* , 2006, pp. 18-21.

40] I. Kr¨uger R. McClure, "SQL DOM: Compile Time Checking of Conference on Software Engineering (ICSE 2005)," in *In Proceedings of the 27th International*, 2005, pp. 88–96.

41] S. Rai W. R. Cook, "Safe Query Objects: Statically Typed Objects as Remotely Executable Queries," in *In Proceedings of the 27th International Conference on Software Engineering (ICSE 2005)*, 2005.

42] Suhaimi Ibrahim, Maslin Masrom Atehfeh Tajpour, "SQL injection Prevnetion and detection Techniques," *International Journal of Advancements in Computing Technology*, vol. 3, no. 7, pp. 85-91, August 2011.

43] Fang Yu , Christian Hang , Chung-Hung Tsai , Der-Tsai Lee , Sy-Yen Kuo Yao-Wen Huang, "Securing web application code by static analysis and runtime protection," in *WWW '04 Proceedings of the 13th international conference on World Wide Web* , New York, 2004, pp. 40 - 52.

44] V. Benjamin Livshits and Monica S. Lam, "Finding security vulnerabilities in java applications with static analysis," in *SSYM'05 Proceedings of the 14th conference on USENIX Security Symposium* , Berkeley, CA, 2005, pp. 18-18.

45] S. Jain A. Anchlia, "A novel injection aware approach for the testing of database applications," in *Proceedings* *of the 2010 International Conference on recent trends in information, telecommunication and computing ITC*, Wasington DC, 2010, pp. 311-313.

46] Jagdish Halde, "SQL Injection analysis, Detection and Prevention," San Jose State University, Technical 2008.

47] L. Coppolino, L. Romano Ficco M., "A weight-based symptom correlation approach to sql injection attacks.," in *In Proceedings of the 2009 Fourth Latin-American Symposium on Dependable Computing, LADC '09*, Wasington DC, 2009, pp. 9–16.

48] A. Grim´an, N. Juristo O. Dieste, "Developing search strategies for detecting relevant experiments.," in *Empirical Software Engineering*, 2009, pp. 513-539.

49] Shrinivas Anand Panchamukhi & Abhilash Sarangi, "Blocking SQL injection in Database Stored Procedures," National Institute of Technology Rourkela, Rourkela Orissa India, Technical 2010.

50] A. Rouf, H. Javed S. ALi, "An authentication mechanism against sql injection," *European Journal of Scientific Research*, vol. 38, no. 4, pp. 604-611, 2009.

51] S.R. Jalalinia, S. Khadem K Amirathimasebi, "A Survey of sql injection defence mechanisms," in *International Conference Internet Technology and Secured Transactions ICITST 2009*, 2009, pp. 1-8.

52] Richard Sharp David Scott, "Abstracting application-level web security," in *Proceedings of the 11th international conference on World Wide Web*, New York, 2002, pp. 396-407.

53] Shih-Kun Huang , Tsung-Po Lin , Chung-Hung Tsai Yao-Wen Huang, "Web application security assessment by fault injection and behavior monitoring," in *Proceedings of the 12th international conference on World Wide Web*, New York, 2003, pp. 148-159.

54] A D Keromytis S W Boyd, "SQLrand: Preventing SQL Injection Attacks," in *Proc. Second Int'l Conf. Applied Cryptography and Network Security*, 2004.

55] Zhendong Su, Premkumar Devanbu Carl Gould, "JDBC Checker: A Static Analysis Tool for SQL/JDBC Applications," in *ICSE '04 Proceedings of the 26th International Conference on Software Engineering*, Washington, DC, USA , 2004, pp. 697-698.

56] Sam M.S. (2005, june) http://www.securitydocs.com/library/3388. [Online]. http://www.securitydocs.com/link.php?action=detail&id=3388&headerfooter=no

57] V. Haldar, D. Chandra, and M Franz, "Dynamic taint propagation for Java," in *21st Annual Computer Security Applications Conference.*, Tucson, AZ , 2005, pp. 9 pp. - 311.

58] Benjamin Livshits , Monica S. Lam Michael Martin, "Finding application errors and security flaws using PQL:

a program query language," in *OOPSLA '05 Proceedings of the 20th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, New York, 2005, pp. 365-383.

59] Fredrik Valeur, Darren Mutz, and Giovanni Vigna, "A learning-based approach to the detection of SQL attacks," in *DIMVA'05 Proceedings of the Second international conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, Springer-Verlag Berlin, Heidelberg , 2005, pp. 123-140.

60] N. Jovanovic, C. Kruegel, and E. Kirda, "Pixy: a static analysis tool for detecting Web application vulnerabilities," in *2006 IEEE Symposium on Security and Privacy*, Berkeley/Oakland, CA, 2006, pp. 6-263.

61] Stefan Kals, Engin Kirda, Christopher Kruegel, and Nenad Jovanovic, "SecuBat: a web vulnerability scanner," in *WWW '06 Proceedings of the 15th international conference on World Wide Web*, New York, NY, USA , 2006, pp. 247-256.

62] Jin-Cherng Lin and Jan-Min Chen, "An Automatic Revised Tool for Anti-Malicious Injection ," in *The Sixth IEEE International Conference on Computer and Information Technology, 2006. CIT '06.*, Seoul , 2006, p. 164.

63] M. Muthuprasanna, Ke Wei, and S. Kothari, "Eliminating SQL Injection Attacks - A Transparent Defense Mechanism," in *Eighth IEEE International Symposium on Web Site Evolution, 2006. WSE '06.*, Philadelphia, PA , 2006, pp. 22-32.

64] Tadeusz Pietraszek and Chris Vanden Berghe, "Defending against Injection Attacks through Context-Sensitive String Evaluation," in *Proceedings of Recent Advances in Intrusion Detection (RAID2005).*, 2005, pp. 3-26.

65] Lijiu Zhang, Qing Gu, Shushen Peng, and Xiang Chen, "D-WAV A Web Application Vulnerabilities Detection Tool Using Characteristics of Web Forms," in *Fifth International Conference on Software Engineering Advances (ICSEA), 2010*, Nice , 2010, pp. 501 - 507.

66] E. Ramaraj. B. Indrani, "X–log authentication technique to prevent sql injection attacks," *International Journal of Information Technology and Knowledge Management.*, vol. 4, pp. 4:323–328, , 2011.

67] Davide Balzarotti , Viktoria Felmetsger , Giovanni Vigna Marco Cova. (2013, December) http://citeseerx.ist.psu.edu/. [Online]. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.127.6909

68] Christian Beyerlein Martin Johns, "Smask: Preventing injection attacks in web applications by approximating automatic data/code separation ," in *In 22nd ACM Symposium on Applied Computing (SAC 2007)*, 2007.

69] Y. Kosuga, Kei, K. Kernel, M. Hanaoka, and M. Hishiyama, "Sania: Syntactic and Semantic Analysis for Automated Testing against SQL Injection," in *Twenty-Third Annual Computer Security Applications Conference, 2007. ACSAC 2007.*, Miami Beach, FL, 2007, pp. 107 - 117.

70] E. Merlo, D. Letarte, and G., Antoniol, "Automated Protection of PHP Applications Against SQL-injection

Attacks," in *11th European Conference on Software Maintenance and Reengineering, 2007. CSMR '07.* , Amsterdam, 2007, pp. 191-202.

71] Laurie Williams Stephen Thomas, "Using Automated Fix Generation to Secure SQL Statements," in *SESS '07 Proceedings of the Third International Workshop on Software Engineering for Secure Systems* , IEEE Computer Society Washington, DC, USA , 2007, p. 9.

72] Qinghua Zheng, Xiaohong Guan, Qing Wang, Tuo Wang Zhefei Zhang, "A method for detecting code security vulnerability based on variables tracking with validated-tree," *Frontiers of Electrical and Electronic Engineering in China* , vol. 3, no. 2, pp. 162-166. , May 2008.

73] Sherriff M. Dysart F., "Automated Fix Generator for SQL Injection Attacks," in *19th International Symposium on Software Reliability Engineering, 2008. ISSRE 2008.* , 2008, pp. 311-312.

74] Y. Yuan, D. Wijesekera, A. Stavrou A. Liu, "Sqlprob: a proxy-based architecture towards preventing sql injection attacks," in *2009 ACM symposium on Applied Computing, SAC '09*, New York, 2009, pp. 2054-2061.

75] S., Madan, S. Madan, "Shielding against SQL Injection Attacks Using ADMIRE Model," in *CICSYN '09. First International Conference on Computational Intelligence, Communication Systems and Networks, 2009. CICSYN '09*, 2009, pp. 314-320.

'6] A. Prasad Sistla , V. N. Venkatakrishnan Prithvi Bisht, "Automatically preparing safe SQL queries," in *FC'10 Proceedings of the 14th international conference on Financial Cryptography and Data Security* , Springer-Verlag Berlin, Heidelberg , 2010, pp. 272-288.

77] Corrado Aaron Visaggio , Massimiliano Di Penta Angelo Ciampa, "A heuristic-based approach for detecting SQL-injection vulnerabilities in web applications," in *Proceeding SESS '10 Proceedings of the 2010 ICSE Workshop on Software Engineering for Secure Systems* , New York, 2010, pp. 43-49.

78] Asaad Moosa, "Artificial Neural Network based Web Application Firewall for SQL Injection," *World Academy of Science, Engineering and Technology*, vol. 40, pp. 42-51, April 2010.